EpiProt: A Java Application Combining Protein Information With Epitope Prediction Software

EpiProt: A Java Application Combining Protein Information with Epitope Prediction Software

Patrick O'Mara

A Thesis in the Field of Biotechnology

for the Degree of Master of Liberal Arts in Extension Studies.

Harvard University

November 2016

Abstract

EpiProt is a bioinformatics suite designed to help users determine antigenic regions within proteins for the production of antibodies. It combines seven epitope prediction programs (five from the Immune Epitope Database, ABCPred and BcePred) with protein annotation information giving users a comprehensive view of a protein targeted for antigen design. The user can perform BLAST searches using UniProtJAPI to find similar proteins to perform pairwise and multiple sequence alignments using ClustalW, Clustal Omega, MUSCLE, T-coffee, or MAFFT. The resulting alignment is shown in a display where the user can annotate the alignments with predicted epitope regions, post-translational modifications (from PhophoSitePlus and UniProt), subcellular location (UniProt), protein processing (UniProt), secondary structure (from SIFTS/PDB) and predicted secondary structure (PsiPred and JPred). Built using Java 8, EpiProt uses Java Swing to create the graphical user interface. EpiProt is designed with a model-view-presenter (MVP) architecture pattern requiring an internet connection to run and collect data via Representational State Transfer (RESTful) for viewing.

Table of Contents

List of Tables

List of Figures

Chapter I:

Introduction

Understanding the molecular mechanisms that drive normal physiology and disease has been central to biology for the better part of the last century (Weaver, 1970). The complexity and yet eloquence observed in these processes is truly awe-inspiring. However, the main problem in trying to understand these processes is not so much trying to understand the complexity behind them (although difficult in it's own right), it is simply trying to observe them happening in their natural state. Many techniques and systems have been created in order to observe these processes as they take place. There is one family of assays that has taken advantage of biology itself to help understand these processes (Bordeaux et al., 2010). These assays would not be possible without the help of one protein, the antibody.

In this thesis I will explain what an antibody is and what it does. I will explain how scientists exploit the natural characteristics of the antibody in order to help understand the molecular processes involved. This will include an explanation of how they go about designing and manufacturing them. After the biological background is complete, I will then delve into the technology and software available for designing these antibodies to bind to specific regions called epitopes. Next I will explain my program, EpiProt (**Epi**tope + **Prot**ein) by describing how it is designed, the components that construct it, the software tools it accesses, how it is similar to current programs and how it differs. Lastly I will discuss its advantages and illustrate them by running proteins with known epitope regions through EpiProt.

Antibodies and Antigens

An antibody is a protein of the adaptive immune system made by B-cells that is able to identify and neutralize pathogens and other foreign objects. This ability to "identify and neutralize" is possible because of two main characteristics: 1) a variable region, which binds specifically to a target, and 2) a constant region which certain immune cells recognize allowing for phagocytosis and other immune defense signaling (Chaplin, 2016.). It is the first characteristic that really defines the antibody, as the variable region determines what the antibody will bind to.

Broadly speaking, the object the antibody binds to is called an antigen. Although, technically speaking, an antigen is much more than an object bound by an antibody; it is also an object that induces an immune response. Antigens are usually some kind of biological molecule such as a carbohydrate, lipid or protein. Sometimes it can be a combination of any or all of these molecules (Parkin, Jacqueline, & Bryony, 2001). For the purpose of this paper, we will consider antigens to be proteins. Thinking of antigens as proteins will become important when discussing the next concept, epitopes.

Epitopes

The antigen region an antibody actually binds to is called an epitope. In general an epitope is around 4-12 amino acids in length. There are two different types of epitopes: 1) discontinuous (sometimes referred to as conformational) epitopes and 2) continuous (sometimes referred to as linear) epitopes. A discontinuous epitope is a region of a protein where an antibody binds to, however the amino acids that the antibody binds to are not in sequential order. This is due protein folding, which allows non-sequential amino acids to come close enough for antibody binding. Linear epitopes are those where an antibody binds to amino acids in sequential order (Buus et al., 2012). Knowing the differences between these two types will become important when talking about epitope prediction software and whether it predicts continuous or discontinuous epitopes.

In Vivo Generation of Antibodies

When a protein antigen enters the body, the immune system acts to break this antigen down into small fragments called peptides. These peptides are presented to CD4+ T-cells by professional antigen presenting cells. B-cells, the cells that generate antibodies, are a type of antigen presenting cells that can endocytose these peptides with the help of B-cell receptors (which are really antibodies bound to the membrane of a B-cell). These peptides are then displayed to T-cells by class II major histocompatibility complex proteins (MHC II). If a B-cell displays a peptide to a T-cell that also recognizes the peptide, the T-cell helps activate the B-cells to proliferate and create antibody against the non-self protein (Miller & Frederick, 1995).

It is important that an individual B-cell is activated to proliferate because each B-cell in the body creates a unique antibody sequence. More specifically, each B-cell can generate an unique antibody with an unique variable region due to V(D)J recombination. This ability for CD4+ T-cells to clonally select B-cells allows the immune system to proliferate those B-cells, which produce the right antibody against the antigen (Rajewsky, 1996). This concept of clonal selection is important to remember later on when discussing manufacturing antibodies.

There are five main classes of antibodies or immunoglobulins: IgM, IgG, IgA, IgD and IgE. These antibodies perform various functions such as agglutination, complement activation, neutralization, opsonization and precipitation. Certain classes of immunoglobulins can perform these certain functions more effectively. IgM, for example, is very effective at activating complement, while the Fc region on IgE preferentially binds to mast cells and basophils to fight off parasitic pathogens. For the purpose of this thesis we will concentrate on IgG. This class of antibody is the most common type of antibody found in the body. It is found in blood and extracellular fluid and can perform all the functions mentioned prior (Janeway, 2001).

Manufacturing Antibodies for Applications

There are two main types of antibodies to consider when making antibodies, polyclonal and monoclonal. Polyclonal, in the truest sense of the word, are antibodies that are derived from different B-cells. They target the same antigen but often bind to different epitopes. Monoclonal antibodies are derived from one B-cell line and therefore target one epitope (Lipman, Jackson, Trudel, & Weis-Garcia, 2005).

The production of polyclonal antibodies starts with immunizing an animal with an antigen. The serum of the animal is then collected and purified using Protein A purification. The resulting purification can be purified again with the antigen itself in what is called affinity purification (Harlow & Lane, 1999).

There are a number of approaches to creating monoclonal antibodies. Some of the more recent advancements involve phage display (Winter, Greg, Griffiths, Hawkins, & Hoogenboom, 1994), p. 433-455) and proteomics (Cheung et al., 2012). For the focus of this paper, we will consider one of the oldest and most widely used methods, the hybridoma method (Milstein, 1999) In this method a scientist immunizes an animal (often a mouse) with an antigen. The B-Cells are collected and fused with myeloma cells (of the same species) thereby immortalizing the B-Cells. These are then cloned into a cell line that creates one type of antibody. This cell line can be stored which allows one to generate more of the antibody if needed.

Both polyclonal and monoclonal production methods start with immunizing an animal with an antigen. There are a number of different types of antigens to consider when designing them for injection. The most common types of antigens are recombinant proteins, whole cell extract and peptides (Harlow & Lane, 1999), p. 64-104).

Recombinant protein antigens are full length or large segments of the target protein usually expressed through a bacterial (Rosano & Ceccarelli, 2014) or mammalian system (Wurm, 2004). This type of antigen can allow the immune system to splice and target many different epitopes on the protein. It also allows the immune system to target discontinuous epitopes as protein antigens can form secondary and tertiary structures (Ebersbach & Geisse, 2012). Fragmenting the protein into many epitopes can be advantageous because it increases the likelihood of an antibody binding to the target protein. However, the disadvantage is that the antibodies that are made may be non-specific to the target protein (especially if the target protein is part of protein family with very similar sequence or folding), and the epitope may be buried within the protein when folded (Pathak & Palan, 2005, pp. 68–69). The latter problem can be a concern when trying to produce antibodies for applications where it is targeting a non-denatured version of the target protein, however, this can be mitigated by the fact that protein antigens can allow for antibodies to bind to discontinuous epitopes (Ebersbach & Geisse, 2012). Since many antibodies are generated as a result of a protein immunization, collecting polyclonal antibodies from serum is often not performed as their use increases the chance of non-specific binding. Instead monoclonal antibodies are often produced for this immunization strategy (Trier, Hansen, & Houen, 2012, pp. 136–144).

Whole cell immunization is a strategy where one over-expresses a target protein in cells, and injects those cells into an animal (Harlow & Lane, 1999, pp. 100–104). This type of immunization is used primarily for cell surface proteins. It is often difficult to generate antibodies toward surface proteins because these proteins can transverse the cell membrane several times forming loops. Antibodies that might bind to such loops may not bind to the same protein region if the protein is denatured. Transfecting a target protein in cells allows the protein to be presented to the immune system in a similar configuration as would be observed in nature. The disadvantage is that many antibodies might be produced that are not specific to the target protein (Tamura & Chiba, 2009).

The last immunization strategy, and really the focus of this paper, is peptide immunization. In this strategy a scientist identifies regions in a target protein that are considered prime epitope regions. How a scientist identifies these regions will be discussed in the next section. Once the regions are identified, they are synthesized as short peptides, usually between 8 and 20 amino acids in length, and injected into the animal (Trier, Hansen, & Houen, 2012, pp. 136–144). These peptides are often conjugated to a carrier protein such as Keyhole limpet hemocyanin (KLH) to promote a stronger immune response to the peptide (Harlow & Lane, 1999, p. 85).

There are advantages to peptide immunization. One advantage is a scientist can target specific regions in a target protein. This allows the scientist to design epitopes that are specific to the target protein, thus making an antibody specific to the target protein. This also allows the scientist to target specific sites, such as post-translational modification sites, in order to create site-specific antibodies. Peptide antigens are also easier to produce than protein or whole cell antigens on a large scale, which makes it an attractive option for manufacturing purposes (Trier et al., 2012, pp. 136–144).

There are a few drawbacks to peptide immunization. Since there may be only a few epitope regions within a peptide, the likelihood of the animal forming an antibody specific to the protein decreases. To overcome this problem, scientists will often synthesize and immunize the animal with several peptides from the target protein to increase their success (Burns & Robert, 2005). Another disadvantage is that peptide antigens are continuous epitopes. This means the antibody generated against the peptide antigen may not bind to the epitope region when the protein is folded as the epitope could be buried (Trier et al., 2012, pp. 136–144).

Antigen Design Principles

Before an animal can be immunized, the antigen has to be designed. Since peptide antigens are easier to synthesize, we will concentrate on peptide antigen design for the purpose of this thesis. When designing peptide antigens, a scientist identifies regions within a protein that are projected to be epitopes an antibody may bind to. There is a great deal of discussion as to what makes a good antigen. The majority of papers claim protein characteristics like amino acid composition (Kolaskar & Tongaonkar, 1990; Larsen, Lund, & Nielsen, 2006), surface accessibility (Emini, 1985) and even beta turns (Chou, 1978) are the key to designing antigens. In the end a “good” antigen needs to be 1) accessible, 2) immunogenic and 3) specific to its target protein(s) (Pisitkun, Hoffert, Saeed, & Knepper, 2012). First, I will explain these concepts then I will describe the antigen design process.

Accessibility

It makes sense that in order for an antibody to bind to an epitope, the epitope must be exposed or accessible. Depending on the application it can be difficult for the antibody to find the epitope if that epitope is buried within the protein. In an application such as a western blot the target protein is denatured which exposes continuous epitopes. Therefore finding accessible regions for this application is not necessary. In other applications such as immunohistochemistry, immunofluorescence and flow cytometry the proteins are typically non-denatured and might possess cross-linking amino acids due to formaldehyde fixation (Shi, Cote, & Taylor, 1997). These types of applications depend on the scientist identifying regions that might be exposed (Trier et al., 2012). Typically a scientist will target regions such as beta-turns, as they are more likely to be on the protein surface. Knowing the 3D structure of protein can help identify beta-turn regions, however not many proteins have 3D structure available, and even if they do, it may not be complete. Epitope prediction methods such as "Chou and Fasman beta turn prediction" can predict beta-turn regions (Chou, 1978). Others such like Emini surface accessibility (Emini, 1985), can help with finding accessible regions. The combination of these tools can help scientists get an idea of the most accessible regions in a folded protein.

Immunogenicity

If the antigen is not immunogenic, it will not be able to elicit the immune response needed in order to produce antibodies. Designing antigens with a certain amino acid composition can help as some amino acids are considered to be more immunogenic than others (Kringelum, Nielsen, Padkjær, Lund, 2013). To make the antigens more immunogenic, scientists normally conjugate the designed antigen to an extremely immunogenic carrier protein such as KLH. Other immunization tactics such as boosting can aid in immunogenicity (Harlow & Lane, 1999).

Specificity

One principle that holds true for all antibody assays is that the antibody must be specific to the target protein(s) it is designed to bind to. Therefore it is imperative that the antigen is unique to its target protein(s). Finding specific epitope regions is accomplished in two steps. First, a scientist will perform a BLAST (Altschul, Gish, Miller, Myers, & Lipman, 1990) (Basic Local Alignment Search Tool) search of the protein to a protein database such as Swiss-Prot (Boeckmann et al., 2003). The scientist will identify the proteins most similar to the query. Second, a scientist will perform a multiple sequence alignment (MSA) (Edgar & Serafim, 2006) of the target protein to the proteins they would like to compare the target protein to (i.e. the most similar proteins from the BLAST). From the alignment(s) the scientist is able to determine similar regions among the proteins, which they are able to target or avoid based on what they wish to accomplish (Pisitkun et al., 2012).

These design principles are the basis for antigen design. There are a lot of opinions of what makes the best antigen based off these principles. This thesis project is focused on providing the tools and information a scientist needs in order to decide the best antigenic regions. However, for the sake of clarity, it is necessary to go over the general process of antigen design. Knowing the process involved will help give a better understanding of the tools necessary for antigen design.

Antigen Design Process

As mentioned in the “Specificity” section, The first step in the antigen design process is performing a BLAST search of the target protein and a subsequent multiple sequence alignment (MSA). There are three main types of alignments a scientist can perform in this initial phase:

Intra-species Alignment

When a scientist performs a BLAST and MSA of a target protein against proteins that originate from the same organism, this can be considered an intra-species alignment. Some refer to it as an paralogous protein alignment, but this can be misleading as the most similar protein to the target protein is not necessarily a true paralog. It is sometimes necessary to compare the target protein against several proteins to find specific regions, especially if that protein belongs to a large family of proteins. Although one can certainly align the target protein to many similar protein in one alignment, it is often advantageous to perform pairwise alignments in order to really compare the target protein to a similar protein (Angeletti, 1999).

Figure 1 below illustrates a section of an intra-species alignment between human AKT1 to human AKT2. In this example, designing a peptide antigen in the blue region will likely generate an antibody specific to AKT1 (UniProt id: P31749) and not to AKT2 (UniProt id: P31751). However, designing a peptide antigen in the red region would likely generate an antibody that could recognize both proteins.

```
CLUSTAL O(1.2.1) Pairwise sequence alignment
SP|P31749|AKT1_HUMAN MSDVAIVKEGWLHKRGEYIKTWRPRYFLLKNDGTFIGYKERPQDVDQREAPLNNFSVAQC 60
SP|P31751|AKT2_HUMAN MNEVSVIKEGWLHKRGEYIKTWRPRYFLLKSDGSFIGYKERPEAPDQTLPPLNNFSVAEC 60
                     *.:*:::***********************.**:********:  **   ********:*
```

Figure 1. Pairwise sequence alignment of human Akt1 and human Akt2.

Inter-species Alignment

Often a scientist will want an antibody to bind to the orthologous protein in other species (Rahman et al., 2015). For example, a scientist may want an antibody to bind to human Sortilin (UniProt id: Q99523) and mouse Sortilin (UniProt id: Q6PHU5) at the C-term. To do this they would perform an alignment between the two proteins, and pick the region similar to both. Figure 2 below shows an alignment of these two proteins at the C-term. The red text is a region that is similar between the two proteins, indicating a region to target if the scientist wants an antibody to target both human and mouse Sortilin. On the other hand, the blue text shows a dissimilar region between human and mouse Sortilin, indicating a region to target if antibody specific to human Sortilin is desired.

```
CLUSTAL O(1.2.1) pairwise sequence alignment
SP|Q99523|SORT_HUMAN YVCGGRFLVHRYSVLQQHAEANGVDGVDALDTASHTNKSGYHDDSDEDLLE 831
SP|Q6PHU5|SORT_MOUSE YVCGGRFLVHRYSVLQQHAEADGVEALDST----SHAKSGYHDDSDEDLLE 825
                     *********************:**:.:*:        **************
```

Figure 2. Pairwise sequence alignment of human Sortilin and mouse Sortilin.

Isoform Alignment

Proteins can sometimes be expressed in a number of varieties called isoforms. Isoforms are a result of mRNA splicing which after translation expresses different protein sequences, even though the proteins come from the same gene. Whether it is desirable for an antibody to bind to all the isoforms of a target protein or not, it is always wise to check if the antigen is isoform specific (Divan & Royds, 2013). Figure 3 below shows two regions in one isoform. The red region is common among both isoforms, human BET1 isoform 1 (UniProt id: O15155) and BET1 isoform 2 (UniProt id: O15155-2). The blue text highlights a region specific to isoform 1.

```
CLUSTAL O(1.2.1) Pairwise sequence alignment
SP|O15155|BET1_HUMAN   NKLLAEMDSQFDSTTGFLGKTMGKLKILSRGSQTKLLCYMMLFSLFVFFIIYWIIKLR 118
SP|O15155-2|BET1_HUMAN NKLLAEMDSQFDSTTGFLGLCPM----------------------------------- 83
                       *******************
```

Figure 3. Pairwise sequence alignment of isoforms 1 and 2 of BET1.

Protein Annotation

The next step in the process is mapping protein information to the alignments. Some of the information that can be mapped to the sequence is structure, secondary structure prediction, post-translational modifications, subcellular location, topology, protein interaction sites, and protein processing. There are many sources from which one can gather this information. I will go over some of the protein data that can be collected, highlighting those used by EpiProt.

Protein structure. Knowing the structure of a protein can be extremely advantageous in antigen design. As mentioned previously, in certain applications the antibody might need to bind to protein in a folded state. By knowing the structure, one can identify the exposed regions that an antibody could bind to. When considering protein structure, there are two types: secondary and tertiary, and there are two methods to obtaining protein structure information: real structure files and structure prediction programs.

The first structure type, secondary structure, is easily mapped to the protein sequence in a linear format as each amino acid can be assigned a secondary structure type such as alpha-helix (H), beta-strand (E) and turn (T). A scientist will often mark or map the secondary structure to the primary sequence in order to gain a better visual of what the structure is like in a certain protein. For example:

```
CLUSTAL O(1.2.1) Pairwise sequence alignment
JPred                  HHHHHHHHHHHHHHH-HHHHHHHHHHHHHH-----HHHHHHHHHHHHHHHHHHHHH--
PsiPred                HHHHHHHHHHHHHCCCCHHHHHHHHHHHHHHCCCCHHHHHHHHHHHHHHHHHHHHHCC
SP|O15155|BET1_HUMAN   NKLLAEMDSQFDSTTGFLGKTMGKLKILSRGSQTKLLCYMMLFSLFVFFIIYWIIKLR 118
```

Figure 4. BET1 with JPred and PsiPred secondary structure prediction aligned above the primary sequence. Note how well the prediction programs agree with each other.

When targeting regions for antigen design, it is generally considered that turn regions are better regions to target, as the chance of them being exposed is higher. Also, the residues within the turn regions could be interacting less with other residues, making them more accessible (Hancock & OReilly, 2005). The blue amino acids in Figure 4 show a potential turn region one might consider during antigen design. At this region both secondary structure prediction (SSP) programs show a gap between the two helices, indicating that this could be a turn region between within the protein.

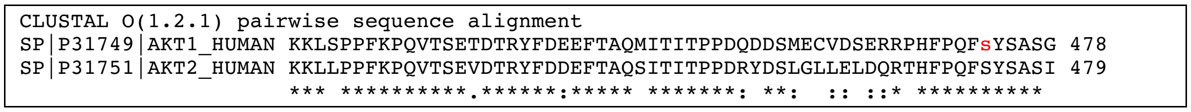
In Figure 4, SSP was used to gather secondary structure information as the target protein, BET1, does not have a structure file available on PDB (Berman et al., 2000). In fact many proteins do not have a structure file available, and even if it did they often times do not cover the entire protein. The existence of a structure file not only allows a scientist to explore the secondary structure of a protein, it allows a scientist to explore the tertiary structure. Secondary structure refers to regions of a protein that interact along the peptide backbone such as alpha helices and beta sheets. Tertiary structure refers to the overall 3D structure of the protein, or how the protein folds (Stephen Stoker, 2015). Knowing the tertiary structure can allow a scientist to determine exposed regions, or regions that are accessible to the target protein (Hoge, 2003). It is difficult to display how exposed amino acids are from a 1D perspective, however it is possible. Essentially, when looking at which regions are the most exposed in a 3D image, a scientist is looking for amino acids with a high solvent accessible surface area (SASA). SASA can be calculated from a protein structure (Ali, Hassan, Islam, & Ahmad, 2014) or it can be predicted (Emini, 1985). Since each amino acid can get a SASA value, that value can be placed on a scale and be displayed in alignment with the protein sequence.

Topology and protein interaction information. Two related pieces of information that are often incorporated into antigen design are membrane topology and protein interactions. Protein membrane topology shows the parts of a protein that span a membrane (von Heijne, 2006). If a part of a protein is buried within a membrane, that region will be very difficult for an antibody to bind to in applications where the membrane is left intact. Knowing the topology of a protein can allow one to design antigens in non-transmembrane regions. It also facilitates the design of the antigen in an intracellular or extracellular region, which can be beneficial depending on the application.

The second part, protein interaction, allows a scientist to determine if certain regions within a protein are hidden when they are interacting with other molecules such as DNA, RNA, other protein and substrates. Once again which region should be targeted for antigen design is dependent on the antibody's application. If the purpose of the antibody is to inhibit the interaction, targeting the interaction site is logical. If the purpose of the antibody is to locate the protein in its natural state without disrupting its function, targeting far away from the interaction site make sense.

Post-translational modifications and mutation sites. If an antigen is designed in a region where an amino acid is post-translationally modified or mutated, it has to be assumed that the modification or mutation will affect the binding of the antibody. Therefore if one is to target the site of the modification or mutation, one must determine what they want the antibody to indicate in testing. If the goal is to create an antibody that detects a phosphorylation at a particular amino acid, the antigen should be designed with the phosphorylation. Likewise if the goal is to create an antibody to recognize a mutated form of a protein, that antigen should express the mutation.

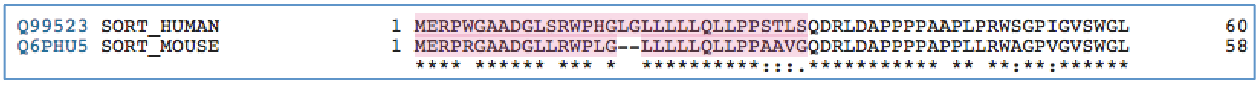
Marking post-translational modifications (PTMs) and mutations can ensure that antigen design either incorporates or avoids the site. Figure 5 below shows an example of what this might look like. Shown is a portion of an alignment of human AKT1 and human AKT2. In red and lowercase is S473, an amino acid that, according to PhosphoSitePlus (Hornbeck et al., 2012), is phosphorylated.



```
CLUSTAL O(1.2.1) pairwise sequence alignment
SP|P31749|AKT1_HUMAN KKLSPPFKPQVTSETDTRYFDEEFTAQMITITPPDQDDSMECVDSERRPHFPQFsYSASG 478
SP|P31751|AKT2_HUMAN KKLLPPFKPQVTSEVDTRYFDDEFTAQSITITPPDRYDSLGLLELDQRTHFPQFSYSASI 479
                     *** **********.******:***** *******: **:  :: ::* **********
```

Figure 5. Pairwise sequence alignment of human Akt1 and human Akt2. Serine at position 473, which is phosphorylated, is indicated in red and lowercase.

Protein processing. Protein processing is another consideration in antigen design. Proteins can be cleaved into smaller fragments, like signal peptides or propeptides. It is therefore important to know if the designed antigen is within a cleaved region. For most purposes it is not preferred to design an antigen within a cleaved region to ensure that the antibody binds to the protein. For example, figure 6 below shows an alignment between human and mouse Sortilin at the N-term. The highlighted signal peptide is cleaved off during protein processing. Designing an antigen within the signal peptide region would result in an antibody that does not bind to the mature protein.



```
Q99523 SORT_HUMAN          1 MERPWGAADGLSRWPHGLGLLLLLQLLPPSTLSQDRLDAPPPPAAPLPRWSGPIGVSWGL     60
Q6PHU5 SORT_MOUSE          1 MERPRGAADGLLRWPLG--LLLLLQLLPPAAVGQDRLDAPPPPAPPLLRWAGPVGVSWGL     58
                             **** ****** *** *  **********:::.*********** ** **:**:******
```

Figure 6. The N-term portion of a Clustal Omega alignment between human Sortilin (UniProt id Q99523) and mouse Sortilin (UniProt id Q6PHU5) performed on UniProt alignment tool. The highlighted region is a signal peptide cleaved off during processing.

General guidelines. Whether one can map all of the aforementioned protein information to the protein sequence or not, the sequence itself can provide valuable information for antigen design. Choosing a region with a heterogeneous mixture of amino acids is normally desired. Designing an antigen in a region where the amino acids are highly repetitive or homogenous can produce non-specific antibodies. Avoiding such regions can reduce producing non-specific antibodies. To reduce non-specific interactions further, a peptide BLAST can be performed on the region to see if any proteins have similar regions. Since antibody epitopes are usually larger than 5 amino acids in length, it is wise to choose regions that are longer than 5 amino acids that are the different between the target protein and the other BLAST hits ("Antibody Basics | Novus Biologicals," 2016.).

Antigen Design Summary

Even with a guide on how to design antigens for the production of antibodies, it can be a challenging endeavor to compile and condense all the data into one view in order to design antigens appropriately. Becoming good at antigen design can take years of practice, and requires one to thoroughly understand biochemistry, molecular biology, immunology, BLAST and alignment software. Even then how one designs antigens can vary between antibody application, and on the opinion of the designer on what regions are the best to target for antigen design. The challenge of EpiProt is to allow a user the ability to combine all the information they might want to see in one view while allowing them the flexibility to decide how to design antigens.

Epitope Prediction Sources

EpiProt would not be complete without integrating epitope prediction programs (EPP). In all ten epitope prediction programs were investigated. Some could not successfully be integrated which will be explained below. A summary can be found in the appendix on table 1.

Unfortunately some epitope prediction tools could not be integrated in EpiProt. BepiPred appeared to be straightforward as it uses the same basic command line code as the other epitope prediction programs from http://tools.immuneepitope.org/bcell/. However, when running the command, it produces a “float division by zero” error. It seems as though there is some problem on the server side producing this error, therefore integration will be impossible until IEDB (Immune Epitope Database) figures out what the problem is.

Ellipro and Discotope epitope prediction were initially intended to be included in EpiProt. Some of the pieces are built, including the service class that retrieves the information. Other pieces could not be built in time including the view and presenter components (explained in subsequent sections). Therefore, I could not include Ellipro and Discotope into the completed version for this thesis. However, since some of the piece are built already, integrating them should not take that long. I am hoping to integrate them after drafting the final version of this thesis.

Similar Approaches and Deficiencies

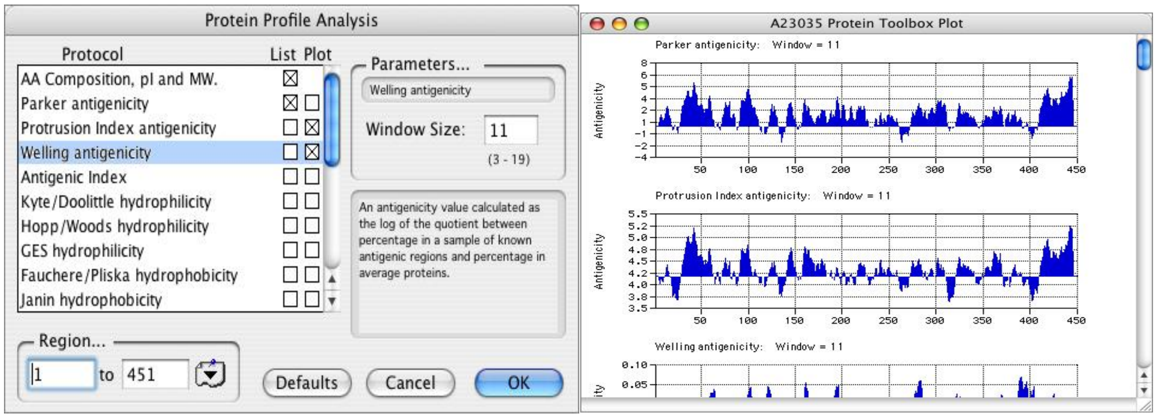
Scientists are limited when it comes to programs that can combine protein and epitope prediction information. Usually programs are geared toward showing one particular type of information over another. Before starting this project, I investigated some of the most commonly used and accessible programs that show protein and/or epitope information. Below I describe those programs I explored as comparison tools (please refer to Table 2 for a summary).

MacVector

MacVector is a well-known tool used by a number of institutions and labs that helps molecular biologists compute and analyze biological data. In particular it is known for analyzing sequence data when trying to perform MSA, phylogenetic tree analysis, primer design, restriction analysis and protein analysis. I tried to find software that was free and accessible. Although MacVector is not free, it is used in many labs since 1994 (Olson, 1994), I considered it highly accessible, and therefore a comparable piece of software. A description of MacVector and its features can be found here http://macvector.com/MacVector/macvector.html.

MacVector has several tools that are able to assist in antigen design. Most of these tools are organized in a part called the Protein Profile Analysis toolkit. For epitope prediction it has Parker antigenicity, protrusion antigenicity, Welling antigenicity, and several hydrophilicity/hydrophobicity prediction tools. It can also show potential transmembrane regions and SSP. To compare the target protein to similar proteins, it provides an NCBI BLAST tool. Once the BLAST is complete, one can align the protein using MUSCLE (Edgar, 2004), T-Coffee (Notredame, Higgins, & Heringa, 2000), or ClustalW (Thompson, Gibson, & Higgins, 2002). The results of all this analysis returns in separate windows from the main protein sequence as displayed in figure 7. This link is a guide for some of the protein analysis tools available:

http://macvector.com/MacVector/proteinanalysis.html.


Protein Profile Analysis
Parameters...
Region...
Defaults
Cancel
OK
A23035 Protein Toolbox Plot
Parker antigenicity: Window = 11
Protrusion Index antigenicity: Window = 11
Welling antigenicity: Window = 11
Antigenicity

Protocol	List	Plot
AA Composition, pI and MW.	☒	
Parker antigenicity	☒	☐
Protrusion Index antigenicity	☐	☒
Welling antigenicity	☐	☒
Antigenic Index	☐	☐
Kyte/Doolittle hydrophilicity	☐	☐
Hopp/Woods hydrophilicity	☐	☐
GES hydrophilicity	☐	☐
Fauchere/Pliska hydrophobicity	☐	☐
Janin hydrophobicity	☐	☐

Welling antigenicity

Window Size: 11 (3 - 19)

An antigenicity value calculated as the log of the quotient between percentage in a sample of known antigenic regions and percentage in average proteins.

Region: 1 to 451

Figure 7. (Left) The Protein Profile Analysis toolkit allows the user to run multiple programs in order to study protein characteristics. (Right) The displayed results after running Parker antigenicity, Protrusion Index antigenicity, and Welling Antigenicity.

Although MacVector has many tools that could help in antigen design, since the results return in separate diagrams and not against the protein sequence alignment, it is hard to visualize all the data together. One of the primary goals of EpiProt is to bring all the data together so the user does not have to look at separate windows or different resources in order to design antigens.

Jalview

Jalview (Waterhouse, Procter, Martin, Clamp, & Barton, 2009) is a "A free, open source program developed for the interactive editing, analysis and visualization of pairwise sequence alignments. It can also work with sequence annotation, secondary structure information, phylogenetic trees and 3D molecular structures." (http://www.jalview.org/About). Its desktop version is similar to EpiProt in that it is Java based, requiring Java 1.6 or later. JalviewLite is a base version that uses a JavaScript

API. It is designed to run on web sites as a bioinformatics tool. When comparing EpiProt, I compared it to the desktop version, as EpiProt is a desktop program.

Opening Jalview brings you to a large window in which you can open three smaller windows. The first window provides a display for MSA, SSP and other analytical information. The second window displays a phylogenetic tree information, and a third shows a PDB 3D structured via Jmol ("Jmol: an open-source Java viewer for chemical structures in 3D," 2016.). These windows can be viewed in figure 8.

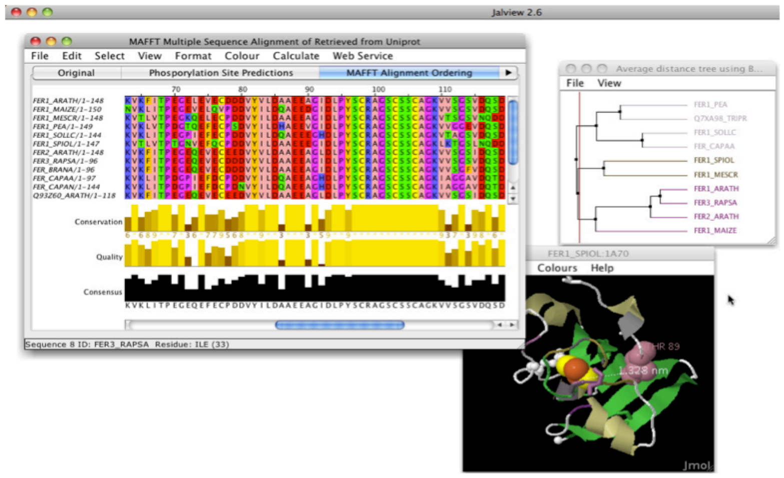


Figure 8. A view of Jalview and its three windows. Top left- first window with MSA, SSP and other analytical data. Top right- Phylogenetic Tree Clustal analysis. Bottom right- 3D visualization of the protein using Jmol.

EpiProt is not designed to display phylogenetic tree or 3D structure information in the first version. Phylogenetic tree data in particular may never be displayed as BLAST and MSA tools provide a much more detailed view into similarities between proteins, especially at the amino acid level where antigen design is performed. Analyzing 3D structure, on the other hand, could be quite beneficial. Visualizing a 3D view of a protein in Jmol can allow scientists to recognize protrusion regions on the antibody, and in later versions this might be included.

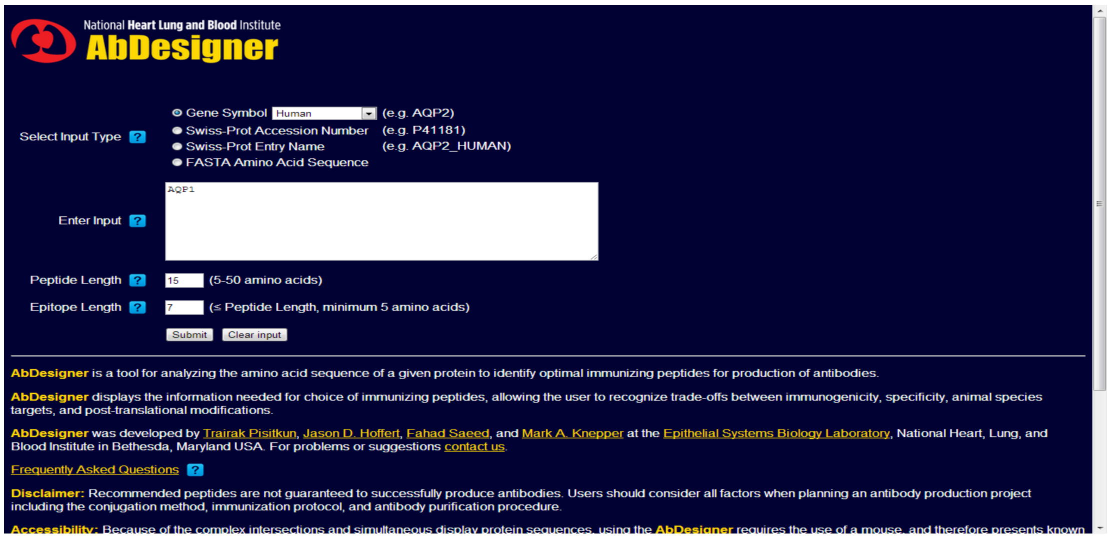
For the purposes of antigen design, Jalview lacks a lot of functionality in comparison with EpiProt. It does not have any epitope prediction programs, perform BLAST searches, display PTM information or show protein topology. That said, the purpose of Jalview is to be a generalized bioinformatics toolkit in order to visualize protein data, not to help in antigen design. The reason I compared it to EpiProt is because it provides a rich environment to explore general protein information, which is crucial to antigen design.

At the core of Jalview is JABAWS or **JA**va **B**ioinformatics **A**nalysis **W**eb **S**ervices (Troshin, Procter, Barton, 2011). JABAWS is a free, well known bioinformatics software that combines 5 MSA programs: MUSCLE, ClustalO (Sievers & Higgins, 2014), ClustalW, MAFFT (Katoh, Misawa, Kuma, & Miyata, 2002), T-Coffee, 1 SSP program: Jpred (Cuff, Clamp, Siddiqui, Finlay, & Barton, 1998), 4 protein disorder programs: DisEMBL (Linding, Jensen, et al., 2003), IUPred (Dosztányi, Csizmok, Tompa, & Simon, 2005), Jronn (Uversky, Kuznetsova, Turoverov, & Zaslavsky, 2015), GlobPlot (Linding, Russell, Neduva, & Gibson, 2003), and 1 amino acid alignment conservation program: AACon ("AACon Web Service," 2016.). When designing EpiProt, it made sense to leverage JABAWS for many of its services. Although EpiProt does not need protein disorder programs in version one, it does need MSA and SSP. Incorporating JABAWS into EpiProt made it very easy to integrate multiple programs at the same time.

AbDesigner

AbDesigner is the only comparison program analyzed whose purpose is the same as EpiProt. It is described as a "tool for analyzing the amino acid sequence of a given protein to identify optimal immunizing peptides for production of antibodies.”

(https://hpcwebapps.cit.nih.gov/AbDesigner/). The input, figure 9, is very similar to EpiProt. Also, it annotates the protein with topology, glycosylation sites, modifications, Chou-Fasman SSP and Kyte-Doolittle Hydrophobicity information. It splits the protein into peptides and ranks those peptides based on three criteria: “Ig-score” (their custom immunogenicity scale), “uniqueness” and “conservation”. These peptides are displayed, figure 10, as a list below the protein sequence and in alignment with the protein sequence.



Select Input Type ?

● Gene Symbol Human (e.g. AQP2)
● Swiss-Prot Accession Number (e.g. P41181)
● Swiss-Prot Entry Name (e.g. AQP2_HUMAN)
● FASTA Amino Acid Sequence

Enter Input ? AQP1

Peptide Length ? 15 (5-50 amino acids)

Epitope Length ? 7 (≤ Peptide Length, minimum 5 amino acids)

AbDesigner is a tool for analyzing the amino acid sequence of a given protein to identify optimal immunizing peptides for production of antibodies.

AbDesigner displays the information needed for choice of immunizing peptides, allowing the user to recognize trade-offs between immunogenicity, specificity, animal species targets, and post-translational modifications.

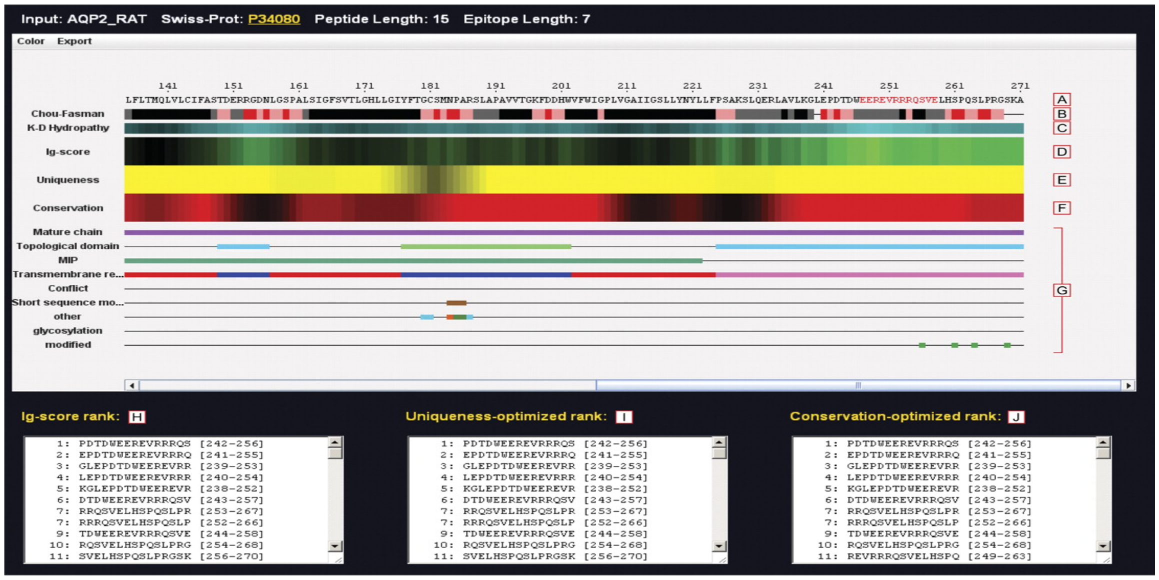
AbDesigner was developed by Trairak Pisitkun, Jason D. Hoffert, Fahad Saeed, and Mark A. Knepper at the Epithelial Systems Biology Laboratory, National Heart, Lung, and Blood Institute in Bethesda, Maryland USA. For problems or suggestions contact us.

Frequently Asked Questions ?

Disclaimer: Recommended peptides are not guaranteed to successfully produce antibodies. Users should consider all factors when planning an antibody production project including the conjugation method, immunization protocol, and antibody purification procedure.

Accessibility: Because of the complex intersections and simultaneous display protein sequences, using the **AbDesigner** requires the use of a mouse, and therefore presents known

Figure 9. AbDesigner input screen. The user can input the protein in 4 different ways: gene symbol, accession number, accession name and FASTA format. Parameters like peptide length and epitope length can be chosen as well.



Input: AQP2_RAT Swiss-Prot: P34080 Peptide Length: 15 Epitope Length: 7

Ig-score rank: H

```
 1:  PDTDWEEREVRRRQS  [242-256]
 2:  EPDTDWEEREVRRRQ  [241-255]
 3:  GLEPDTDWEEREVRR  [239-253]
 4:  LEPDTDWEEREVRRR  [240-254]
 5:  KGLEPDTDWEEREVR  [238-252]
 6:  DTDWEEREVRRRQSV  [243-257]
 7:  RRQSVELHSPQSLPR  [253-267]
 7:  RRRQSVELHSPQSLP  [252-266]
 9:  TDWEEREVRRRQSVE  [244-258]
10:  RQSVELHSPQSLPRG  [254-268]
11:  SVELHSPQSLPRGSK  [256-270]
```

Uniqueness-optimized rank: I

```
 1:  PDTDWEEREVRRRQS  [242-256]
 2:  EPDTDWEEREVRRRQ  [241-255]
 3:  GLEPDTDWEEREVRR  [239-253]
 4:  LEPDTDWEEREVRRR  [240-254]
 5:  KGLEPDTDWEEREVR  [238-252]
 6:  DTDWEEREVRRRQSV  [243-257]
 7:  RRQSVELHSPQSLPR  [253-267]
 7:  RRRQSVELHSPQSLP  [252-266]
 9:  TDWEEREVRRRQSVE  [244-258]
10:  RQSVELHSPQSLPRG  [254-268]
11:  SVELHSPQSLPRGSK  [256-270]
```

Conservation-optimized rank: J

```
 1:  PDTDWEEREVRRRQS  [242-256]
 2:  EPDTDWEEREVRRRQ  [241-255]
 3:  GLEPDTDWEEREVRR  [239-253]
 4:  LEPDTDWEEREVRRR  [240-254]
 5:  KGLEPDTDWEEREVR  [238-252]
 6:  DTDWEEREVRRRQSV  [243-257]
 7:  RRQSVELHSPQSLPR  [253-267]
 7:  RRRQSVELHSPQSLP  [252-266]
 9:  TDWEEREVRRRQSVE  [244-258]
10:  RQSVELHSPQSLPRG  [254-268]
11:  REVRRRQSVELHSPQ  [249-263]
```

Figure 10. AbDesigner output. The protein sequence is displayed first with protein information and epitope information aligned below. The bottom displays three ranked peptide lists. Those peptides at the top of the list are considered better picks to produce antibodies.

The lists produced at the bottom of the screen are unique to AbDesigner. As mentioned, AbDesigner ranks the peptides in three categories:

Ig-score. Ig-score is calculated using a custom algorithm to predict how immunogenic the peptides are. The algorithm uses the Kyte-Doolittle hydropathy index and the Chou-Fasman conformational parameters of beta turn values in order to determine the most immunogenic peptides.

Uniqueness. In order to figure out how unique a peptide is, AbDesigner does something quite interesting. It takes the peptides from the protein sequence and performs a BLAST search on them against the protein database. The BLAST search returns with similar regions found in other proteins. These regions are broken into epitope size pieces. The number of epitope pieces that have an exact match in the subject peptide returned as a result of the BLAST are counted and placed into a formula. The higher the number, the less unique the peptide.

Conservation. Conservation is similar to Uniqueness except it expresses how different a peptide is to interspecies proteins. It is calculated in a similar fashion to Uniqueness.

For a better understanding of the calculations and how the peptide are ranked, please refer to the appendix of Pisitkun, Trairak et al. The purpose of these lists is to give the user the best peptides for the production of antibodies based on the three criteria mentioned. This is not something EpiProt provides, nor will it be designed to do so. Designing antigens, as previously described, is dependent on many factors, and thus EpiProt does not try to inform the user what might be the best peptide. It is imperative that the user has a basic understanding of antigen design principles in order to use EpiProt. Depending on the user's understanding of antigen design principles, this could be a deficiency in EpiProt.

Another feature of AbDesigner that is different from EpiProt is that it is web based. When originally researching how I wanted to design EpiProt, I wanted to make it web based. With the move to cloud computing and web based programs (Jewell et al., 2013), the days of downloading large-clunky programs onto one's desktop are fading. Making EpiProt web based would certainly make EpiProt more accessible and easier to use. The reason I did not make EpiProt web-based is simple, time. I was not confident that I could develop a web based EpiProt within a year. There are many reason why, but ultimately I have never developed a web-based program as ambitious as EpiProt. At some point I hope to convert it to a web-based program.

That being said, EpiProt offers more services than AbDesigner. Some of the services EpiProt will provide that AbDesigner does not are MSA, BLAST, PDB structure and PsiPred (McGuffin, Bryson, & Jones, 2000). It will also provide several epitope prediction programs that AbDesigner does not such as Emini surface accessibility (Emini, 1985), Karplus and Schulz flexibility (Karplus & Schulz, 1985), Kolaskar and Tongaonkar antigenicity (Kolaskar & Tongaonkar, 1990), Parker Hydrophilicity (Kolaskar & Tongaonkar, 1990; Parker, Guo, & Hodges, 1986), ABCpred (Saha, Sudipto, & Raghava, 2007) and BcePred (Saha, Sudipto, & Raghava, 2004). Therefore even though EpiProt will not suggest peptide antigens, it will provide a much richer environment for users to determine the best epitope regions.

Chapter II

Software Design

This chapter gives a description of the software used to help develop EpiProt, and the third party software incorporated into it.

Development Software

1. Operating system: Mac OS X version 10.9.5
2. Processor: 2.5 GHz Intel Core i5
3. Memory: 4GB 1600z DD3
4. Java 1.8 (update 45)
5. Eclipse IDE Luna 4.4.2
6. Python 2.7
7. Maven 3.3.3

Third-Party Software

There are three third party software packages incorporated into EpiProt. Each can be used for academic purposes. For the first two, PsiPred and JPred, I give detailed instructions on how I downloaded and installed it so EpiProt can use it. I believe this was necessary to included as the download and installation was relatively complicated.

PsiPred 3.5

PsiPred needs to be downloaded and installed separately in order to work. Here is a link to the download page http://bioinfadmin.cs.ucl.ac.uk/downloads/psipred/. It can be a tricky download as it requires installing the NCBI BLAST+ toolkit found here ftp://ftp.ncbi.nih.gov/blast/executables/blast+/, and the PDB database files can be found here: ftp://ftp.ncbi.nlm.nih.gov/blast/db/. Here is a simple guide to installing PsiPred and its dependencies:

1. PDB Database. Requires ~80 GB of storage space.
    a. Open Terminal (on Mac)
    b. cd to place where you want the files downloaded
        i. cd Downloads
    c. FTP to NCBI
        i. ftp ftp.ncbi.nih.gov
    d. Login as anonymous:
        i. anonymous
        ii. <email>
    e. cd to /blast/db/
        i. cd /blast/db
    f. Turn "prompt" off
        i. prompt off
    g. Get the database files, this might take a few hours
        i. mget nr.*tar.gz cdd_delta.tar.gz pdbaa.tar.gz
    h. Move files to place you want them. Preferably /usr/local/bin. Look up mv command or move them using Finder (or Window Explorer)

- i. Untar the files, this could take several minutes
    - i. tar -zxvf nr.*tar.gz
    - ii. tar -zxvf cdd_delta.tar.gz
    - iii. tar -zxvf pdbaa.tar.gz
- j. Remove the untarred files
    - i. rm nr.*tar.gz cdd_delta.tar.gz pdbaa.tar.gz

2. Install the NCBI BLAST+ Toolkit
    - a. Open Terminal, cd to folder /usr/local
        - i. cd /usr/local
    - b. Download the NCBI Tool Kit (version 2.3.0)
        - i. ftp ftp.ncbi.nih.gov (login as described above)
        - ii. cd /blast/executables/blast+/2.3.0/
        - iii. get ncbi-blast-2.3.0+-universal-macosx.tar.gz
            1. The version to download is dependent on your operating system. Go to ftp://ftp.ncbi.nih.gov/blast/executables/blast+/2.3.0/ to find which version is right for you.
    - c. Untar the file
        - i. Tar -zxvf ncbi-blast-2.3.0+-universal-macosx.tar.gz
    - d. Remove the untarred file
        - i. rm ncbi-blast-2.3.0+-universal-macosx.tar.gz
3. Download and set up PsiPred (version 3.5)
    - a. Go to http://bioinfadmin.cs.ucl.ac.uk/downloads/psipred/

b. Download version 3.5

c. Move file to /usr/local

   i. Must be named /usr/local/psipred

d. Open go to psipred/BLAST+

e. Open runpsipredplus with a text editor (eg. TextWrangler)

f. Change the following variables to where you place the above files (if you followed the instructions above, do the following):

   i.
   ```
   # The name of the BLAST+ data bank
   set dbname = /usr/local/bin/pdb
   ```

   ii.
   ```
   # Where the NCBI BLAST+ programs have been installed
   set ncbidir = /usr/local/ncbi/blast/bin
   ```

   iii.
   ```
   # Where the PSIPRED V3 programs have been installed
   set execdir = /usr/local/psipred/bin
   ```

   iv.
   ```
   # Where the PSIPRED V3 data files have been installed
   set datadir = /usr/local/psipred/data
   ```

4. Test PsiPred

   a. Open Terminal, cd to file where runpsipredplus is

      i. `cd /usr/local/psipred/BLAST+`

   b. Run the following command

      i. `./runpsipredplus ../example/example.fasta`

   c. You should see this output

```
Running PSI-BLAST with sequence ../example/example.fasta ...
Predicting secondary structure...
Pass1 ...
Pass2 ...
Cleaning up ...
Final output files: example.ss2 example.horiz
Finished.
```

JPred 1.5

JPred is downloaded here: http://www.compbio.dundee.ac.uk/jpred/api.shtml#download. It is a Perl script that runs via command line in EpiProt. Follow these steps for successful integration:

1. Download version 1.5
2. Untar the file
3. Move to /usr/local/jpred (might need to make the directory jpred)
4. Test submission

```
perl jpredapi submit mode=single format=raw email=name@domain.com
name=my_test_job skipPDB=on
seq=MQVWPIEGIKKFETLSYLPPLTVEDLLKQIEYLLRSKWVPCLEFSKVGFVYR
ENHRSPGYYDGRYWTMWKLPMFGCTDATQVLKELEEAKKAYPDAFVRIIGFDN
VRQVQLISFIAYKPPGC
```

5. Check results (I like to do it in a browser)
    a. Open a browser
    b. Type and insert the jobid at the indicated spots:

    http://www.compbio.dundee.ac.uk/jpred4/results/<jobid>/<jobid>.results.html

    e.g.

    http://www.compbio.dundee.ac.uk/jpred4/results/jp_cPiwxcg/jp_cPiwxcg.results.html

JABAWS 2.1.0

EpiProt uses JABAWS in order to perform MSA. The JABAWS jar file can be found here: http://www.compbio.dundee.ac.uk/jabaws/. EpiProt uses command line to run the jar file and parses the results.

Chapter III

Software Architecture

EpiProt follows a Model-View-Presenter (MVP) design pattern (Potel, 2011). Figure 11 is a diagram of the general flow:

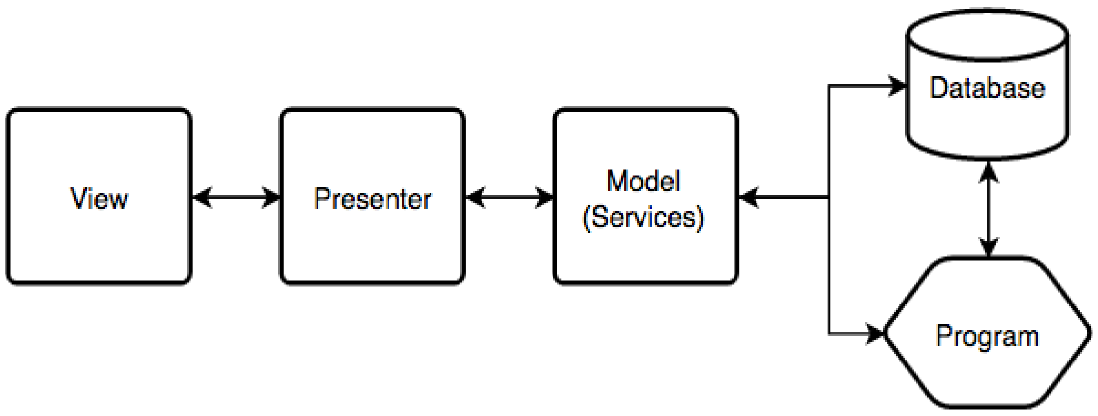


Figure 11. Standard MVP architecture. The view and model are completely separated and only joined by the presenter. The model in EpiProt consists of many "service" objects that access either a database or program to return data.

The general pattern of data flow works as follows:

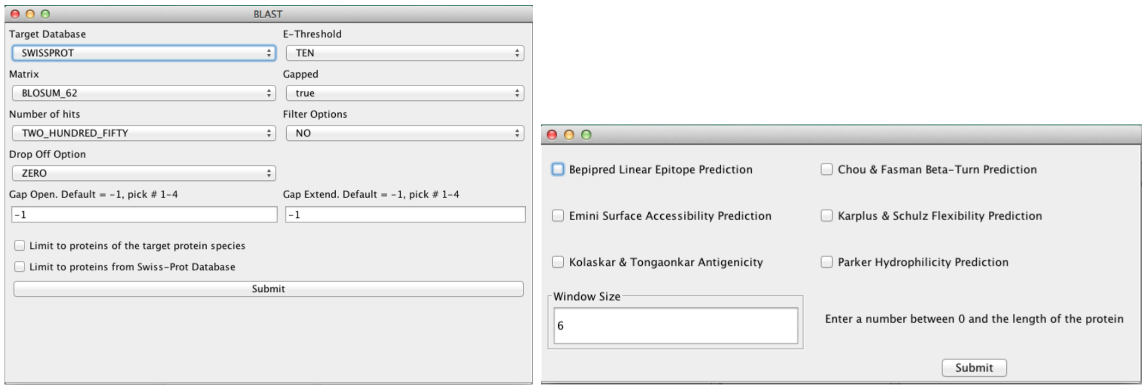
1. The user interacts with the view.
2. All actions in the view are processed in the presenter.
3. The processed script in the presenter gathers data from the model (separated into service objects).
4. The service object either:

    a. Retrieves the data from a database (e.g. UniProt, PhosphoSitePlus, PDB/SIFTS)
    b. Retrieves the data through a program (e.g. PsiPred, JPred, SSP)
5. The service collects the data and returns it to the presenter
6. Presenter processes the data according to the logic defined in the view.
7. Presenter takes the data and inserts it into the view.

In this process, the view is not aware of the model, nor is the model aware of the view. They are only connected by the presenter. The view and model each implement separate interfaces defined in the presenter. This means the presenter is only aware of the methods in the view or model that are defined in the interface that they implement in the presenter.

This pattern ensures a few things. First, if you want the model or view to return data to the presenter, you can place the method in their respective interfaces thus ensuring that the model or view implements the given method. This can be quite handy as you can "outline" the methods in the interface while building the various components and come back to implement them later. However, the main thing it ensures is compartmentalization. Making the components modular ensures each component is responsible for a certain task. This allows the developer to fix and update components relatively seamlessly.

The MVP design is seen throughout all of EpiProt. Each window has its own View and Presenter class, which can invoke one or more services. Figure 12 shows an example of a few what are termed service windows.


BLAST

Target Database: SWISSPROT
E-Threshold: TEN
Matrix: BLOSUM_62
Gapped: true
Number of hits: TWO_HUNDRED_FIFTY
Filter Options: NO
Drop Off Option: ZERO
Gap Open. Default = -1, pick # 1-4: -1
Gap Extend. Default = -1, pick # 1-4: -1

Limit to proteins of the target protein species
Limit to proteins from Swiss-Prot Database

Submit

Bepipred Linear Epitope Prediction
Chou & Fasman Beta-Turn Prediction
Emini Surface Accessibility Prediction
Karplus & Schulz Flexibility Prediction
Kolaskar & Tongaonkar Antigenicity
Parker Hydrophilicity Prediction

Window Size: 6

Enter a number between 0 and the length of the protein

Submit

Figure 12. Examples of two service windows. Left, the BLAST service window, Right, the IEDB Epitope Prediction service window. Each window has their own view and presenter class. The BLAST window invokes one service class to perform a BLAST, while the IEDB Epitope Prediction window invokes several separate service objects to retrieve the epitope prediction information.

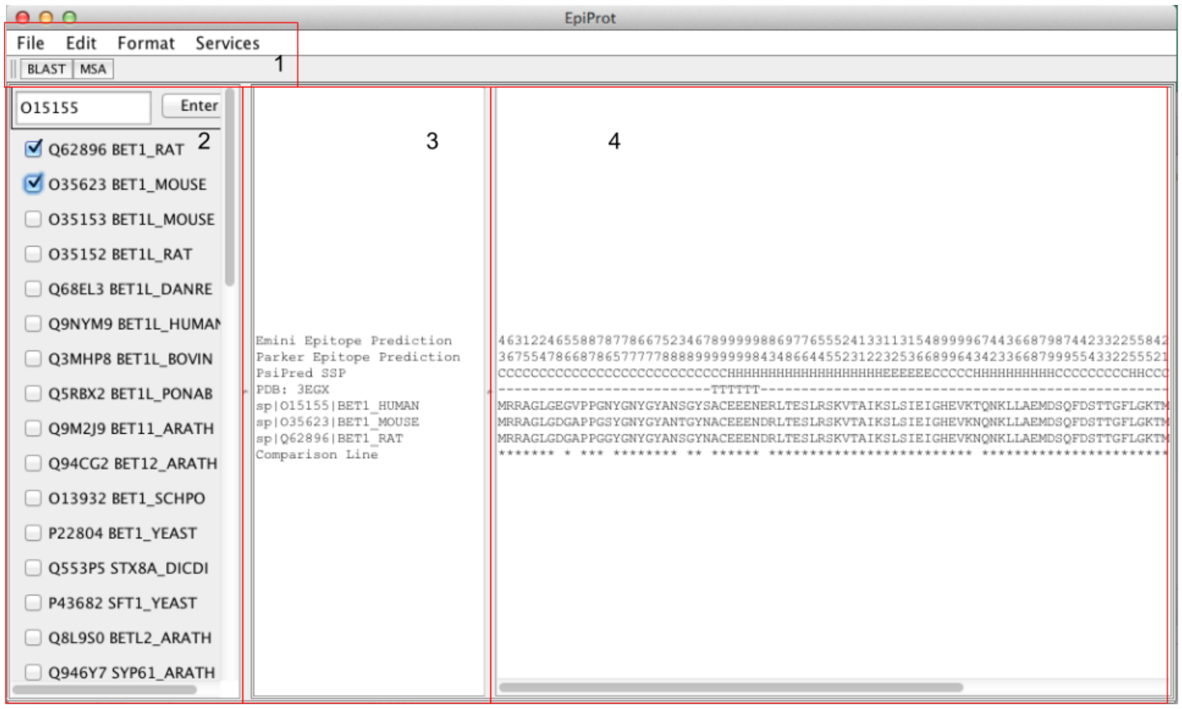
What are the tasks each view, presenter and model (or service) in EpiProt responsible for? First I will explain the overall responsibilities of the components in a MVP, highlighting the main the features of the main display. Then I will explain at a high level the dataflow of each service available in EpiProt. A more detailed explanation of how to use the EpiProt and its features will be provided in a manual upon release.

View

The view is built with Java Swing. Like any view of a MVP design, it is considered “dumb”, or it processes none of the business logic. It is not aware of the model and never calls upon any service directly. In fact, none of the actions from the service JComponents (“JComponent (Java Platform SE 7),” 2016.) are bound in the view. This makes the easier to fix, but more importantly it makes the view “lightweight” which is important if EpiProt ever evolves into a client-server model.

General Display

Looking at the layout of the display, there are four main areas:



```
Emini Epitope Prediction    4631224655887877866752346789999988697765552413311315489999674436687987442332255842
Parker Epitope Prediction   3675547866878657777788889999999843486644552312232536689964342336687999554332255521
PsiPred SSP                 CCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHHEEEEEECCCCHHHHHHHHHHCCCCCCCCCHHCCC
PDB: 3EGX                   --------------------------TTTTTT--------------------------------------------------
sp|O15155|BET1_HUMAN        MRRAGLGEGVPPGNYGNYGYANSGYSACEEENERLTESLRSKVTAIKSLSIEIGHEVKTQNKLLAEMDSQFDSTTGFLGKTM
sp|O35623|BET1_MOUSE        MRRAGLGDGAPPGSYGNYGYANTGYNACEEENDRLTESLRSKVTAIKSLSIEIGHEVKNQNKLLAEMDSQFDSTTGFLGKTM
sp|Q62896|BET1_RAT          MRRAGLGDGAPPGGYGNYGYANSGYNACEEENDRLTESLRSKVTAIKSLSIEIGHEVKNQNKLLAEMDSQFDSTTGFLGKTM
Comparison Line             ******* * *** ******** ** ****** ************************* ***********************
```

Figure 13. EpiProt main window view. 1) Menu bars. 2) BLAST panel. 3) Header pane. 4) Editor pane.

Menu Bars. There are two menu bars in EpiProt. The first three menu options in the top menu bar are typical menu options found in most applications. Most of these options have shortcut keys as well.

- The File menu allows the user to save, open and exit the application.
- The Edit menu contains operations such as cut, copy, paste, undo, redo, select all and clear.
- The Format menu allows text formatting such as bold, underline, italicize and color.

The last menu option, Services, has all the services available in EpiProt. The services are grouped together by type. For example, epitope prediction programs are grouped together, as shown in figure 14.

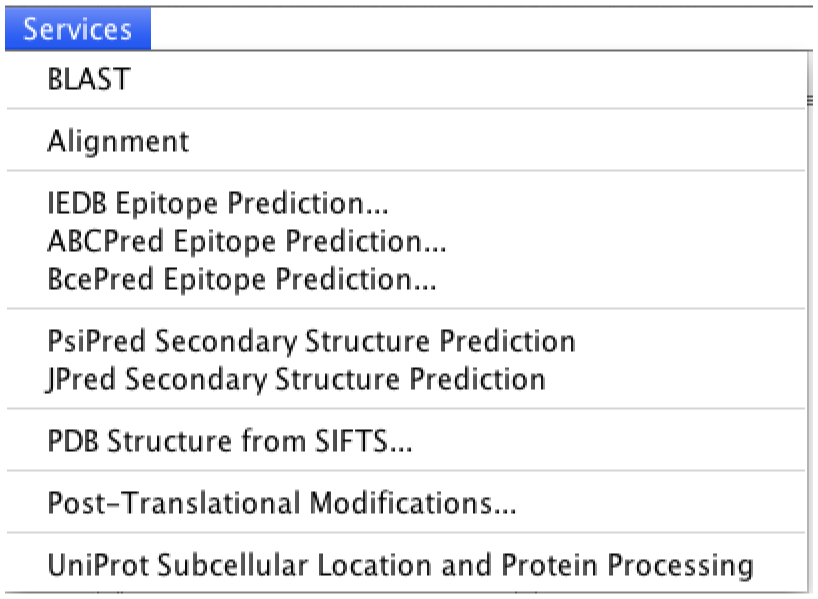

Services
BLAST
Alignment
IEDB Epitope Prediction...
ABCPred Epitope Prediction...
BcePred Epitope Prediction...
PsiPred Secondary Structure Prediction
JPred Secondary Structure Prediction
PDB Structure from SIFTS...
Post-Translational Modifications...
UniProt Subcellular Location and Protein Processing

Figure 14. The Service menu. The services available in EpiProt are grouped by type.

The second menu bar is simply a shortcut to common services such as BLAST and MSA.

BLAST panel. When running a BLAST, the panel on the left fills in with the proteins from the BLAST result. The BLAST hits are selectable with a checkbox. This allow the user to specify the protein from the BLAST search to create the MSA with.

At the top of the is panel is a text field and an “enter” button. This field is extremely important as this is where the user will place the UniProt (The UniProt Consortium, 2014) accession id of the protein being targeted for antigen design. Clicking enter inserts the protein sequence into the header and editor panes on the right.

Header Pane. The middle pane is where the header for each line is placed. This pane can be hidden or revealed by the user. Although the header for each line is important in order to recognize the line, I wanted to leave the user the ability to hide or expand the header, depending on the user’s needs.

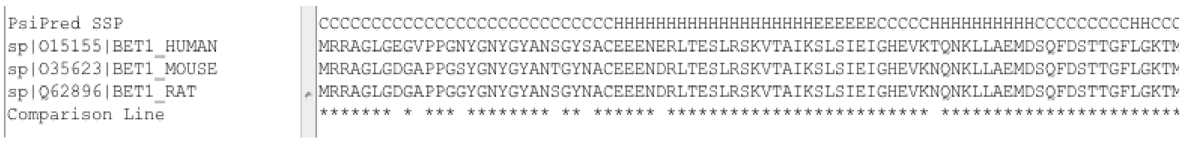
Editor Pane. The editor pane, or the main pane, is where the protein sequence, the alignments and all the other protein information is placed. The user can annotate the text in order to indicate regions of interest or to mark certain features.

Purpose

Since the View does not process any business logic, and does not handle the actions fired by the service components, its main purpose is to hold all the code to create the display, perform basic save/open operations, and format text. This can be extremely valuable to a developer because they would know any changes in the code in the view will not change the business logic of the application.

One major responsibility of the View is line insertion. The UI (user interface) of application presents two main problems:

1) EpiProt has to know where the main protein sequence line is in both the header and main displays in order to insert lines above or below it. For example, when inserting the results of PsiPred into the UI, we would want to see the results aligned above the target sequence line as shown in figure 15. EpiProt will take the results, find the main target sequence line and insert the PsiPred line above it.



```
PsiPred SSP              CCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHEEEEEECCCCHHHHHHHHHHCCCCCCCCCHHCCC
sp|O15155|BET1_HUMAN     MRRAGLGEGVPPGNYGNYGYANSGYSACEEENERLTESLRSKVTAIKSLSIEIGHEVKTQNKLLAEMDSQFDSTTGFLGKT
sp|O35623|BET1_MOUSE     MRRAGLGDGAPPGSYGNYGYANTGYNACEEENDRLTESLRSKVTAIKSLSIEIGHEVKNQNKLLAEMDSQFDSTTGFLGKT
sp|Q62896|BET1_RAT       MRRAGLGDGAPPGGYGNYGYANSGYNACEEENDRLTESLRSKVTAIKSLSIEIGHEVKNQNKLLAEMDSQFDSTTGFLGKT
Comparison Line          ******* * *** ******** ** ****** ************************** *********************
```

Figure 15. Demonstration of line insertion in EpiProt. The Presenter is designed to insert lines based on type. For example the PsiPred SSP line is always inserted above the main target sequence line (BET1_HUMAN) while other sequences from the MSA and the comparison line are always inserted below.

The trick to ensuring that a line is always inserted properly is HTML ("HTML Tutorial," 2016.). Both the header pane and editor pane are instances of the type

JEditorPane (“JEditorPane (Java Platform SE 7),” 2016.). This Swing JTextComponent (“JTextComponent (Java Platform SE 7),” 2016.) allows one to insert HTML documents into the UI. Although it could seem strange to use HTML in an application that is not web-based, its structure allows one to find lines easily making it the perfect document type for EpiProt. Figure 16 shows an example of what the HTML file migh look like.

```
<html>
 <head>
 </head>
 <body>
  <br><br><br><br><br><br><br><br><br><br><br><br><br><br><br>

  <pre id="PsiPred SSP" name="PsiPred SSP" style="line-height:
0px">CCCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHHHEEEEEEECCCCCHHHHHHHHHHCCC
CCCCCCHHCCCCCEEEECCCCHHHHHHHHHHHHHHHHHHHHHHHHHCC</pre>

  <pre id="O15155" name="sp|O15155|BET1_HUMAN" style="line-height:
0px">MRRAGLGEGVPPGNYGNYGYANSGYSACEEENERLTESLRSKVTAIKSLSIEIGHEVKTQNKLLAEMDSQFDST
TGFLGKTMGKLKILSRGSQTKLLCYMMLFSLFVFFIIYWIIKLR</pre>

  <pre id="sp|O35623|BET1_MOUSE" name="sp|O35623|BET1_MOUSE" style="line-height:
0px">MRRAGLGDGAPPGSYGNYGYANTGYNACEEENDRLTESLRSKVTAIKSLSIEIGHEVKNQNKLLAEMDSQFDS
TTGFLGKTMGRLKILSRGSQTKLLCYMMLFSLFVFFVIYWIIKLR</pre>

  <pre id="sp|Q62896|BET1_RAT" name="sp|Q62896|BET1_RAT" style="line-height:
0px">MRRAGLGDGAPPGGYGNYGYANSGYNACEEENDRLTESLRSKVTAIKSLSIEIGHEVKNQNKLLAEMDSQFDS
TTGFLGKTMGRLKILSRGSQTKLLCYMMLFSLFVFFVIYWIIKLR</pre>
```

Figure 16. The HTML code for the display in Figure 15. The header in the header pane is the id in for the line in the editor pane.

The attributes for each line make it easy to search for the target protein sequence line in the document. Using Jsoup (Hedley, 2016.) EpiProt is able to search for elements by attribute value, then it is just a matter of getting the element’s start and end position within the document in order to insert the new line.

2) EpiProt has to insert the header line in the header pane at the same position as its corresponding line in the editor pane. Since the header pane holds an HTML document as well, EpiProt addresses this problem by inserting the header line as an element with the same id as the corresponding line in the editor pane (figure 17). Then it

is simply a matter of finding the target protein header line and inserting the new line either above or below.

```
<pre style="LINE-HEIGHT:0px;" id="O15155" name="O15155|BET1_HUMAN|Swiss-
Prot">O15155|BET1_HUMAN|Swiss-Prot</pre>

<pre style="LINE-HEIGHT:0px;" id="O15155" name="O15155|BET1_HUMAN|Swiss-
Prot">MRRAGLGEGVPPGNYGNYGYANSGYSACEEENERLTESLRSKVTAIKSLSIEIGHEVKTQNKLLAEMDSQFDS
TTGFLGKTMGKLKILSRGSQTKLLCYMMLFSLFVFFIIYWIIKLR</pre>
```

Figure 17. Comparing the header and main lines for the target protein. Notice the attributes are exactly the same between the two elements. This ensures the two elements are connected when changes to the document are performed.

Since HTML is used, this makes it easy to format text and subsequently save it. All text formatting and document saving/opening/clearing operations are processed within the view itself.

Presenter

The Presenter is central to the flow of data within EpiProt making it responsible for a number of processes. As mentioned, the Presenter is the component that connects the View to the Model. It takes the parameters from the view to retrieve data from the model using data access objects (DAOs) ("Core J2EE Patterns - Data Access Object," 2016.). It can process those DAOs to show the needed information on the view. It is also the place where all the action listeners for the various service JComponents in the View are bound. Therefore the action of any click or keystroke initiated for a service is processed in the presenter, not the view.

Model

The model starts with what EpiProt calls "services". Each service class retrieves one piece of information and brings that back to either the Model or a Presenter class. A service object extends *Service*, which is itself, an extension of TimerTask ("TimerTask (Java Platform SE 7)," 2016.). Each service operates by first calling the run() method from TimerTask, and then getting the data once run() is complete.

Almost all the data comes back as information tied to an amino acid. For example, when running PsiPred the protein sequence is inputted into PsiPred through a file called PsiPredService.java which produces an output file, when run() is called, with the format in figure 18:

```
# PSIPRED HFORMAT (PSIPRED V3.5)

Conf: 976788999999977665434566530134367878988754212200155624401204
Pred: CCCCCCCCCCCCCCCCCCCCCCCCCCCCHHHHHHHHHHHHHHHHHHHEEEEEECCCCCHH
  AA: MRRAGLGEGVPPGNYGNYGYANSGYSACEEENERLTESLRSKVTAIKSLSIEIGHEVKTQ
               10        20        30        40        50        60

Conf: 4688875403434444101111310230257514489999999999999999998709
Pred: HHHHHHHHCCCCCCCCCHHCCCCCEEEECCCCHHHHHHHHHHHHHHHHHHHHHHHHCC
  AA: NKLLAEMDSQFDSTTGFLGKTMGKLKILSRGSQTKLLCYMMLFSLFVFFIIYWIIKLR
               70        80        90       100       110
```

Figure 18. PsiPred output of human BET1, UniProt O15155.

Notice that each amino acid gets assigned a secondary structure (H, E, C), and a confidence number (0= lowest, 9=highest). For each amino acid a PsiPredAminoAcid.java object is created where this information is recorded. The method getAminoAcids() returns an ArrayList ("ArrayList (Java Platform SE 7)," 2016.) of PsiPredAminoAcid.java the length of the protein sequence. Almost every service follows this pattern of run/get amino acids. This makes transferring the information easier.

Depending on the information it is either gathered from a website such as UniProt, PDB/SIFTS (Velankar et al., 2013) or PhosphoSitePlus, or it is calculated such as BLAST, MSA, SSP and epitope prediction. No matter what, each point of data gets initiated through a service object. If it is database information, such as UniProt, it simply connects to UniProt and returns a org.w3c.dom.Document ("Document (Java Platform

SE 7),” 2016.) of the protein in UniProt for parsing. If it is something more complicated like PsiPred, it will run the program, write the results to a temporary file, parse the file and return the results.

Data Access Objects

Central to any MVP are DAOs. These are objects that allow a program to transfer data between various layers, such as a model and presenter. In EpiProt the most common type of DAO is of the type AminoAcid.java. As mentioned previously, most data in EpiProt is carried at the amino acid level. Almost every service creates an extended AminoAcid.java object with special fields in it to carry the information back to the presenter.

For example, Figure 19 shows the results of running human BET1 on PsiPred. A service object named PsiPredService.java initiates PsiPred to generate this file. It then parses the file creating a PsiPredAminoAcid.java for each amino acid in the protein and places the amino acid object in an ArrayList. PsiPredAminoAcid.java holds four pieces of information: 1) amino acid residue, 2) amino acid position, 3) predicted secondary structure and 4) confidence score. Amino acid residue and position information is inherited from AminoAcid.java while predicted secondary structure and confidence score are special attributes of PsiPredAminoAcid.java. Ultimately only the predicted secondary structure information is shown in the display, making it the only needed information. This architecture however allows one to easily display more information without changing much code.

Not MVP: Protein.java

Perhaps one of the most pivotal and different classes in EpiProt is Protein.java. Essentially it represents all the protein information one could gather from UniProt. Using UniProtService.java, it retrieves the UniProt XML file of the protein specified and parses it for information. This makes it extremely important as it is the place where the protein sequence, database, organism, UniProt id and PDB id information are derived from. The presenter in fact has a global variable called "protein" that represents the target protein. This makes it easier to access information about the target protein in any method without having to go through a separate model classes to retrieve information about the protein.

It is different in that it does not fit the general mold of a MVP. It is part model, part DAO, making it one of the more unique classes in EpiProt. It does gather and parse the information from the UniProt page, but it also transfers that information to the Presenter in a logical container to display information and help run other programs. This demonstrates that while EpiProt follows an MVP design, the needs of the program outweighs the architecture plan when appropriate.

Although Protein.java is essentially the interface between UniProt and EpiProt, only certain information is pulled from UniProt page for EpiProt. This includes protein sequence, organism, database, subcellular location, protein processing, post-translational modifications and PDB structure entries. Some of this information is used as an input for other services (e.g. PDB ids), some can be chosen by the user to be displayed (e.g. subcellular locations and protein processing) and some is used as input and displayed (e.g. protein sequence).

Subcellular location and protein processing information are displayed similarly above the target protein sequence line using colored block characters. The color of the block determines the type of subcellular location or protein processing that occurs at that region of the protein. A full description of the colors and what they represent will be provided in a manual when EpiProt is released.

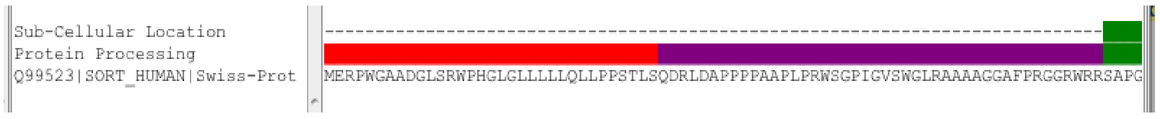

Sub-Cellular Location
Protein Processing
Q99523|SORT_HUMAN|Swiss-Prot
MERPWGAADGLSRWPHGLGLLLLLQLLPPSTLSQDRLDAPPPPAAPLPRWSGPIGVSWGLRAAAAGGAFPRGGRWRRSAPG

Figure 19. Image of EpiProt displaying subcellular and protein processing information at the N-terminus of human Sortilin Q99523. The green in the subcellular location line represents extracellular regions. For the protein processing line red represents signal peptides, purple propeptides, green the main chain.

PTMs information from UniProt is another piece of annotation that can be displayed. Although Protein.java collects it, this information is ultimately displayed to the view through a presenter named PhosphoSitePresenter.java. UniProt does capture some of the same PTMs as PhosphoSitePlus including phosphorylations, acetylations, methylations ubiquitination and glycosylation. Others that UniProt captures but PhosphoSitePlus does not include amidation, flavination, hydroxylation, isomerization, pyrrolidone and sulfation.

The Services

Last under Software Architecture is Services. Previous sections of this thesis have discussed what information is needed for antigen design, and the way that information is ultimately transferred through various software layers to the view, but no real sense of how and where EpiProt gathers this information. Here I will briefly describe each service used in EpiProt.

UniprotService.java. Used exclusively by Protein.java, this service retrieves two files based on the UniProt accession id passed into it: 1) the UniProt XML representation of the protein and 2) the FASTA file of the protein. It does this by connecting to UniProt with a URL.

UniprotBlastService.java. This service is used whenever performing a BLAST search. Using UniProtJAPI (Patient et al., 2008, pp. 1321–1322)) it uses the parameters from the user to perform a BLAST search. Once the BLAST is complete it returns the results as a list of UniProtHit ("UniProtHit (UniProt JAPI 1.0.6 API)," 2016.). Each UniProtHit is represented as a checkbox in the BLAST panel (#2 in Figure 13).

Its dependency on UniProtJAPI makes it unstable as I had to change a considerable amount of code halfway through the development process because of upgrades to UniProtJAPI. Since UniProtJAPI is a wrapper around NCBI BLAST, future versions of EpiProt might replace this BLAST service with a much more reliable version.

PsiPredService.java. One of two SSP services, this class takes in a FASTA file of the protein and inputs it into a command line that runs PsiPred. PsiPred produces a file, which this service parses and returns the results as an ArrayList of PsiPredAminoAcid.java.

In order for this service to work, PsiPred has to be properly installed on your computer in the "usr/local" directory with a directory name of "psipred".

JPredService.java. Similar to PsiPredService.java, this service takes in a protein sequence and through command line collects SSP information from JPredAPI ("JPred: A Protein Secondary Structure Prediction Server," 2016.). The results are returned as an ArrayList of JPredAminoAcid.java.

To work properly, JPredAPI must be properly installed on your computer in the "usr/local" directory with the directory name of "jpred".

SiftService.java. Using the PDB id as input, SiftService.java collects information from the SIFTS (Velankar et al., 2013) database. The SIFTS database is a collection of XML files of PDB structures correctly indexed to the UniProt sequence. Retrieving a SIFTS XML file of a particular PDB structure requires one to insert the PDB id into the URL. The PDB ids are retrieved from the UniProt protein page using Protein.java. After the XML file is retrieved, SiftService.java parses the file and returns a list of SiftAminoAcid.java objects the length of the structure with all the available structure information.

PhosphoSitePlus.java. As this name implies, this service is responsible for collecting PTM information from PhosphoSitePlus.org. It does this by downloading the various PTM dataset files available at http://www.phosphosite.org/downloads/ and parses them looking for PTM sites by the UniProt accession id. The user has the choice of selecting 7 PTM types to analyze. This includes phosphorylations, acetylations, methylations, O-linked glycosylations (O-GalNAc and O-GlcNAc), sumoylations and ubiquitinations. Each PTM type is inserted as a line above the target protein sequence, annotating which amino acids have a PTM. If the PTM does not occur in the protein, and the user chooses it, the line for that type will simply not show.

IedbEpitopePredictionService.java. To retrieve any epitope prediction information from IEDB, IEDB provides a simple "curl" command to get results ("Tools-API," 2016.). Here is an example of a command that retrieves Emini Surface Accessibility with a window size of 9:

```
curl --data
"method=Emini&sequence_text=VLSEGEWQLVLHVWAKVEADVAGHGQDILIRLF
KSHPETLEKFDRFKHLKTE&window_size=9" http://tools-api.iedb.org/tools_api/bcell/
```

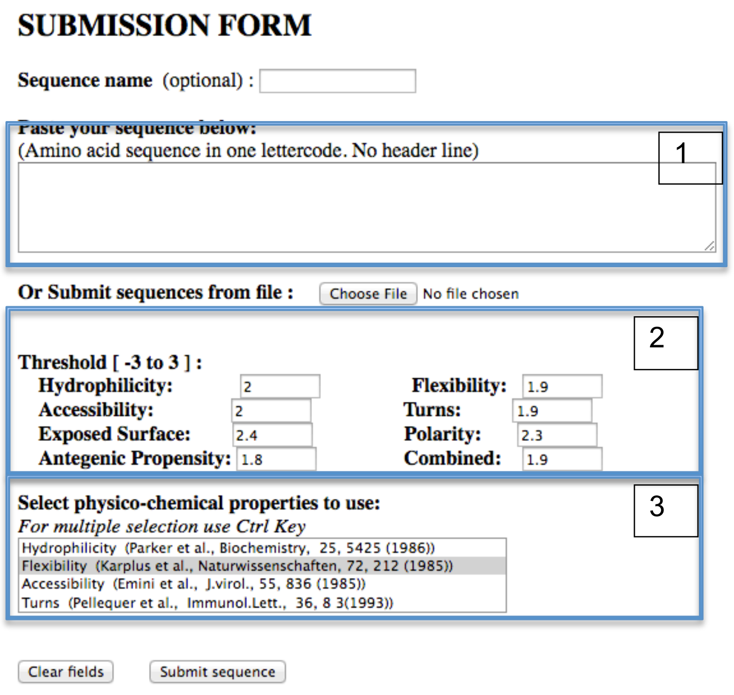
There are three parameters in this command: 1) the epitope prediction program, 2) the sequence, and 3) the window size. IedbEpitopePredictionService.java runs this command, but replaces the parameter values with those defined by the user.

The results return as a tab-delimited chart with scores for almost every amino acid in the sequence. Depending on the window size, certain amino acids at the ends of the sequence will not receive scores. The file is parsed, each amino acid is given a score, and the scores are ranked. The program will then calculate a single digit score for each amino acid based off the score from the results. Those in the lower 10% will receive a 0, between 10% and 20% receive a 1, those between 20% and 30% a 2, etc. The score is set in each amino acid and the amino acids are returned to the presenter as a ArrayList of IedbEpitopePredictionAminoAcid.java. The presenter presents the values to the view as a long string of numbers, which is inserted above the sequence line of the target protein.

PredictionService.java. BcePred is a website provided by G. P. S. Raghava and Sudipto Saha at the Institute of Microbial Technology located in Chandigarh, India (Saha et al., 2004). It provides several epitope prediction tools for the user, including some that are available at IEDB (for a summary please refer to table 3). The epitope prediction tools are named based on the property they use to predict epitopes.

If there are two or more programs chosen, it also provides the minimum and maximum values for each residue among the programs chosen. It will also average the values for each residue. The user can also enter a threshold number between -3 and 3 for each program. If the score for a residue is above the threshold, the number will be colored blue in the web display.

In order to access BcePred, BcePredPredictionService.java has to submit the request through the form, figure 20, on its website since there is no API (Application Program Interface) to submit the request. To do this BcePredPredictionService.java uses a dependency called HtmlUnit (Surhone, Tennoe, & Henssonow, 2010) to parse HTML pages, input data, submit data and retrieve results.



SUBMISSION FORM

Sequence name (optional) :

Paste your sequence below:
(Amino acid sequence in one lettercode. No header line)

Or Submit sequences from file : Choose File No file chosen

Threshold [-3 to 3] :

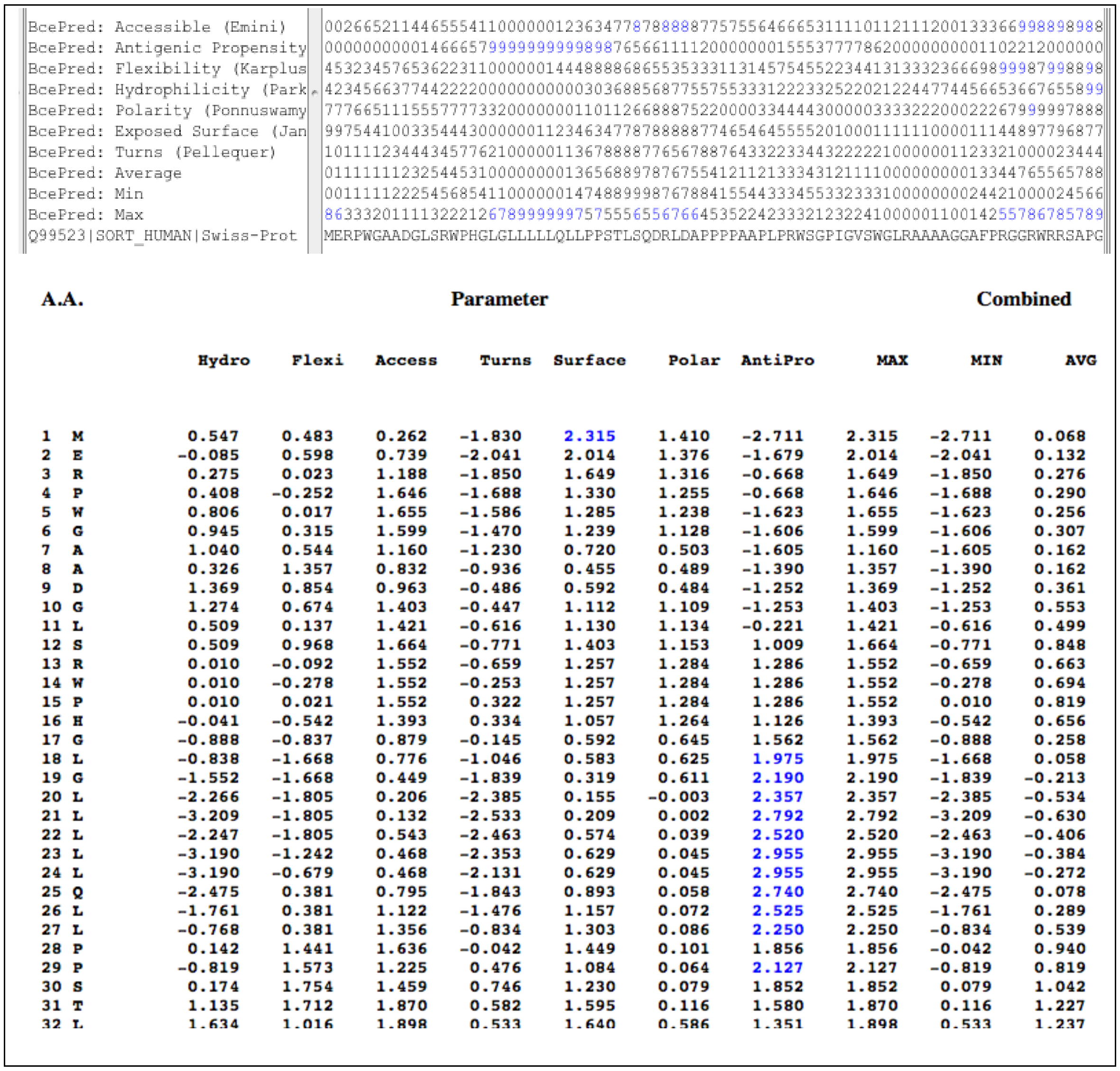
Hydrophilicity:	2	**Flexibility:**	1.9
Accessibility:	2	**Turns:**	1.9
Exposed Surface:	2.4	**Polarity:**	2.3
Antegenic Propensity:	1.8	**Combined:**	1.9

Select physico-chemical properties to use:
For multiple selection use Ctrl Key

Hydrophilicity (Parker et al., Biochemistry, 25, 5425 (1986))
Flexibility (Karplus et al., Naturwissenschaften, 72, 212 (1985))
Accessibility (Emini et al., J.virol., 55, 836 (1985))
Turns (Pellequer et al., Immunol.Lett., 36, 8 3(1993))

Figure 20. BcePred submission form. There are 3 main inputs: 1) the sequence, 2) the thresholds for each program, 3) the programs.

Once the data is received, each amino acid is given a value for each program ran, plus any min, max and average number that residue might have. Just like with IedbEpitopePredictionService.java, the scores are ranked and another single digit score is assigned to each amino acid for each program ran. These scores are displayed in the view above the target protein line as seen in figure 21. If the score from results is above the threshold in the input, the displayed score in the view will be colored blue.



```
BcePred: Accessible (Emini)     0026652114465554110000001236347787888887757556466653111101121112001333669988989 88
BcePred: Antigenic Propensity   00000000001466657999999999998987656611112000000015553777786200000000011022120000 00
BcePred: Flexibility (Karplus   45323457653622311000000144488886865535333113145754552234413133323666989998799889 8
BcePred: Hydrophilicity (Park   42345663774422220000000000030368856877557553331222332522021224477445665366765589 9
BcePred: Polarity (Ponnuswamy   77766511155577773320000000110112668887522000033444430000033332220002226799997888
BcePred: Exposed Surface (Jan   99754410033544430000001123463477878888877465464555520100011111100001114489779687 7
BcePred: Turns (Pellequer)      10111123444345776210000011367888877656788764332233443222221000000112332100002344 4
BcePred: Average                01111111232544531000000001365688978767554121121333431211110000000000133447656578 8
BcePred: Min                    00111111222545685411000000147488999876788415544333455332333100000000244210000245 66
BcePred: Max                    86333201111322212678999999757555655676645352242333212322410000011001425578678578 9
Q99523|SORT_HUMAN|Swiss-Prot    MERPWGAADGLSRWPHGLGLLLLLQLLPPSTLSQDRLDAPPPPAAPLPRWSGPIGVSWGLRAAAAGGAFPRGGRWRRSAPG
```

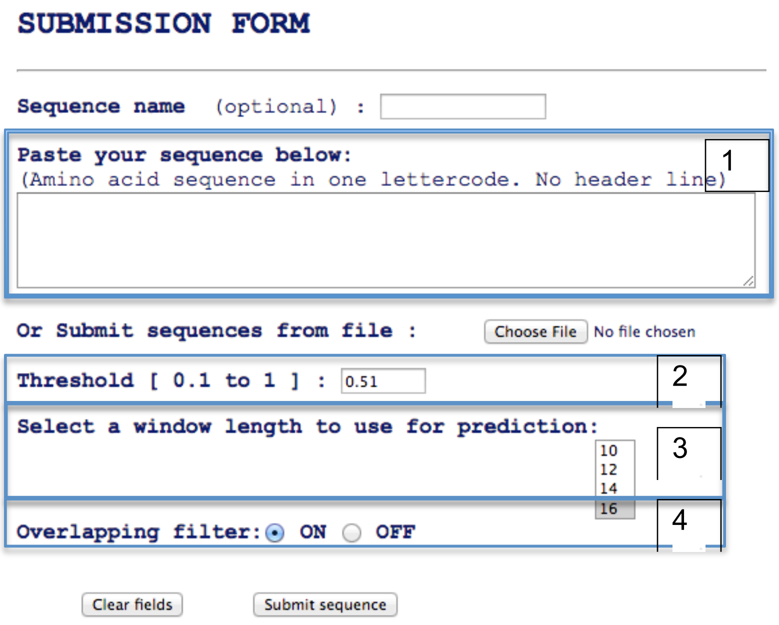
A.A.		Parameter							Combined		
		Hydro	Flexi	Access	Turns	Surface	Polar	AntiPro	MAX	MIN	AVG
1	M	0.547	0.483	0.262	-1.830	2.315	1.410	-2.711	2.315	-2.711	0.068
2	E	-0.085	0.598	0.739	-2.041	2.014	1.376	-1.679	2.014	-2.041	0.132
3	R	0.275	0.023	1.188	-1.850	1.649	1.316	-0.668	1.649	-1.850	0.276
4	P	0.408	-0.252	1.646	-1.688	1.330	1.255	-0.668	1.646	-1.688	0.290
5	W	0.806	0.017	1.655	-1.586	1.285	1.238	-1.623	1.655	-1.623	0.256
6	G	0.945	0.315	1.599	-1.470	1.239	1.128	-1.606	1.599	-1.606	0.307
7	A	1.040	0.544	1.160	-1.230	0.720	0.503	-1.605	1.160	-1.605	0.162
8	A	0.326	1.357	0.832	-0.936	0.455	0.489	-1.390	1.357	-1.390	0.162
9	D	1.369	0.854	0.963	-0.486	0.592	0.484	-1.252	1.369	-1.252	0.361
10	G	1.274	0.674	1.403	-0.447	1.112	1.109	-1.253	1.403	-1.253	0.553
11	L	0.509	0.137	1.421	-0.616	1.130	1.134	-0.221	1.421	-0.616	0.499
12	S	0.509	0.968	1.664	-0.771	1.403	1.153	1.009	1.664	-0.771	0.848
13	R	0.010	-0.092	1.552	-0.659	1.257	1.284	1.286	1.552	-0.659	0.663
14	W	0.010	-0.278	1.552	-0.253	1.257	1.284	1.286	1.552	-0.278	0.694
15	P	0.010	0.021	1.552	0.322	1.257	1.284	1.286	1.552	0.010	0.819
16	H	-0.041	-0.542	1.393	0.334	1.057	1.264	1.126	1.393	-0.542	0.656
17	G	-0.888	-0.837	0.879	-0.145	0.592	0.645	1.562	1.562	-0.888	0.258
18	L	-0.838	-1.668	0.776	-1.046	0.583	0.625	1.975	1.975	-1.668	0.058
19	G	-1.552	-1.668	0.449	-1.839	0.319	0.611	2.190	2.190	-1.839	-0.213
20	L	-2.266	-1.805	0.206	-2.385	0.155	-0.003	2.357	2.357	-2.385	-0.534
21	L	-3.209	-1.805	0.132	-2.533	0.209	0.002	2.792	2.792	-3.209	-0.630
22	L	-2.247	-1.805	0.543	-2.463	0.574	0.039	2.520	2.520	-2.463	-0.406
23	L	-3.190	-1.242	0.468	-2.353	0.629	0.045	2.955	2.955	-3.190	-0.384
24	L	-3.190	-0.679	0.468	-2.131	0.629	0.045	2.955	2.955	-3.190	-0.272
25	Q	-2.475	0.381	0.795	-1.843	0.893	0.058	2.740	2.740	-2.475	0.078
26	L	-1.761	0.381	1.122	-1.476	1.157	0.072	2.525	2.525	-1.761	0.289
27	L	-0.768	0.381	1.356	-0.834	1.303	0.086	2.250	2.250	-0.834	0.539
28	P	0.142	1.441	1.636	-0.042	1.449	0.101	1.856	1.856	-0.042	0.940
29	P	-0.819	1.573	1.225	0.476	1.084	0.064	2.127	2.127	-0.819	0.819
30	S	0.174	1.754	1.459	0.746	1.230	0.079	1.852	1.852	0.079	1.042
31	T	1.135	1.712	1.870	0.582	1.595	0.116	1.580	1.870	0.116	1.227
32	L	1.634	1.016	1.898	0.533	1.640	0.586	1.351	1.898	0.533	1.237

Figure 21. (Top) EpiProt display. (Bottom) BcePred display. If the score is above the threshold, the score is colored blue.

ABCPredService.java. ABCPred is another epitope prediction service by G. P. S. Raghava and Sudipto Saha that uses neural networks trained on a clean, non-redundant dataset from the BciPep database to predict epitopes (Saha, Sudipto, & Raghava, 2006).

Once again it requires one to fill out an online form, figure 22, therefore EpiProt has to use HtmlUnit to submit and retrieve data.

There are 4 required fields to enter for submission: 1) the sequence, 2) threshold, 3) window size and 4) a boolean for whether the windows should overlap:


SUBMISSION FORM

Sequence name (optional) :

Paste your sequence below:
(Amino acid sequence in one lettercode. No header line)

Or Submit sequences from file : No file chosen

Threshold [0.1 to 1] : 0.51

Select a window length to use for prediction: 10 12 14 [✓ 16]

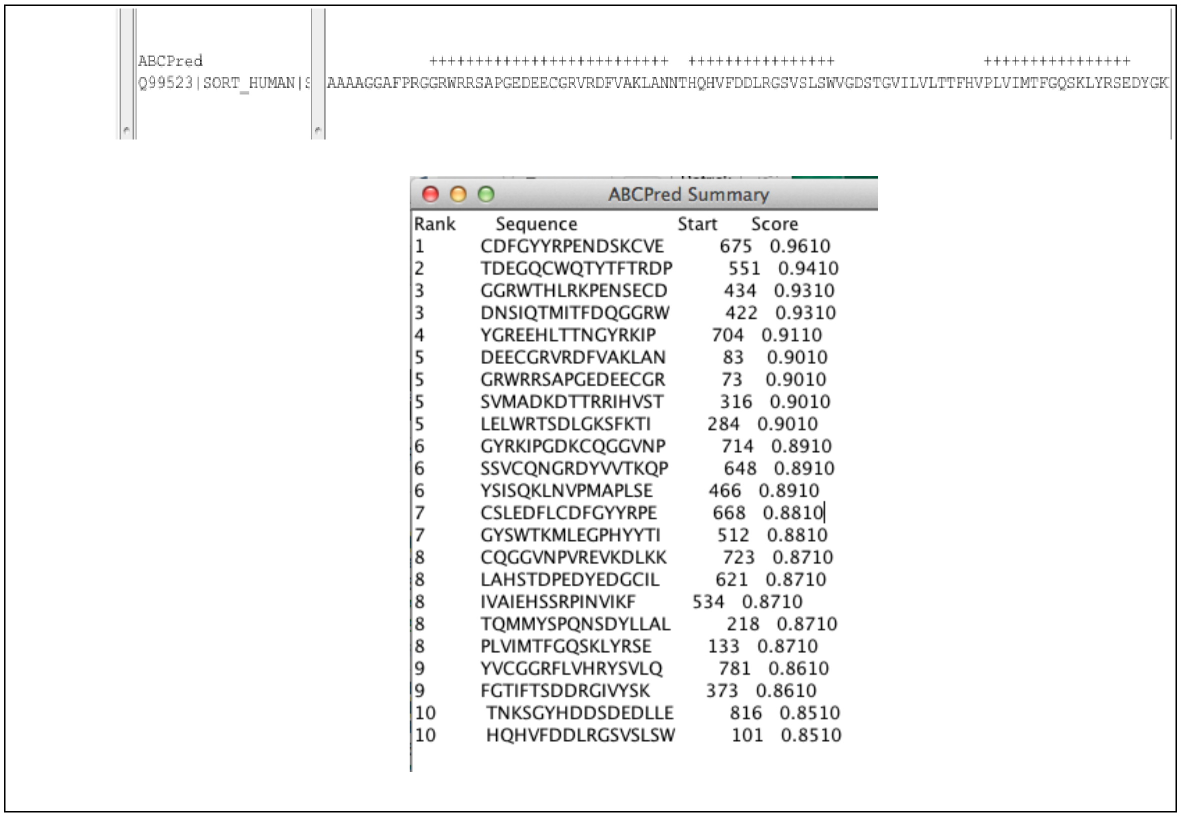
Overlapping filter: [✓ ON] OFF

Figure 22. ABCPred submission screen.

ABCPred returns results slightly different from most of the other epitope prediction programs. It produces a large table of peptides, each the length of the window size, each with a score. Only those peptides with a score higher than the threshold number are shown. The peptides are ranked based on their scores.

Unlike many of the of the other services, the fundamental DAO for ABCPredService.java represents a peptide, not an amino acid. An ArrayList of ABCPredPeptide.java (the DAO for the peptides) is transferred to the presenter. The presenter inserts a line in the view with the peptides from the results marked with "+"

characters. Also, a separate window is populated with the peptides from the submission. Figure 23 illustrates what this line look like in EpiProt.



```
ABCPred                        ++++++++++++++++++++++++++  ++++++++++++++++              ++++++++++++++++
Q99523|SORT_HUMAN|S  AAAAGGAFPRGGRWRRSAPGEDEECGRVRDFVAKLANNTHQHVFDDLRGSVSLSWVGDSTGVILVLTTFHVPLVIMTFGQSKLYRSEDYGK
```

ABCPred Summary

Rank	Sequence	Start	Score
1	CDFGYYRPENDSKCVE	675	0.9610
2	TDEGQCWQTYTFTRDP	551	0.9410
3	GGRWTHLRKPENSECD	434	0.9310
3	DNSIQTMITFDQGGRW	422	0.9310
4	YGREEHLTTNGYRKIP	704	0.9110
5	DEECGRVRDFVAKLAN	83	0.9010
5	GRWRRSAPGEDEECGR	73	0.9010
5	SVMADKDTTRRIHVST	316	0.9010
5	LELWRTSDLGKSFKTI	284	0.9010
6	GYRKIPGDKCQGGVNP	714	0.8910
6	SSVCQNGRDYVVTKQP	648	0.8910
6	YSISQKLNVPMAPLSE	466	0.8910
7	CSLEDFLCDFGYYRPE	668	0.8810
7	GYSWTKMLEGPHYYTI	512	0.8810
8	CQGGVNPVREVKDLKK	723	0.8710
8	LAHSTDPEDYEDGCIL	621	0.8710
8	IVAIEHSSRPINVIKF	534	0.8710
8	TQMMYSPQNSDYLLAL	218	0.8710
8	PLVIMTFGQSKLYRSE	133	0.8710
9	YVCGGRFLVHRYSVLQ	781	0.8610
9	FGTIFTSDDRGIVYSK	373	0.8610
10	TNKSGYHDDSDEDLLE	816	0.8510
10	HQHVFDDLRGSVSLSW	101	0.8510

Figure 23. (Top) Peptides are annotated above the target protein sequence with "+" signs. (Bottom) Peptides are ranked in a display.

JabawsService.java. JabawsService.java is a wrapper for Jabaws.jar ("JDK - javaws.jar - JDK 1.6.0_06 Java Web Start," 2016.) which is an executable jar file for many well-known bioinformatics programs. EpiProt only uses Jabaws.jar to perform MSAs. In doing so, JabawsService.java takes two parameters: 1) the type of MSA (ClustalO, MUSCLE, T-coffee, etc) and 2) an ArrayList of the accession ids of the proteins in the alignment. It first creates a file with the FASTA of every protein in the ArrayList. It then executes Jabaws.jar with two parameters: 1) the type of MSA to perform and 2) the path to the file containing all the FASTAs.

Once the MSA is complete, JabawsService.java parses the results and returns the alignment as a LinkedHashMap (“LinkedHashMap (Java Platform SE 7),” 2016.). Each element in the LinkedHashMap represents one of the proteins aligned with the protein accession number as the key and the protein alignment as the value.

Figure 24 is a summary of all the components and programs EpiProt accesses.

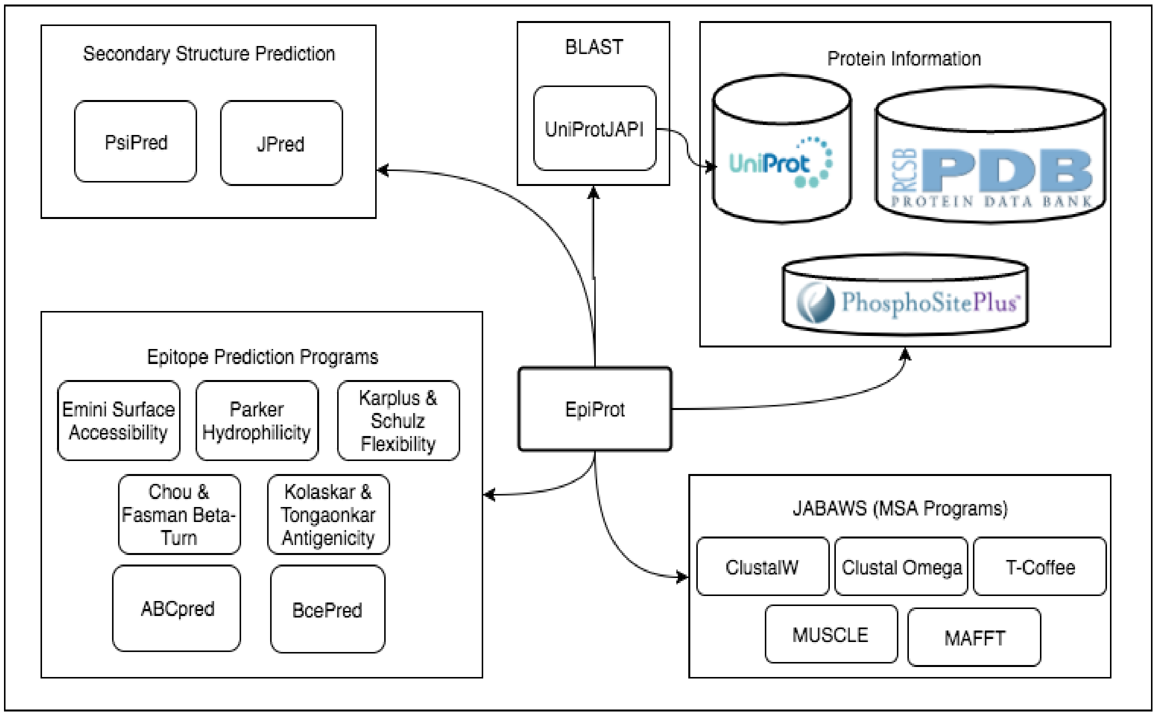


Figure 24. High-level view of EpiProt and what information it retrieves.

Chapter IV

Summary and Conclusion

Anyone familiar with the Agile framework (Beck et al, 2001) used in software development would know that there should always be an infinite backlog of work in queue. This is true whether you are developing large-scale enterprise programs or, in this case, small programs to help researchers find epitope regions in proteins. Agile methodology also recognizes that time cannot be taken back, and emphasizes that retrospective analysis of development cycles are necessary in order to optimize further development. Here I do the same, reflecting on what was proposed vs. delivered, and what the backlog could look like in the future.

Proposed vs. Actual Program

In the course of creating EpiProt there were several proposed elements that were changed or abandoned during the development process. Some elements did not make sense in the end to construct. For example, creating the *Line* and *Position* classes to help with line insert into the view did not really make much sense as the structure of HTML took care of the nuances of line insertion. Since *Position* did not exist, *AminoAcid* never extended it. In the end these changes made sense from an architectural standpoint, and helped EpiProt become better program.

Other changes were not quite as desirable. As mentioned, the command line for Bepipred never ended up working, so I had to remove it as an EPP. I also wanted to provide a fully functional text formating suite that included highlighting text. This proved surprisingly challenging, as I could not find a good way to save the highlighted text. Since time management was a major consideration during the development of EpiProt, and highlighting text was not critical, I abandoned highlighting altogether.

I also abandoned the concept of breaking up the protein sequence into small peptides and performing a BLAST search on those small peptides in order to determine which regions where specific to the target protein. The BLAST search would have taken forever as thousands of peptides could have been generated. My thesis director suggested that I create a peptide BLAST feature where the user could highlight a region to BLAST, which would return with results that the user could then align to the main protein sequence. Unfortunately, due to time, I could not build this piece of functionality. It is something I do intend to integrate in the near future.

As mentioned, the epitope predictions programs Ellipro and Discotope cannot be integrated into EpiProt before completion of this thesis. This is quite a disappointment as these epitope prediction programs use protein 3D structure to find epitope regions, thereby including potential discontinuous epitopes to be displayed. Pieces of the functionality are already built, including the services that retrieve the information.

Future Considerations

Throughout this thesis, I have pointed out several feature I would like to incorporate into future version of EpiProt. The main one, and most challenging, is making it web-based. That would require a overhaul of the View, and a way to place all the various elements on a server. With a little more knowledge, guidance and time, making it web-based could be a possibility.

As mentioned, the BLAST service uses UniProtJAPI to perform a BLAST search. I made this decision because UniProtJAPI was developed using Java making integration easy. The problem I ran into was the developers of UniProtJAPI decided to change not only the server links of the BLAST service, they changed parameters and the general structure of how the BLAST search executes and returns results. This resulted in me refactoring the BLAST service after it was developed. Seeing the BLAST service is critical in the antigen design process, it makes sense to reconstruct EpiProt with a more stable BLAST service in the future.

There are some software design basics that I would like to incorporate into later versions. One is to make all the services asynchronous. Right now, once a user initiates a service, that user cannot perform any other functions until that service is complete. I would also like to include more progress bars and error messages when running services. This would give the user a better sense of what is happening when a service is executed. Providing more text formatting tools (such as highlighting) would be something I would like to include in future versions.

Some features that I could not include due to time, but would like to in the near future include Ellipro, Discotope and peptide BLAST. The value these tools would bring to EpiProt would greatly increase its potential.

Summary

The importance of antibodies in research (Biocompare, 2015.), diagnostics (Siddiqui, 2010)) and therapeutics (Brekke & Inger, 2003; Chames, Patrick, Van Regenmortel, Etienne, & Daniel, 2009) is indisputable today and for the foreseeable future. As such, designing antigens that produce an antibody that binds to a selected target in the right application is of crucial importance. Designing the right antigen from the start can curb the number of attempts to produce the desired antibody. Since the development of a desired antibody can take months (Leenaars & Hendriksen, 2005)), getting it right the first time can save a significant amount of time and money, not only for the producer but also for the buyer (Bradbury & Plückthun, 2015). Therefore the demand to create tools to aid in antigen design is high.

Many tools and programs have been developed to meet that demand. Many have attempted to answer the question, "What is the best region to target for antigen design?". While this is an important question to answer, it is difficult to answer because how one designs antigens is dependent on what the user wants that antibody to do. Therefore EpiProt was constructed not for the purpose of finding "the best antigen", but to bring together as much protein and epitope prediction information as possible to let the user to decide.

I can say confidently EpiProt does provide enough information for the user to decide the best regions to target for antigen design. Not only does it provide BLAST, MSA, 2 secondary structure programs, PDB/SIFTS data, 9 epitope prediction programs, topology and PTM data, it provides the information in alignment with the target protein sequence. It also provides basic text editing capabilities with the ability to save and open projects. Some programs provide these features, however none quite as well as EpiProt.

Its MVP design makes it easy to fix bugs and add new features. The ultimate goal is to design a web-based version, which would make it more accessible. Since it has an MVP design, changing it to be web based should not require a major overhaul of the presenter and model. However, the challenges are still significant requiring a substantial amount of time and resources to complete.

An interesting feature to add would be to allow the user to define what information or tools they think are the best at showing the best epitopes. Based on their input, EpiProt could theoretically insert a line that shows a score for each amino acid the parameters they had set. This would fit the main mission of EpiProt as a tool that brings together protein and epitope information in order to let the user decide the best epitope regions.

Future versions of EpiProt may expand the functionality of EpiProt beyond its role as an antigen design tool for antibody. One direction to go is to include more protein information such as phylogenetic tree analysis or Jmol. Another direction is to include different types of epitope prediction tools, such as T-cell receptor epitope prediction ("T Cell Tools," 2016.). Adding new functionalities is desirable if the role of EpiProt is expanded from an "antibody production" tool to a complete protein bioinformatics suite.

Appendix

Additional Tables

Table 1.

Summary of the epitope prediction programs available in EpiProt

EPP	Successfully Integrated?	Input	Citation/Website	Description
BepiPred	No	Primary Sequence	Larsen, Lund, Nielsen, 2006. http://tools.immuneepitope.org/bcell/	Uses a combination of hidden Markov modeling and propensity scaling to determine the best epitope regions
Chou and Fasman	Yes	Primary Sequence	Chou & Gerald, 1978. http://tools.immuneepitope.org/bcell/	Using a scale to predict beta-turns, this tool identifies potential beta-turn regions which they argue is prime regions for epitopes
Emini surface accessibility	Yes	Primary Sequence	Emini, Hughes, Perlow, Boger, 1985. http://tools.immuneepitope.org/bcell/	Predicts epitope regions based on regions that are predicted to be exposed
Karplus and Schulz flexibility scale	Yes	Primary Sequence	Karplus & Schulz, 1985. http://tools.immuneepitope.org/bcell/	Predicts flexible regions, which they argue, are more antigenic.
Kolaskar and	Yes	Primary	Kolaskar &	Uses

Tongaonkar antigenicity scale		Sequence	Tongaonkar, 1990. http://tools.immuneepitope.org/bcell/	physicochemical properties and amino acid frequencies in known epitopes to predict epitopes
Parker Hydrophilicity	Yes	Primary Sequence	Parker, Guo & Hodges, 1986. http://tools.immuneepitope.org/bcell/	Evaluates the protein and determines which regions are more hydrophilic. It is argued that the more hydrophilic the region is the more likely it is to an epitope.
ABCpred	Yes	Primary Sequence	Saha & Raghava, 2006. http://www.imtech.res.in/raghava/abcpred/	Uses an artificial neural network trained on a known, cleaned epitope dataset from the Bcipep database to recognize patterns in epitopes
BcePred	Yes	Primary Sequence	Saha & Raghava, 2004. http://www.imtech.res.in/raghava/bcepred/	Combines the six IEDB epitope prediction programs to predict epitope.
ElliPro	No	PDB Structure	Ponomarenko, 2008. http://tools.iedb.org/ellipro/	Analyzes 3D structure to determine protrusion regions which are argued to be better epitope sites
DiscoTope	No	PDB Structure	Kringelum, Lundegaard, Lund, Nielsen, 2012. http://www.cbs.dtu.dk/services/DiscoTope/	Determines surface accessibility and combines that data with a novel

				amino acid epitope propensity scale to predict epitopes

Table 2.

Summary of all the services available in EpiProt, MacVector, Jalview and AbDesigner

		EpiProt	Jalview	MacVector	AbDesigner
Protein Information	Protein Sequence	Yes	Yes	Yes	Yes
	PTMs (UniProt/PhosphoSite Plus)	Yes	No	No	Yes**
	Subcellular location and Protein Processing	Yes	No	No	Yes
Alignments and BLAST	MSA	Yes	Yes	Yes	No
	BLAST	Yes	No	Yes	No
	Ability to BLAST peptide regions	Yes	No	No	Yes
Structure	SSP	Yes	Yes	Yes*	Yes
	PDB structure	Yes	Yes	No	No
	Jmol	No	Yes	No	No
Epitope Prediction Tools	IEDB Epitope Prediction Tools	Yes	No	Yes*	No
	ABCpred	Yes	No	No	No
	Bcepred	Yes	No	No	No
	Ig-score	No	No	No	Yes
Other	Phylogenetic Tree	No	Yes	Yes	No
	Ranks Peptides	No	No	No	Yes

*Note: *Does not display in alignment with main protein sequence. **Only shows glycosylation sites from UniProt.*

Table 3.

A table of the epitope prediction programs available with BcePred

Epitope Prediction Program Name (on BcePred)	Citation	Description	On IEDB?
Polarity	(Ponnuswamy, Prabhakaran, & Manavalan, 1980)	Predicts epitope regions based on a scale of hydrophobic indices for each amino acid residue.	No
Accessibility	(Emini, Hughes, Perlow, Boger, 1985.)	Predicts epitope regions based on regions that are predicted to be exposed	Yes
Flexibility	(Karplus & Schulz, 1985.)	Predicts flexible regions, which they argue, are more antigenic.	Yes
Antigenic Propensity	(Kolaskar & Tongaonkar, 1990.)	Uses physicochemical properties and amino acid frequencies in known epitopes to predict epitopes	Yes
Hydrophilicity	(Parker, Guo & Hodges, 1986.)	Evaluates the protein and determines which regions are more hydrophilic. It is argued that the more hydrophilic the region is the more likely it is to be an epitope.	Yes
Exposed Surface	(Janine, Joël, Shoshanna, Michael, & Bernard, 1978)	Predicts epitope regions based on the exposure of those regions.	No
Turns	(Pellequer, Westhof, & Van Regenmortel, 1993)	Predicts epitope regions based on regions that are predicted to be turns.	No

Note: Many are available at http://tools.iedb.org/bcell/.

Source Code

```
package epiprot.services.epitopePrediction;


public class ABCPredPeptide {


	private int rank;

	private String sequence;

	private int start;

	private double score;


	public ABCPredPeptide() {

		// TODO Auto-generated constructor stub

	}


	public int getRank() {

		return rank;

	}


	public void setRank(int rank) {

		this.rank = rank;
```

```
}

public String getSequence() {
	return sequence;
}

public void setSequence(String sequence) {
	this.sequence = sequence;
}

public int getStart() {
	return start;
}

public void setStart(int start) {
	this.start = start;
}

public double getScore() {
	return score;
}

public void setScore(double score) {
		this.score = score;
	}


	@Override
	public String toString() {
		return "Rank: " + rank + " Sequence: " + sequence + " start: " + start + "
score: " + score;
	}


}
package epiprot.services.views;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;


import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;
import epiprot.Presenter;
import epiprot.services.epitopePrediction.ABCPredPeptide;
import epiprot.services.epitopePrediction.ABCPredService;

public class ABCPredPresenter {

	private Presenter presenter;

	public interface View {
		JTextField textField();
		JComboBox comboBox();
		JCheckBox chckbxOverlappingFilter();
		JButton btnSubmit();
	}

	public interface SummaryView {
		JTextPane textPane();
		void appendString(String str);
	}

	View view = new ABCPredView();
	SummaryView summaryView = new ABCPredSummaryView();
	public ABCPredPresenter(Presenter presenter) {
		// TODO Auto-generated constructor stub
		this.presenter = presenter;
		bindHandlers();
	}

	void bindHandlers() {
		view.textField().getDocument().addDocumentListener(new
DocumentListener() {
			@Override
			public void insertUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
			@Override
			public void removeUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
			@Override
			public void changedUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
```

```
                checkTextField();
            }
            public void checkTextField() {
                String text = view.textField().getText();
                if(isDouble(text) && Double.parseDouble(text) < 1 &&
Double.parseDouble(text) >= 0.1) {
                    view.btnSubmit().setEnabled(true);
                }
                else {
                    view.btnSubmit().setEnabled(false);
                }
            }
        });

        view.btnSubmit().addActionListener(new ActionListener() {

            @Override
            public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                ABCPredService abcPredService = new
ABCPredService(presenter.protein.getSequence(),view.comboBox().getSelectedItem().to
String(),
            view.textField().getText(),view.chckbxOverlappingFilter().isSelected());


                                abcPredService.run();
                                ArrayList<ABCPredPeptide> peptideList =
abcPredService.getPeptides();
                                String line = "";
                                for(int i = 0; i < presenter.protein.getSequence().length();
i++) {
                                        line = line + " ";
                                }
                                for(ABCPredPeptide p: peptideList) {
                                        String s = String.format("%1$-10s %2$-25s %3$-
5d %4$.2f10\n", p.getRank(), p.getSequence(), p.getStart(), p.getScore());
                                        System.out.println(s);
                                        summaryView.appendString(s);
                                        for(int i = p.getStart()-1; i <
(p.getStart()+p.getSequence().length())-1; i++) {
                                                line =
line.substring(0,i)+"+"+line.substring(i+1);
                                        }
                                }


                                presenter.insertLineAboveTarget("ABCPred",
getInsertLine(line));

                        }

                });

        }

        public String getInsertLine(String inputLine) {
                String mainLine = presenter.getMainLine();
                for(int i = 0; i < mainLine.length(); i++) {
                        if(mainLine.charAt(i) == '-') {
                                inputLine = new StringBuilder(inputLine).insert(i, "
").toString();
                        }
                }
                return inputLine;
        }

        private boolean isDouble(String s) {
                try {
                        Double.parseDouble(s);
           } catch(NumberFormatException e) {
                return false;
            } catch(NullPointerException e) {
                return false;
            }
            // only got here if we didn't return false
            return true;
        }


}
package epiprot.services.epitopePrediction;


import java.io.IOException;
import java.util.ArrayList;
import java.util.List;


import com.gargoylesoftware.htmlunit.FailingHttpStatusCodeException;
import com.gargoylesoftware.htmlunit.WebClient;
import com.gargoylesoftware.htmlunit.html.DomElement;
import com.gargoylesoftware.htmlunit.html.HtmlForm;
import com.gargoylesoftware.htmlunit.html.HtmlOption;
import com.gargoylesoftware.htmlunit.html.HtmlPage;
import com.gargoylesoftware.htmlunit.html.HtmlRadioButtonInput;
import com.gargoylesoftware.htmlunit.html.HtmlSelect;
import com.gargoylesoftware.htmlunit.html.HtmlSubmitInput;

import com.gargoylesoftware.htmlunit.html.HtmlTextArea;

import com.gargoylesoftware.htmlunit.html.HtmlTextInput;

import epiprot.Protein;

import epiprot.services.Service;


public class ABCPredService extends Service {


        private String sequence;

        private String windowSize;

        private String threshold;

        private boolean isOverLapping;


        private ArrayList<ABCPredPeptide> peptideList = new
ArrayList<ABCPredPeptide>();


        public ABCPredService(String input, String windowSize, String threshold,
boolean isOverLapping) {

                // TODO Auto-generated constructor stub

                this.sequence = getSequence(input);

                this.windowSize = windowSize;

                this.threshold = threshold;

                this.isOverLapping = isOverLapping;
```

```
	}
	public ABCPredService(String input) {
		// TODO Auto-generated constructor stub
		this.sequence = getSequence(input);
		this.isOverLapping = true;
	}

	@Override
	public void run() {
		// TODO Auto-generated method stub
		final WebClient webClient = new WebClient();

	   // Get the first page
	   HtmlPage page1;
		try {
			page1 =
webClient.getPage("http://www.imtech.res.in/raghava/abcpred/ABC_submission.html");
			// Get the form that we are dealim ng with and within that form,
		   // find the submit button and the field that we want to change.
		   HtmlForm form = page1.getFormByName("form");

		   HtmlSubmitInput button = form.getInputByValue("Submit sequence");
		   HtmlTextArea seqTextField = form.getTextAreaByName("SEQ");
```

```
                    seqTextField.setText(sequence);


                    if (windowSize != null) {
                              HtmlSelect windowSelect = form.getSelectByName("window");
                              HtmlOption option =
windowSelect.getOptionByValue(windowSize);
                              windowSelect.setSelectedAttribute(option, true);
                    }
                    if (threshold != null) {
                         HtmlTextInput thresholdTextField =
form.getInputByName("Threshold");
                         thresholdTextField.setValueAttribute(threshold);
                    }


                    if (!isOverLapping) {
                         HtmlRadioButtonInput overLappingSelect =
form.getCheckedRadioButton("filter");
                         overLappingSelect.setValueAttribute("off");
                         overLappingSelect.click();
                 }


                    // Now submit the form by clicking the button and get back the second
page.
```

```
                    HtmlPage page2 = button.click();
                    List<DomElement> tables = page2.getElementsByTagName("table");
                  Iterable<DomElement> tableElement = tables.get(1).getChildElements();
                  java.util.Iterator<DomElement> it = tableElement.iterator();
                  while(it.hasNext()) {
                        DomElement peptideElement = it.next();
                        java.util.Iterator<DomElement> it2 =
peptideElement.getChildElements().iterator();
                        while(it2.hasNext()) {
                              DomElement peptideAttrsElement = it2.next();
                              Iterable<DomElement> peptides =
peptideAttrsElement.getChildElements();
                              java.util.Iterator<DomElement> it3 = peptides.iterator();
                              ABCPredPeptide peptide = new ABCPredPeptide();
                              int count = 0;
                              while(it3.hasNext()) {
                                    String attr = it3.next().asText();
                                    if(!attr.equals("Rank") && count == 0) {
                                          peptide.setRank(Integer.parseInt(attr));
                                    }
                                    else if(!attr.equals("Sequence") && count == 1) {
                                          peptide.setSequence(attr);
                                    }
```

```
                                        else if(!attr.equals("Start position") && count == 2)
{
                                             peptide.setStart(Integer.parseInt(attr));
                                        }
                                        else if(!attr.equals("Score") && count == 3) {
                                             peptide.setScore(Double.parseDouble(attr));
                                        }
                                        count++;
                                   }
                                   if (peptide.getSequence() != null) {
                                        peptideList.add(peptide);
                                   }
                              }


                         }
                           webClient.close();
                         } catch (FailingHttpStatusCodeException | IOException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                         }
                    }

                    private String getSequence(String input) {
```

```
		if (input.length() == 6) {
			Protein protein = new Protein(input,true);
			return protein.getSequence();
		}
		return input;
	}

	public ArrayList<ABCPredPeptide> getPeptides() {
		return peptideList;
	}

	public static void main (String[]args) {
		ABCPredService abcPredService = new
ABCPredService("P31749","16","0.7",true);
		abcPredService.run();
		ArrayList<ABCPredPeptide> peptideList = abcPredService.getPeptides();
		for(ABCPredPeptide p: peptideList) {
			String s = String.format("%1$-10s %2$-25s %3$-5d %4$.2f10",
p.getRank(), p.getSequence(), p.getStart(), p.getScore());
			System.out.println(s);
		}
	}
}
```

```
package epiprot.services.views;


import java.awt.Dimension;

import javax.swing.JFrame;

import javax.swing.JScrollPane;

import javax.swing.JTextPane;

import javax.swing.text.BadLocationException;

import javax.swing.text.StyledDocument;

import java.awt.BorderLayout;


public class ABCPredSummaryView extends JFrame implements

ABCPredPresenter.SummaryView {


	private JTextPane textPane;


	public ABCPredSummaryView() {

		setTitle("ABCPred Summary");

		setSize(new Dimension(400, 600));


		JScrollPane scrollPane = new JScrollPane();

		getContentPane().add(scrollPane, BorderLayout.CENTER);


		textPane = new JTextPane();
		scrollPane.setViewportView(textPane);


		appendString(String.format("%1$-10s %2$-25s %3$-5s %4$10s\n",
"Rank", "Sequence", "Start", "Score"));


		setVisible(true);
	}


	@Override
	public JTextPane textPane() {
		// TODO Auto-generated method stub
		return textPane;
	}


	@Override
	public void appendString(String str) {
		StyledDocument document = (StyledDocument) textPane.getDocument();
	   try {
			document.insertString(document.getLength(), str, null);
		} catch (BadLocationException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}




}
package epiprot.services.views;


import javax.swing.JFrame;
import javax.swing.JLabel;


import java.awt.Dimension;
import com.jgoodies.forms.layout.FormLayout;
import com.jgoodies.forms.layout.ColumnSpec;
import com.jgoodies.forms.layout.RowSpec;
import com.jgoodies.forms.layout.FormSpecs;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.JButton;
import javax.swing.JCheckBox;

public class ABCPredView extends JFrame implements ABCPredPresenter.View  {


	private JTextField textField;
private JComboBox comboBox;

private JCheckBox chckbxOverlappingFilter;

private JButton btnSubmit;


public ABCPredView() {

        setTitle("ABCPred");

        setSize(new Dimension(400, 200));


        getContentPane().setLayout(new FormLayout(new ColumnSpec[] {

                        FormSpecs.RELATED_GAP_COLSPEC,

                        FormSpecs.DEFAULT_COLSPEC,

                        FormSpecs.RELATED_GAP_COLSPEC,

                        ColumnSpec.decode("default:grow"),

                        FormSpecs.RELATED_GAP_COLSPEC,

                        FormSpecs.DEFAULT_COLSPEC,},

                new RowSpec[] {

                        FormSpecs.RELATED_GAP_ROWSPEC,

                        FormSpecs.DEFAULT_ROWSPEC,

                        FormSpecs.RELATED_GAP_ROWSPEC,

                        FormSpecs.DEFAULT_ROWSPEC,

                        FormSpecs.RELATED_GAP_ROWSPEC,

                        FormSpecs.DEFAULT_ROWSPEC,

                        FormSpecs.RELATED_GAP_ROWSPEC,
                        FormSpecs.DEFAULT_ROWSPEC,
                        FormSpecs.RELATED_GAP_ROWSPEC,
                        FormSpecs.DEFAULT_ROWSPEC,
                        FormSpecs.RELATED_GAP_ROWSPEC,
                        FormSpecs.DEFAULT_ROWSPEC,
                        FormSpecs.RELATED_GAP_ROWSPEC,
                        FormSpecs.DEFAULT_ROWSPEC,
                        FormSpecs.RELATED_GAP_ROWSPEC,
                        FormSpecs.DEFAULT_ROWSPEC,}));

        JLabel lblNewLabel = new JLabel("Window Size");
        getContentPane().add(lblNewLabel, "4, 4");


        String[] windowSizes = {"10","12","14","16","18","20"};
        comboBox = new JComboBox(windowSizes);
        comboBox.setSelectedItem("16");
        getContentPane().add(comboBox, "4, 6, fill, default");


        JLabel lblThreshold = new JLabel("Threshold [0.1 to 1.0]");
        getContentPane().add(lblThreshold, "4, 8");


        textField = new JTextField();
        getContentPane().add(textField, "4, 10, fill, default");
```

```
		textField.setText("0.85");
		textField.setColumns(10);

		chckbxOverlappingFilter = new JCheckBox("Overlapping filter? (check =
on, uncheck = off)");
		chckbxOverlappingFilter.setSelected(true);
		getContentPane().add(chckbxOverlappingFilter, "4, 12");



		btnSubmit = new JButton("Submit");
		getContentPane().add(btnSubmit, "4, 16");
		// TODO Auto-generated constructor stub

		setVisible(true);

	}

	@Override
	public JTextField textField() {
		// TODO Auto-generated method stub
		return textField;
	}
	@Override
	public JComboBox comboBox() {
		// TODO Auto-generated method stub
		return comboBox;
	}

	@Override
	public JCheckBox chckbxOverlappingFilter() {
		// TODO Auto-generated method stub
		return chckbxOverlappingFilter;
	}

	@Override
	public JButton btnSubmit() {
		// TODO Auto-generated method stub
		return btnSubmit;
	}

}
package epiprot.services.epitopePrediction;

import epiprot.AminoAcid;
```

```
public class BcePredAminoAcid extends AminoAcid {


        private double hydro;

        private double flexi;

        private double access;

        private double turns;

        private double surface;

        private double polar;

        private double antiPro;

        private double max;

        private double min;

        private double average;


        private int hydroRelative;

        private int flexiRelative;

        private int accessRelative;

        private int turnsRelative;

        private int surfaceRelative;

        private int polarRelative;

        private int antiProRelative;

        private int maxRelative;

        private int minRelative;
private int averageRelative;

public BcePredAminoAcid() {
	// TODO Auto-generated constructor stub
	hydroRelative = -1;
	flexiRelative = -1;
	accessRelative = -1;
	turnsRelative = -1;
	surfaceRelative = -1;
	polarRelative = -1;
	antiProRelative = -1;
	maxRelative = -1;
	minRelative = -1;
	averageRelative = -1;
}

public double getHydro() {
	return hydro;
}

public void setHydro(double hydro) {
	this.hydro = hydro;
}
```

```
public double getFlexi() {

        return flexi;

}


public void setFlexi(double flexi) {

        this.flexi = flexi;

}


public double getAccess() {

        return access;

}


public void setAccess(double access) {

        this.access = access;

}


public double getTurns() {

        return turns;

}


public void setTurns(double turns) {

        this.turns = turns;
```

```
}

public double getSurface() {
	return surface;
}

public void setSurface(double surface) {
	this.surface = surface;
}

public double getPolar() {
	return polar;
}

public void setPolar(double polar) {
	this.polar = polar;
}

public double getAntiPro() {
	return antiPro;
}

public void setAntiPro(double antiPro) {
        this.antiPro = antiPro;
}

public double getMax() {
        return max;
}

public void setMax(double max) {
        this.max = max;
}

public double getMin() {
        return min;
}

public void setMin(double min) {
        this.min = min;
}

public double getAverage() {
        return average;
}
```

```
        public void setAverage(double average) {
                this.average = average;
        }

        @Override
        public String toString() {
                return "Res: "+super.getResidue()+super.getPosition()+" H: "+hydro+" F:
"+ flexi+" A: "+ access+" T: "+ turns+" S: "+ surface+" P: "+ polar+" AP: "+
                                antiPro+" Max: "+ max+" Min: "+ min+" Avg: "+ average;
        }

        public int getHydroRelative() {
                return hydroRelative;
        }

        public void setHydroRelative(int hydroRelative) {
                this.hydroRelative = hydroRelative;
        }

        public int getFlexiRelative() {
                return flexiRelative;
        }
```

```
public void setFlexiRelative(int flexiRelative) {
        this.flexiRelative = flexiRelative;
}

public int getAccessRelative() {
        return accessRelative;
}

public void setAccessRelative(int accessRelative) {
        this.accessRelative = accessRelative;
}

public int getTurnsRelative() {
        return turnsRelative;
}

public void setTurnsRelative(int turnsRelative) {
        this.turnsRelative = turnsRelative;
}

public int getSurfaceRelative() {
        return surfaceRelative;
}
```

```
public void setSurfaceRelative(int surfaceRelative) {
        this.surfaceRelative = surfaceRelative;
}

public int getPolarRelative() {
        return polarRelative;
}

public void setPolarRelative(int polarRelative) {
        this.polarRelative = polarRelative;
}

public int getAntiProRelative() {
        return antiProRelative;
}

public void setAntiProRelative(int antiProRelative) {
        this.antiProRelative = antiProRelative;
}

public int getMaxRelative() {
        return maxRelative;
```

```
	}

	public void setMaxRelative(int maxRelative) {
		this.maxRelative = maxRelative;
	}

	public int getMinRelative() {
		return minRelative;
	}

	public void setMinRelative(int minRelative) {
		this.minRelative = minRelative;
	}

	public int getAverageRelative() {
		return averageRelative;
	}

	public void setAverageRelative(int averageRelative) {
		this.averageRelative = averageRelative;
	}
}
package epiprot.services.views;
```

```
import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.ArrayList;


import javax.swing.JButton;

import javax.swing.JCheckBox;

import javax.swing.JTextField;

import javax.swing.JTextPane;

import javax.swing.event.DocumentEvent;

import javax.swing.event.DocumentListener;


import epiprot.Presenter;

import epiprot.services.epitopePrediction.BcePredAminoAcid;

import epiprot.services.epitopePrediction.BcePredService;

import epiprot.services.uniprot.UniProtAminoAcid;


public class BcePredPresenter {


        Presenter presenter;


        String mainLine;
```

```
private String hydro = "";
private String flexi = "";
private String access = "";
private String turns = "";
private String surface = "";
private String polar = "";
private String antipro = "";
private String max = "";
private String min = "";
private String average = "";

interface View {
        JTextField hydrophilicityTextField();
        JTextField accessibilityTextField();
        JTextField exposedSurfaceTextField();
        JTextField antegenicPropensityTextField();
        JTextField flexibilityTextField();
        JTextField turnsTextField();
        JTextField polarityTextField();
        JTextField combinedTextField();
        JCheckBox hydrophilicityCheckBox();
        JCheckBox accessibilityCheckBox();
        JCheckBox exposedSurfaceCheckBox();
        JCheckBox antegenicPropensityCheckBox();

        JCheckBox flexibilityCheckBox();

        JCheckBox turnsCheckBox();

        JCheckBox polarityCheckBox();

        JCheckBox combinedCheckBox();

        JCheckBox chckbxSelectAll();

        JButton btnSubmit();

}


interface SummaryView {

        JTextPane textPane();

        void appendString(String str);

}


public BcePredPresenter(Presenter presenter) {

        // TODO Auto-generated constructor stub

        this.presenter = presenter;

        bindHandlers();



}


View view = new BcePredView();
```

```
        public void bindHandlers() {
                view.btnSubmit().addActionListener(new ActionListener() {
                        @Override
                        public void actionPerformed(ActionEvent e) {
                                mainLine = presenter.getMainLine();
                                for(int i = 0; i < presenter.protein.getSequence().length();
i++) {
                                        addDashToAllLines();
                                }
                                BcePredService bps = new
BcePredService(presenter.proteinAcc, view.combinedTextField().getText(),
view.polarityTextField().getText(),
                                                view.turnsTextField().getText(),
view.flexibilityTextField().getText(), view.antegenicPropensityTextField().getText(),
                                                view.exposedSurfaceTextField().getText(),
view.hydrophilicityTextField().getText(), view.accessibilityTextField().getText());

        bps.setSelectAccess(view.accessibilityCheckBox().isSelected());

        bps.setSelectAntipro(view.antegenicPropensityCheckBox().isSelected());

        bps.setSelectFlexi(view.flexibilityCheckBox().isSelected());
        bps.setSelectHydro(view.hydrophilicityCheckBox().isSelected());
                                bps.setSelectPolar(view.polarityCheckBox().isSelected());

        bps.setSelectSurface(view.exposedSurfaceCheckBox().isSelected());
                                bps.setSelectTurns(view.turnsCheckBox().isSelected());

                                bps.run();

                                ArrayList<BcePredAminoAcid> aaList =
bps.getAminoAcids();

                                ArrayList<LineElement> accessLineList = getLineList();
                                ArrayList<LineElement> antiproLineList = getLineList();
                                ArrayList<LineElement> averageLineList = getLineList();
                                ArrayList<LineElement> flexiLineList = getLineList();
                                ArrayList<LineElement> hydroLineList = getLineList();
                                ArrayList<LineElement> maxLineList = getLineList();
                                ArrayList<LineElement> minLineList = getLineList();
                                ArrayList<LineElement> polarLineList = getLineList();
                                ArrayList<LineElement> surfaceLineList = getLineList();
                                ArrayList<LineElement> turnsLineList = getLineList();

                                for(int i = 0; i < aaList.size(); i++) {
                                        BcePredAminoAcid aa = aaList.get(i);
                                        System.out.println(aa.toString());
                                        if(aa.getAccessRelative() != -1 && aa.getAccess()
> Double.parseDouble(view.accessibilityTextField().getText())) {
                                             accessLineList.get(i).setCharacter(
aa.getAccessRelative());
                                             accessLineList.get(i).setColor("blue");
                                        }
                                        else if(aa.getAccessRelative() != -1) {
                                             accessLineList.get(i).setCharacter(
aa.getAccessRelative());
                                        }

                                        if(aa.getAntiProRelative() != -1 && aa.getAntiPro()
> Double.parseDouble(view.antegenicPropensityTextField().getText())) {
                                             antiproLineList.get(i).setCharacter(
aa.getAntiProRelative());
                                             antiproLineList.get(i).setColor("blue");
                                        }
                                        else if(aa.getAntiProRelative() != -1) {
                                             antiproLineList.get(i).setCharacter(
aa.getAntiProRelative());
                                        }
```

```
					if(aa.getAverageRelative() != -1 &&
aa.getAverage() > Double.parseDouble(view.combinedTextField().getText())) {
						averageLineList.get(i).setCharacter(
aa.getAverageRelative());
						averageLineList.get(i).setColor("blue");
					}
					else if(aa.getAverageRelative() != -1) {
						averageLineList.get(i).setCharacter(
aa.getAverageRelative());
					}

					if(aa.getFlexiRelative() != -1 && aa.getFlexi() >
Double.parseDouble(view.flexibilityTextField().getText())) {
						flexiLineList.get(i).setCharacter(
aa.getFlexiRelative());
						flexiLineList.get(i).setColor("blue");
					}
					else if(aa.getFlexiRelative() != -1) {
						flexiLineList.get(i).setCharacter(
aa.getFlexiRelative());
					}
```

```
                                        if(aa.getHydroRelative() != -1 && aa.getHydro() >
Double.parseDouble(view.hydrophilicityTextField().getText())) {
                                                hydroLineList.get(i).setCharacter(
aa.getHydroRelative());
                                                hydroLineList.get(i).setColor("blue");
                                        }
                                        else if(aa.getHydroRelative() != -1) {
                                                hydroLineList.get(i).setCharacter(
aa.getHydroRelative());
                                        }

                                        if(aa.getMaxRelative() != -1 && aa.getMax() >
Double.parseDouble(view.combinedTextField().getText())) {
                                                maxLineList.get(i).setCharacter(
aa.getMaxRelative());
                                                maxLineList.get(i).setColor("blue");
                                        }
                                        else if(aa.getMaxRelative() != -1) {
                                                maxLineList.get(i).setCharacter(
aa.getMaxRelative());
                                        }

                                        if(aa.getMinRelative() != -1 && aa.getMin() >
Double.parseDouble(view.combinedTextField().getText())) {
                                                minLineList.get(i).setCharacter(
aa.getMinRelative());
                                                minLineList.get(i).setColor("blue");
                                        }
                                        else if(aa.getMinRelative() != -1) {
                                                minLineList.get(i).setCharacter(
aa.getMinRelative());
                                        }

                                        if(aa.getPolarRelative() != -1 && aa.getPolar() >
Double.parseDouble(view.polarityTextField().getText())) {
                                                polarLineList.get(i).setCharacter(
aa.getPolarRelative());
                                                polarLineList.get(i).setColor("blue");
                                        }
                                        else if(aa.getPolarRelative() != -1) {
                                                polarLineList.get(i).setCharacter(
aa.getPolarRelative());
                                        }

                                        if(aa.getSurfaceRelative() != -1 && aa.getSurface()
> Double.parseDouble(view.exposedSurfaceTextField().getText())) {
        surfaceLineList.get(i).setCharacter(aa.getSurfaceRelative());
                                        surfaceLineList.get(i).setColor("blue");
                                    }
                                    else if(aa.getSurfaceRelative() != -1) {
                                        surfaceLineList.get(i).setCharacter(
aa.getSurfaceRelative());
                                    }

                                    if(aa.getTurnsRelative() != -1 && aa.getTurns() >
Double.parseDouble(view.turnsTextField().getText())) {
                                        turnsLineList.get(i).setCharacter(
aa.getTurnsRelative());
                                        turnsLineList.get(i).setColor("blue");
                                    }
                                    else if(aa.getTurnsRelative() != -1) {
                                        turnsLineList.get(i).setCharacter(
aa.getTurnsRelative());
                                    }
                                }

                                access = getLine(accessLineList);
                                antipro = getLine(antiproLineList);
```

```
flexi = getLine(flexiLineList);

hydro = getLine(hydroLineList);

polar = getLine(polarLineList);

surface = getLine(surfaceLineList);

turns = getLine(turnsLineList);

average = getLine(averageLineList);

min = getLine(minLineList);

max = getLine(maxLineList);


System.out.println("access"+access);

System.out.println("antipro"+antipro);

System.out.println("flexi"+flexi);

System.out.println("hydro"+hydro);

System.out.println("polar"+polar);

System.out.println("surface"+surface);

System.out.println("turns"+turns);

System.out.println("average"+average);

System.out.println("min"+min);

System.out.println("max"+max);



if (containsAnythingButDashAndZero(access)) {

        presenter.insertLineAboveTarget("BcePred:
```

```
Accessible (Emini)", access);
                                }
                                if (containsAnythingButDashAndZero(antipro)) {
                                        presenter.insertLineAboveTarget("BcePred:
Antigenic Propensity (Kolaskar)", antipro);
                                }
                                if (containsAnythingButDashAndZero(flexi)) {
                                        presenter.insertLineAboveTarget("BcePred:
Flexibility (Karplus)", flexi);
                                }
                                if (containsAnythingButDashAndZero(hydro)) {
                                        presenter.insertLineAboveTarget("BcePred:
Hydrophilicity (Parker)", hydro);
                                }
                                if (containsAnythingButDashAndZero(polar)) {
                                        presenter.insertLineAboveTarget("BcePred:
Polarity (Ponnuswamy)", polar);
                                }
                                if (containsAnythingButDashAndZero(surface)) {
                                        presenter.insertLineAboveTarget("BcePred:
Exposed Surface (Janin)", surface);
                                }
                                if (containsAnythingButDashAndZero(turns)) {
                                        presenter.insertLineAboveTarget("BcePred: Turns
(Pellequer)", turns);
                                }
                                if (containsAnythingButDashAndZero(average)) {
                                        presenter.insertLineAboveTarget("BcePred:
Average", average);
                                }
                                if (containsAnythingButDashAndZero(min)) {
                                        presenter.insertLineAboveTarget("BcePred: Min",
min);
                                }
                                if (containsAnythingButDashAndZero(max)) {
                                        presenter.insertLineAboveTarget("BcePred: Max",
max);
                                }
                        }
                });

                view.accessibilityTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.accessibilityTextField()));

        view.antegenicPropensityTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.antegenicPropensityTextField()));
```

```
                    view.combinedTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.combinedTextField()));


          view.exposedSurfaceTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.exposedSurfaceTextField()));
                    view.flexibilityTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.flexibilityTextField()));
                    view.hydrophilicityTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.hydrophilicityTextField()));
                    view.polarityTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.polarityTextField()));
                    view.turnsTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.turnsTextField()));


                    view.accessibilityCheckBox().addActionListener(new
CheckBoxListener(view));
                    view.antegenicPropensityCheckBox().addActionListener(new
CheckBoxListener(view));
                    view.exposedSurfaceCheckBox().addActionListener(new
CheckBoxListener(view));
                    view.flexibilityCheckBox().addActionListener(new
CheckBoxListener(view));
                    view.hydrophilicityCheckBox().addActionListener(new
CheckBoxListener(view));
                view.polarityCheckBox().addActionListener(new
CheckBoxListener(view));
                view.turnsCheckBox().addActionListener(new CheckBoxListener(view));

                view.chckbxSelectAll().addActionListener(new
CheckBoxListener(view));
                view.chckbxSelectAll().addActionListener(new ActionListener() {

                        @Override
                        public void actionPerformed(ActionEvent e) {
                                // TODO Auto-generated method stub
                                if (view.chckbxSelectAll().isSelected()) {
                                        view.accessibilityCheckBox().setSelected(true);
                                        view.combinedCheckBox().setSelected(true);

        view.antegenicPropensityCheckBox().setSelected(true);
                                        view.exposedSurfaceCheckBox().setSelected(true);
                                        view.flexibilityCheckBox().setSelected(true);
                                        view.hydrophilicityCheckBox().setSelected(true);
                                        view.polarityCheckBox().setSelected(true);
                                        view.turnsCheckBox().setSelected(true);
                                        view.btnSubmit().setEnabled(true);
```

```
                    }
                    else {
                              view.accessibilityCheckBox().setSelected(false);
                              view.combinedCheckBox().setSelected(false);

view.antegenicPropensityCheckBox().setSelected(false);
                              view.exposedSurfaceCheckBox().setSelected(false);
                              view.flexibilityCheckBox().setSelected(false);
                              view.hydrophilicityCheckBox().setSelected(false);
                              view.polarityCheckBox().setSelected(false);
                              view.turnsCheckBox().setSelected(false);
                              view.btnSubmit().setEnabled(false);
                    }
          }


     });

     view.combinedCheckBox().addActionListener(new ActionListener() {

          @Override
          public void actionPerformed(ActionEvent e) {
                    // TODO Auto-generated method stub
```

```
            }

        });

    }

    private String getLine(ArrayList<LineElement> lineList) {
        String line = "";
        for(int i = 0; i < mainLine.length(); i++) {
            System.out.print(mainLine.charAt(i) == '-' ? "*" : "f");
            if(mainLine.charAt(i) == '-') {
                lineList.add(i,new LineElement(-1));
            }
        }
        System.out.println();
        for (LineElement le : lineList) {
            if(le.color == null && le.character != -1) {
                line = line + le.getCharacter();
            }
            else if(le.color == null && le.character == -1) {
                line = line + " ";
            }
```

```
                else {
                        line = line + getBlueColoredText(le.character);
                }
        }
        return line;
}

private String getBlueColoredText(int text) {
        return "<font color=\"blue\">"+text+"</font>";
}
private boolean containsAnythingButDashAndZero(String s) {
        for(int i = 0; i < s.length(); i++) {
                if (s.charAt(i) != '-' && s.charAt(i) != '0') {return true;}
        }
        return false;
}

private boolean isDouble(String s) {
        try {
      Double.parseDouble(s);
   } catch(NumberFormatException e) {
      return false;
   } catch(NullPointerException e) {
```

```
            return false;
          }
          // only got here if we didn't return false
          return true;
        }

        public void checkTextField(String text) {
                if(isDouble(text) && Double.parseDouble(text) >= -3 &&
Double.parseDouble(text) <= 3) {
                        view.btnSubmit().setEnabled(true);
                }
                else {
                        view.btnSubmit().setEnabled(false);
                }
        }

        public boolean isTextFieldOK(JTextField textField) {
                return (isDouble(textField.getText()) &&
Double.parseDouble(textField.getText()) >= -3 &&
Double.parseDouble(textField.getText()) <= 3);
        }

        public boolean isAllTextFieldsOK() {
```

```
		if(!isTextFieldOK(view.accessibilityTextField())) {return false;}
		if(!isTextFieldOK(view.antegenicPropensityTextField())) {return false;}
		if(!isTextFieldOK(view.combinedTextField())) {return false;}
		if(!isTextFieldOK(view.exposedSurfaceTextField())) {return false;}
		if(!isTextFieldOK(view.flexibilityTextField())) {return false;}
		if(!isTextFieldOK(view.hydrophilicityTextField())) {return false;}
		if(!isTextFieldOK(view.polarityTextField())) {return false;}
		if(!isTextFieldOK(view.turnsTextField())) {return false;}
		return true;
	}

	public ArrayList<UniProtAminoAcid> getAminoAcids() {
		ArrayList<UniProtAminoAcid> aaList = new
ArrayList<UniProtAminoAcid>();
		for (int i = 0; i< presenter.protein.getSequence().length(); i++) {
			aaList.add(new UniProtAminoAcid());
		}
		return aaList;
	}

	private void addDashToAllLines() {
		hydro = hydro + "-";
		flexi = flexi +"-";
```

```
            access = access+"-";
    turns = turns+"-";
            surface = surface+"-";
            polar = polar+"-";
            antipro = antipro+"-";
            max = max+"-";
            min = min+"-";
            average = average +"-";
}

private ArrayList<LineElement> getLineList() {
            ArrayList<LineElement> lineList = new ArrayList<LineElement>();
            for(int i = 0; i < presenter.protein.getSequence().length(); i++) {
                        LineElement le = new LineElement();
                        lineList.add(le);
            }
            return lineList;
}

public class SubmitButtonListener implements DocumentListener {

            JTextField textField;
```

```
		public SubmitButtonListener(JTextField textField){
			this.textField = textField;
		}

		@Override
		public void insertUpdate(DocumentEvent e) {
			// TODO Auto-generated method stub
			checkTextField(textField.getText());
		}

		@Override
		public void removeUpdate(DocumentEvent e) {
			// TODO Auto-generated method stub
			checkTextField(textField.getText());
		}

		@Override
		public void changedUpdate(DocumentEvent e) {
			// TODO Auto-generated method stub
			checkTextField(textField.getText());
		}

	}
	public class CheckBoxListener implements ActionListener {

		View bcePredView;

		public CheckBoxListener(View bcePredView) {
			this.bcePredView = bcePredView;
		}

		@Override
		public void actionPerformed(ActionEvent e) {
			// TODO Auto-generated method stub
			if ((bcePredView.accessibilityCheckBox().isSelected() ||
bcePredView.antegenicPropensityCheckBox().isSelected() ||

	bcePredView.exposedSurfaceCheckBox().isSelected() ||
bcePredView.flexibilityCheckBox().isSelected() ||

	bcePredView.hydrophilicityCheckBox().isSelected() ||
bcePredView.polarityCheckBox().isSelected() ||
bcePredView.turnsCheckBox().isSelected()) &&
						isAllTextFieldsOK()) {
					view.btnSubmit().setEnabled(true);
```

```
                        }
                        else {
                                view.btnSubmit().setEnabled(false);
                        }



        if(toInt(view.hydrophilicityCheckBox().isSelected())+toInt(view.flexibilityCheck
Box().isSelected())+toInt(view.accessibilityCheckBox().isSelected())+toInt(view.turnsCh
eckBox().isSelected())+toInt(view.exposedSurfaceCheckBox().isSelected())+toInt(view.
polarityCheckBox().isSelected())+toInt(view.antegenicPropensityCheckBox().isSelected(
)) >= 2) {
                                view.combinedCheckBox().setEnabled(true);
                                view.combinedTextField().setEnabled(true);
                        }
                        else {
                                view.accessibilityCheckBox().setSelected(false);
                                view.combinedCheckBox().setEnabled(false);
                                view.combinedTextField().setEnabled(false);
                        }
                }

        }
```

```
private int toInt(boolean bool) {
        return bool ? 1 : 0;
}


private class LineElement {
private int character;
private int positionInLine;
private String color;
public LineElement(){}
public LineElement(int character){
        this.character = character;
}
public LineElement(int character, int positionInLine, String color){
        this.setCharacter(character);
        this.setPositionInLine(positionInLine);
        this.setColor(color);
}
public int getCharacter() {
                return character;
        }
        public void setCharacter(int character) {
                this.character = character;
```

```
			}
			public String getColor() {
				return color;
			}
			public void setColor(String color) {
				this.color = color;
			}
			public int getPositionInLine() {
				return positionInLine;
			}
			public void setPositionInLine(int positionInLine) {
				this.positionInLine = positionInLine;
			}
			@Override
			public String toString() {
				return character+"|"+color;
			}
	}

	private class ListPlusPosition {
		public String line;
		public int position;
		public ListPlusPosition(){}
```

```
            @Override

                        public String toString() {

                        return line+position;

            }

    }




}

package epiprot.services.epitopePrediction;


import java.io.IOException;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;


import com.gargoylesoftware.htmlunit.FailingHttpStatusCodeException;

import com.gargoylesoftware.htmlunit.WebClient;

import com.gargoylesoftware.htmlunit.html.DomElement;

import com.gargoylesoftware.htmlunit.html.HtmlForm;

import com.gargoylesoftware.htmlunit.html.HtmlInput;

import com.gargoylesoftware.htmlunit.html.HtmlOption;

import com.gargoylesoftware.htmlunit.html.HtmlPage;
import com.gargoylesoftware.htmlunit.html.HtmlSelect;

import com.gargoylesoftware.htmlunit.html.HtmlSubmitInput;

import com.gargoylesoftware.htmlunit.html.HtmlTextArea;


import epiprot.Protein;

import epiprot.services.Service;


public class BcePredService extends Service {


        private String sequence;

        private String hydrophilicity;

        private String accessibility;

        private String exposedSurface;

        private String antegenicPropensity;

        private String flexibility;

        private String turns;

        private String polarity;

        private String combined;


        private boolean selectHydro;

        private boolean selectFlexi;

        private boolean selectAccess;

        private boolean selectTurns;
```

```
	private boolean selectSurface;
	private boolean selectPolar;
	private boolean selectAntipro;
	private boolean selectAll;

	private ArrayList<BcePredAminoAcid> aminoAcids = new
ArrayList<BcePredAminoAcid>();

	public BcePredService(String input, String combined, String polarity, String
turns, String flexibility, String antegenicPropensity, String exposedSurface, String
hydrophilicity, String accessibility) {
		// TODO Auto-generated constructor stub
		this.sequence = getSequence(input);
		this.hydrophilicity = hydrophilicity;
		this.accessibility = accessibility;
		this.exposedSurface = exposedSurface;
		this.antegenicPropensity = antegenicPropensity;
		this.flexibility = flexibility;
		this.turns = turns;
		this.polarity = polarity;
		this.combined = combined;
	}
	public BcePredService(String input) {
```

```
            // TODO Auto-generated constructor stub
            this.sequence = getSequence(input);
    }

    @Override
    public void run() {
            // TODO Auto-generated method stub
            final WebClient webClient = new WebClient();

        // Get the first page
        HtmlPage page1;
            try {
                    page1 =
webClient.getPage("http://www.imtech.res.in/raghava/bcepred/bcepred_submission.html
");
                    // Get the form that we are dealim ng with and within that form,
                // find the submit button and the field that we want to change.
                HtmlForm form = page1.getFormByName("form");

                HtmlSubmitInput button = form.getInputByValue("Submit sequence");
                HtmlTextArea seqTextField = form.getTextAreaByName("SEQ");
                seqTextField.setText(sequence);
                List<HtmlInput> thresholds = form.getInputsByName("Threshold");

                if (hydrophilicity != null) {
                    thresholds.get(0).setValueAttribute(hydrophilicity);
                }
                if (flexibility != null) {
                    thresholds.get(1).setValueAttribute(flexibility);
                }
                if (accessibility != null) {
                    thresholds.get(2).setValueAttribute(accessibility);
                }
                if (turns != null) {
                    thresholds.get(3).setValueAttribute(turns);
                }
                if (exposedSurface != null) {
                    thresholds.get(4).setValueAttribute(exposedSurface);
            }
                if (polarity != null) {
                    thresholds.get(5).setValueAttribute(polarity);
                }
                if (antegenicPropensity != null) {
                    thresholds.get(6).setValueAttribute(antegenicPropensity);
                }
```

```
                    if (combined != null) {
                            thresholds.get(7).setValueAttribute(combined);
                    }

                    HtmlSelect physioChemicalSelect = form.getSelectByName("propno");
                    List<HtmlOption> optionList = physioChemicalSelect.getOptions();
                    //if (selectAll) {
                                for(HtmlOption option: optionList) {
                                        option.setSelected(true);
                                }
                   /* }
                    else {
                                optionList.get(0).setSelected(selectHydro);
                                optionList.get(1).setSelected(selectFlexi);
                                optionList.get(2).setSelected(selectAccess);
                                optionList.get(3).setSelected(selectTurns);
                                optionList.get(4).setSelected(selectSurface);
                                optionList.get(5).setSelected(selectPolar);
                                optionList.get(6).setSelected(selectAntipro);
                    }*/
                    // Now submit the form by clicking the button and get back the second
page.
                    HtmlPage page2 = button.click();
```

```
                String[] page2Array = page2.asText().split("\\r?\\n");
                printArray(page2Array);
                for(int i = 0; i < page2Array.length; i++) {
                    String line = page2Array[i];
                    if (line.contains("   Flexi")) {
                        line = page2Array[i+2];
                        String[] lineArray = line.split("(?=[A-Z])");
                        for(int j = 1; j < lineArray.length; j++) {
                            String[]lineAttrs = lineArray[j].split("\\s+");
                            BcePredAminoAcid aa = new
BcePredAminoAcid();
                            for (int k = 0; k < lineAttrs.length; k++) {
                                String attr = lineAttrs[k];
                                aa.setPosition(j);
                                if(k == 0) {aa.setResidue(attr);}
                                else {
                                    double d =
Double.parseDouble(attr);
                                    if(k == 1){aa.setHydro(d);}
                                    else if(k == 2){aa.setFlexi(d);}
                                    else if(k == 3){aa.setAccess(d);}
                                    else if(k == 4){aa.setTurns(d);}
```

```
                                        else if(k == 5){aa.setSurface(d);}
                                        else if(k == 6){aa.setPolar(d);}
                                        else if(k == 7){aa.setAntiPro(d);}
                                        else if(k == 8){aa.setMax(d);}
                                        else if(k == 9){aa.setMin(d);}
                                        else if(k == 10){aa.setAverage(d);}
                                    }
                                }
                                aminoAcids.add(aa);
                            }
                            break;
                        }
                    }
                    calcDisplayedScores();
                    DomElement element = page2.getElementByName("opt2");
                    //System.out.println(element.asXml());
                    webClient.close();
                } catch (FailingHttpStatusCodeException | IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
            }
```

```
	private void calcDisplayedScores() {
		int tenth = aminoAcids.size()/10;
	int remainder = aminoAcids.size()%10;


		if(selectHydro) {
			Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
			@Override
				public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
				if(o1.getHydro() == o2.getHydro())
				   return 0;
				return o1.getHydro() < o2.getHydro() ? -1 : 1;
			}
		});

		for(int i = 0; i < 10; i++) {
			for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
				aminoAcids.get(j).setHydroRelative(i);
			}
		}

		for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
```

```
                aminoAcids.get(i).setHydroRelative(9);
            }
            }


            if(selectFlexi) {
                Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
              @Override
                    public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                  if(o1.getFlexi() == o2.getFlexi())
                    return 0;
                  return o1.getFlexi() < o2.getFlexi() ? -1 : 1;
              }
            });


            for(int i = 0; i < 10; i++) {
                for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                    aminoAcids.get(j).setFlexiRelative(i);
                }
            }


            for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                aminoAcids.get(i).setFlexiRelative(9);
        }
        }


        if(selectAccess) {
                Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
            @Override
                        public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
              if(o1.getAccess() == o2.getAccess())
                 return 0;
              return o1.getAccess() < o2.getAccess() ? -1 : 1;
            }
        });


        for(int i = 0; i < 10; i++) {
                for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                        aminoAcids.get(j).setAccessRelative(i);
                }
        }


        for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
```

```
                        aminoAcids.get(i).setAccessRelative(9);
                }
                }

                if(selectTurns) {
                        Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
                   @Override
                                public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                      if(o1.getTurns() == o2.getTurns())
                         return 0;
                      return o1.getTurns() < o2.getTurns() ? -1 : 1;
                   }
                });

                for(int i = 0; i < 10; i++) {
                        for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                                aminoAcids.get(j).setTurnsRelative(i);
                        }
                }

                for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                    aminoAcids.get(i).setTurnsRelative(9);
                }
            }

            if(selectSurface) {
                    Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
                @Override
                        public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                    if(o1.getSurface() == o2.getSurface())
                        return 0;
                    return o1.getSurface() < o2.getSurface() ? -1 : 1;
                }
            });

            for(int i = 0; i < 10; i++) {
                    for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                            aminoAcids.get(j).setSurfaceRelative(i);
                    }
            }

            for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                    aminoAcids.get(i).setSurfaceRelative(9);
            }
            }

            if(selectPolar) {
                    Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
              @Override
                         public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                  if(o1.getPolar() == o2.getPolar())
                     return 0;
                  return o1.getPolar() < o2.getPolar() ? -1 : 1;
              }
            });

            for(int i = 0; i < 10; i++) {
                    for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                         aminoAcids.get(j).setPolarRelative(i);
                    }
            }

            for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                    aminoAcids.get(i).setPolarRelative(9);
            }
            }

            if(selectAntipro) {
                    Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
              @Override
                            public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                 if(o1.getAntiPro() == o2.getAntiPro())
                   return 0;
                 return o1.getAntiPro() < o2.getAntiPro() ? -1 : 1;
              }
            });

            for(int i = 0; i < 10; i++) {
                    for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                            aminoAcids.get(j).setAntiProRelative(i);
                    }
            }

            for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                    aminoAcids.get(i).setAntiProRelative(9);
            }
            }

            if(selectAntipro) {
                    Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
              @Override
                            public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                 if(o1.getAntiPro() == o2.getAntiPro())
                   return 0;
                 return o1.getAntiPro() < o2.getAntiPro() ? -1 : 1;
              }
            });

            for(int i = 0; i < 10; i++) {
                    for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                            aminoAcids.get(j).setAntiProRelative(i);
                    }
            }

            for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
```

```
                    aminoAcids.get(i).setAntiProRelative(9);
                }
            }



    if(toInt(selectHydro)+toInt(selectFlexi)+toInt(selectAccess)+toInt(selectTurns)+t
oInt(selectSurface)+toInt(selectPolar)+toInt(selectAntipro) >= 2) {
                    Collections.sort(aminoAcids, new
Comparator<BcePredAminoAcid>(){
                @Override
                        public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                    if(o1.getMax() == o2.getMax())
                        return 0;
                    return o1.getMax() < o2.getMax() ? -1 : 1;
                }
            });

            for(int i = 0; i < 10; i++) {
                for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                    aminoAcids.get(j).setMaxRelative(i);
                }
            }
```

```
                    for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                            aminoAcids.get(i).setMaxRelative(9);
                    }

                    Collections.sort(aminoAcids, new Comparator<BcePredAminoAcid>(){
                        @Override
                                    public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                            if(o1.getMin() == o2.getMin())
                                return 0;
                            return o1.getMin() < o2.getMin() ? -1 : 1;
                        }
                    });

                    for(int i = 0; i < 10; i++) {
                            for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                                    aminoAcids.get(j).setMinRelative(i);
                            }
                    }

                    for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                            aminoAcids.get(i).setMinRelative(9);
```

```
                }

                Collections.sort(aminoAcids, new Comparator<BcePredAminoAcid>(){
                    @Override
                                public int compare(BcePredAminoAcid o1,
BcePredAminoAcid o2){
                        if(o1.getAverage() == o2.getAverage())
                            return 0;
                        return o1.getAverage() < o2.getAverage() ? -1 : 1;
                    }
                });

                for(int i = 0; i < 10; i++) {
                        for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                                aminoAcids.get(j).setAverageRelative(i);
                        }
                }

                for (int i = aminoAcids.size()-1; i >= aminoAcids.size()-remainder; i--) {
                        aminoAcids.get(i).setAverageRelative(9);
                }
                }
```

```
        Collections.sort(aminoAcids, new Comparator<BcePredAminoAcid>(){
                        @Override
                        public int compare(BcePredAminoAcid o1, BcePredAminoAcid
    o2){
                            if(o1.getPosition() == o2.getPosition())
                                return 0;
                            return o1.getPosition() < o2.getPosition() ? -1 : 1;
                        }
        });
    }

        private int toInt(boolean bool) {
                return bool ? 1 : 0;
        }

        private String getSequence(String input) {
                if (input.length() == 6) {
                        Protein protein = new Protein(input,true);
                        return protein.getSequence();
                }
                return input;
        }
```

```
public ArrayList<BcePredAminoAcid> getAminoAcids() {
        return aminoAcids;
}


public boolean isSelectHydro() {
        return selectHydro;
}
public void setSelectHydro(boolean selectHydro) {
        this.selectHydro = selectHydro;
}
public boolean isSelectFlexi() {
        return selectFlexi;
}
public void setSelectFlexi(boolean selectFlexi) {
        this.selectFlexi = selectFlexi;
}
public boolean isSelectAccess() {
        return selectAccess;
}
public void setSelectAccess(boolean selectAccess) {
        this.selectAccess = selectAccess;
}
public boolean isSelectTurns() {
        return selectTurns;
}
public void setSelectTurns(boolean selectTurns) {
        this.selectTurns = selectTurns;
}
public boolean isSelectSurface() {
        return selectSurface;
}
public void setSelectSurface(boolean selectSurface) {
        this.selectSurface = selectSurface;
}
public boolean isSelectPolar() {
        return selectPolar;
}
public void setSelectPolar(boolean selectPolar) {
        this.selectPolar = selectPolar;
}
public boolean isSelectAntipro() {
        return selectAntipro;
}
public void setSelectAntipro(boolean selectAntipro) {
        this.selectAntipro = selectAntipro;
}
```

```
	public boolean isSelectAll() {
		return selectAll;
	}
	public void setSelectAll() {
		this.selectAll = true;
	}

	public static void printArray(String[]array) {
		for(String s: array) {
			System.out.println(s);
		}
	}

	public static void main (String[]args) {
		BcePredService bps = new
BcePredService("Q99523","1.9","2.3","1.9","1.9","1.8","2.4","2","2");

		//bps.setSelectAccess(true);
		bps.setSelectAll();
		bps.run();
		ArrayList<BcePredAminoAcid> aaList = bps.getAminoAcids();
		for(BcePredAminoAcid aa : aaList) {
			System.out.println(aa.toString());
```

```
                }
        }


}
package epiprot.services.views;


import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.border.TitledBorder;


import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JCheckBox;
import javax.swing.JButton;
import java.awt.Font;


public class BcePredView extends JFrame implements BcePredPresenter.View {
```

```
private JTextField hydrophilicityTextField;

private JTextField accessibilityTextField;

private JTextField exposedSurfaceTextField;

private JTextField antegenicPropensityTextField;

private JTextField flexibilityTextField;

private JTextField turnsTextField;

private JTextField polarityTextField;

private JTextField combinedTextField;

private JCheckBox hydrophilicityCheckBox;

private JCheckBox accessibilityCheckBox;

private JCheckBox exposedSurfaceCheckBox;

private JCheckBox antegenicPropensityCheckBox;

private JCheckBox flexibilityCheckBox;

private JCheckBox turnsCheckBox;

private JCheckBox polarityCheckBox;

private JCheckBox combinedCheckBox;

private JCheckBox chckbxSelectAll;

private JButton btnSubmit;

private JLabel lblNumbersMustBe;


public BcePredView() {

        setTitle("BcePred");

        // TODO Auto-generated constructor stub
```

```
setSize(new Dimension(610, 400));


JScrollPane scrollPane = new JScrollPane();

getContentPane().add(scrollPane, BorderLayout.CENTER);


JPanel panel = new JPanel();

scrollPane.setViewportView(panel);

GridBagLayout gbl_panel = new GridBagLayout();

gbl_panel.columnWidths = new int[] {260, 286};

gbl_panel.rowWeights = new double[]{1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0};

gbl_panel.columnWeights = new double[]{1.0,1.0};

panel.setLayout(gbl_panel);


JPanel panel_1 = new JPanel();

GridBagConstraints gbc_panel_1 = new GridBagConstraints();

gbc_panel_1.insets = new Insets(5, 5, 5, 5);

gbc_panel_1.fill = GridBagConstraints.BOTH;

gbc_panel_1.weightx = 1.0;

gbc_panel_1.weighty = 1.0;

gbc_panel_1.gridx = 0;

gbc_panel_1.gridy = 0;

panel.add(panel_1, gbc_panel_1);

TitledBorder phosphoTitle = new TitledBorder("Hydrophilicity");
```

```
			panel_1.setBorder(phosphoTitle);
			GridBagLayout gbl_panel_1 = new GridBagLayout();
			gbl_panel_1.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
			gbl_panel_1.rowHeights = new int[]{0, 0};
			gbl_panel_1.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
			gbl_panel_1.rowWeights = new double[]{0.0, Double.MIN_VALUE};
			panel_1.setLayout(gbl_panel_1);

			hydrophilicityTextField = new JTextField();
			GridBagConstraints gbc_textField = new GridBagConstraints();
			gbc_textField.insets = new Insets(0, 0, 5, 5);
			gbc_textField.fill = GridBagConstraints.BOTH;
			gbc_textField.gridx = 1;
			gbc_textField.gridy = 0;
			panel_1.add(hydrophilicityTextField, gbc_textField);
			hydrophilicityTextField.setColumns(10);

			hydrophilicityCheckBox = new JCheckBox("");
			GridBagConstraints gbc_checkBox = new GridBagConstraints();
			gbc_checkBox.insets = new Insets(0, 0, 5, 0);
			gbc_checkBox.fill = GridBagConstraints.BOTH;
			gbc_checkBox.gridx = 4;
```

```
        gbc_checkBox.gridy = 0;
        panel_1.add(hydrophilicityCheckBox, gbc_checkBox);


        JPanel panel_2 = new JPanel();
        GridBagConstraints gbc_panel_2 = new GridBagConstraints();
        gbc_panel_2.insets = new Insets(5, 5, 5, 5);
        gbc_panel_2.fill = GridBagConstraints.BOTH;
        gbc_panel_2.weightx = 1.0;
        gbc_panel_2.weighty = 1.0;
        gbc_panel_2.gridx = 0;
        gbc_panel_2.gridy = 1;
        panel.add(panel_2, gbc_panel_2);
        TitledBorder acetylationTitle = new TitledBorder("Accessibility");
        panel_2.setBorder(acetylationTitle);
        GridBagLayout gbl_panel_2 = new GridBagLayout();
        gbl_panel_2.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
        gbl_panel_2.rowHeights = new int[]{0, 0};
        gbl_panel_2.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
        gbl_panel_2.rowWeights = new double[]{0.0, Double.MIN_VALUE};
        panel_2.setLayout(gbl_panel_2);


        accessibilityTextField = new JTextField();
        GridBagConstraints gbc_textField_2 = new GridBagConstraints();
        gbc_textField_2.insets = new Insets(0, 0, 0, 5);
        gbc_textField_2.fill = GridBagConstraints.HORIZONTAL;
        gbc_textField_2.gridx = 1;
        gbc_textField_2.gridy = 0;
        panel_2.add(accessibilityTextField, gbc_textField_2);
        accessibilityTextField.setColumns(10);

        accessibilityCheckBox = new JCheckBox();
        GridBagConstraints gbc_chckbxNewCheckBox = new
GridBagConstraints();
        gbc_chckbxNewCheckBox.gridx = 4;
        gbc_chckbxNewCheckBox.gridy = 0;
        panel_2.add(accessibilityCheckBox, gbc_chckbxNewCheckBox);

        JPanel panel_3 = new JPanel();
        GridBagConstraints gbc_panel_3 = new GridBagConstraints();
        gbc_panel_3.insets = new Insets(5, 5, 5, 5);
        gbc_panel_3.fill = GridBagConstraints.BOTH;
        gbc_panel_3.weightx = 1.0;
        gbc_panel_3.weighty = 1.0;
        gbc_panel_3.gridx = 0;
        gbc_panel_3.gridy = 2;
```

```
		panel.add(panel_3, gbc_panel_3);
		TitledBorder methylationTitle = new TitledBorder("Exposed Surface");
		panel_3.setBorder(methylationTitle);
		GridBagLayout gbl_panel_3 = new GridBagLayout();
		gbl_panel_3.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
		gbl_panel_3.rowHeights = new int[]{0, 0};
		gbl_panel_3.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
		gbl_panel_3.rowWeights = new double[]{0.0, Double.MIN_VALUE};
		panel_3.setLayout(gbl_panel_3);

		exposedSurfaceTextField = new JTextField();
		GridBagConstraints gbc_textField_4 = new GridBagConstraints();
		gbc_textField_4.insets = new Insets(0, 0, 0, 5);
		gbc_textField_4.fill = GridBagConstraints.HORIZONTAL;
		gbc_textField_4.gridx = 1;
		gbc_textField_4.gridy = 0;
		panel_3.add(exposedSurfaceTextField, gbc_textField_4);
		exposedSurfaceTextField.setColumns(10);

		exposedSurfaceCheckBox = new JCheckBox("");
		GridBagConstraints gbc_checkBox_1 = new GridBagConstraints();
		gbc_checkBox_1.gridx = 4;
		gbc_checkBox_1.gridy = 0;
		panel_3.add(exposedSurfaceCheckBox, gbc_checkBox_1);


		JPanel panel_4 = new JPanel();
		GridBagConstraints gbc_panel_4 = new GridBagConstraints();
		gbc_panel_4.insets = new Insets(5, 5, 5, 5);
		gbc_panel_4.fill = GridBagConstraints.BOTH;
		gbc_panel_4.weightx = 1.0;
		gbc_panel_4.weighty = 1.0;
		gbc_panel_4.gridx = 0;
		gbc_panel_4.gridy = 3;
		panel.add(panel_4, gbc_panel_4);
		TitledBorder oGalnacTitle = new TitledBorder("Antigenic Propensity");
		panel_4.setBorder(oGalnacTitle);
		GridBagLayout gbl_panel_4 = new GridBagLayout();
		gbl_panel_4.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
		gbl_panel_4.rowHeights = new int[]{0, 0};
		gbl_panel_4.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
		gbl_panel_4.rowWeights = new double[]{0.0, Double.MIN_VALUE};
		panel_4.setLayout(gbl_panel_4);


		antegenicPropensityTextField = new JTextField();
		GridBagConstraints gbc_textField_6 = new GridBagConstraints();
		gbc_textField_6.insets = new Insets(0, 0, 0, 5);
		gbc_textField_6.fill = GridBagConstraints.HORIZONTAL;
		gbc_textField_6.gridx = 1;
		gbc_textField_6.gridy = 0;
		panel_4.add(antegenicPropensityTextField, gbc_textField_6);
		antegenicPropensityTextField.setColumns(10);

		antegenicPropensityCheckBox = new JCheckBox();
		GridBagConstraints gbc_chckbxNewCheckBox_1 = new
GridBagConstraints();
		gbc_chckbxNewCheckBox_1.gridx = 4;
		gbc_chckbxNewCheckBox_1.gridy = 0;
		panel_4.add(antegenicPropensityCheckBox,
gbc_chckbxNewCheckBox_1);

		JPanel panel_5 = new JPanel();
		GridBagConstraints gbc_panel_5 = new GridBagConstraints();
		gbc_panel_5.insets = new Insets(5, 5, 5, 0);
		gbc_panel_5.fill = GridBagConstraints.BOTH;
		gbc_panel_5.weightx = 1.0;
		gbc_panel_5.weighty = 1.0;
		gbc_panel_5.gridx = 1;
```

```
                gbc_panel_5.gridy = 0;

                panel.add(panel_5, gbc_panel_5);

                TitledBorder oGlcnacTitle = new TitledBorder("Flexibility");

                panel_5.setBorder(oGlcnacTitle);

                GridBagLayout gbl_panel_5 = new GridBagLayout();

                gbl_panel_5.columnWidths = new int[]{0, 0, 0, 0, 0, 0};

                gbl_panel_5.rowHeights = new int[]{0, 0};

                gbl_panel_5.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,

Double.MIN_VALUE};

                gbl_panel_5.rowWeights = new double[]{0.0, Double.MIN_VALUE};

                panel_5.setLayout(gbl_panel_5);


                flexibilityTextField = new JTextField();

                GridBagConstraints gbc_textField_8 = new GridBagConstraints();

                gbc_textField_8.insets = new Insets(0, 0, 0, 5);

                gbc_textField_8.fill = GridBagConstraints.HORIZONTAL;

                gbc_textField_8.gridx = 1;

                gbc_textField_8.gridy = 0;

                panel_5.add(flexibilityTextField, gbc_textField_8);

                flexibilityTextField.setColumns(10);


                flexibilityCheckBox = new JCheckBox("");

                GridBagConstraints gbc_checkBox_2 = new GridBagConstraints();
                gbc_checkBox_2.gridx = 4;

                gbc_checkBox_2.gridy = 0;

                panel_5.add(flexibilityCheckBox, gbc_checkBox_2);


                JPanel panel_6 = new JPanel();

                GridBagConstraints gbc_panel_6 = new GridBagConstraints();

                gbc_panel_6.insets = new Insets(5, 5, 5, 0);

                gbc_panel_6.fill = GridBagConstraints.BOTH;

                gbc_panel_6.weightx = 1.0;

                gbc_panel_6.weighty = 1.0;

                gbc_panel_6.gridx = 1;

                gbc_panel_6.gridy = 1;

                panel.add(panel_6, gbc_panel_6);

                TitledBorder sumoylationTitle = new TitledBorder("Turns");

                panel_6.setBorder(sumoylationTitle);

                GridBagLayout gbl_panel_6 = new GridBagLayout();

                gbl_panel_6.columnWidths = new int[]{0, 0, 0, 0, 0, 0};

                gbl_panel_6.rowHeights = new int[]{0, 0};

                gbl_panel_6.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,

Double.MIN_VALUE};

                gbl_panel_6.rowWeights = new double[]{0.0, Double.MIN_VALUE};

                panel_6.setLayout(gbl_panel_6);
```

```
turnsTextField = new JTextField();

GridBagConstraints gbc_textField_10 = new GridBagConstraints();

gbc_textField_10.insets = new Insets(0, 0, 0, 5);

gbc_textField_10.fill = GridBagConstraints.HORIZONTAL;

gbc_textField_10.gridx = 1;

gbc_textField_10.gridy = 0;

panel_6.add(turnsTextField, gbc_textField_10);

turnsTextField.setColumns(10);


turnsCheckBox = new JCheckBox("");

GridBagConstraints gbc_checkBox_3 = new GridBagConstraints();

gbc_checkBox_3.gridx = 4;

gbc_checkBox_3.gridy = 0;

panel_6.add(turnsCheckBox, gbc_checkBox_3);


JPanel panel_7 = new JPanel();

GridBagConstraints gbc_panel_7 = new GridBagConstraints();

gbc_panel_7.insets = new Insets(5, 5, 5, 0);

gbc_panel_7.fill = GridBagConstraints.BOTH;

gbc_panel_7.weightx = 1.0;

gbc_panel_7.weighty = 1.0;

gbc_panel_7.gridx = 1;

gbc_panel_7.gridy = 2;
```

```
                    panel.add(panel_7, gbc_panel_7);
                    TitledBorder ubiquitinationTitle = new TitledBorder("Polarity");
                    panel_7.setBorder(ubiquitinationTitle);
                    GridBagLayout gbl_panel_7 = new GridBagLayout();
                    gbl_panel_7.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
                    gbl_panel_7.rowHeights = new int[]{0, 0};
                    gbl_panel_7.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
                    gbl_panel_7.rowWeights = new double[]{0.0, Double.MIN_VALUE};
                    panel_7.setLayout(gbl_panel_7);

                    polarityTextField = new JTextField();
                    GridBagConstraints gbc_textField_12 = new GridBagConstraints();
                    gbc_textField_12.insets = new Insets(0, 0, 0, 5);
                    gbc_textField_12.fill = GridBagConstraints.HORIZONTAL;
                    gbc_textField_12.gridx = 1;
                    gbc_textField_12.gridy = 0;
                    panel_7.add(polarityTextField, gbc_textField_12);
                    polarityTextField.setColumns(10);

                    polarityCheckBox = new JCheckBox("");
                    GridBagConstraints gbc_checkBox_4 = new GridBagConstraints();
                    gbc_checkBox_4.gridx = 4;
```

```
		gbc_checkBox_4.gridy = 0;
		panel_7.add(polarityCheckBox, gbc_checkBox_4);


		JPanel panel_8 = new JPanel();
		GridBagConstraints gbc_panel_8 = new GridBagConstraints();
		gbc_panel_8.insets = new Insets(5, 5, 5, 0);
		gbc_panel_8.fill = GridBagConstraints.BOTH;
		gbc_panel_8.weightx = 1.0;
		gbc_panel_8.weighty = 1.0;
		gbc_panel_8.gridx = 1;
		gbc_panel_8.gridy = 3;
		panel.add(panel_8, gbc_panel_8);
		panel_8.setBorder(new TitledBorder("Combined"));
		GridBagLayout gbl_panel_8 = new GridBagLayout();
		gbl_panel_8.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
		gbl_panel_8.rowHeights = new int[]{0, 0};
		gbl_panel_8.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
		gbl_panel_8.rowWeights = new double[]{0.0, Double.MIN_VALUE};
		panel_8.setLayout(gbl_panel_8);


		combinedTextField = new JTextField();
		GridBagConstraints gbc_textField_13 = new GridBagConstraints();
```

```
gbc_textField_13.insets = new Insets(0, 0, 0, 5);
gbc_textField_13.fill = GridBagConstraints.HORIZONTAL;
gbc_textField_13.gridx = 1;
gbc_textField_13.gridy = 0;
panel_8.add(combinedTextField, gbc_textField_13);
combinedTextField.setColumns(10);
combinedTextField.setEnabled(false);

combinedCheckBox = new JCheckBox("");
GridBagConstraints gbc_checkBox_5 = new GridBagConstraints();
gbc_checkBox_5.gridx = 4;
gbc_checkBox_5.gridy = 0;
panel_8.add(combinedCheckBox, gbc_checkBox_5);
combinedCheckBox.setEnabled(false);

chckbxSelectAll = new JCheckBox("Select All");
GridBagConstraints gbc_chckbxSelectAll = new GridBagConstraints();
gbc_chckbxSelectAll.insets = new Insets(0, 0, 5, 5);
gbc_chckbxSelectAll.gridx = 0;
gbc_chckbxSelectAll.gridy = 4;
gbc_chckbxSelectAll.gridwidth = 2;
panel.add(chckbxSelectAll, gbc_chckbxSelectAll);
```

```
		JPanel panel_9 = new JPanel();
		GridBagConstraints gbc_panel_9 = new GridBagConstraints();
		gbc_panel_9.insets = new Insets(0, 0, 5, 5);
		gbc_panel_9.fill = GridBagConstraints.BOTH;
		gbc_panel_9.gridx = 0;
		gbc_panel_9.gridy = 5;
		gbc_panel_9.gridwidth = 2;
		panel.add(panel_9, gbc_panel_9);

		lblNumbersMustBe = new JLabel("Numbers must be between -3 and 3. At
least two checkboxes must be selected in order to select \"Combined\".");
		lblNumbersMustBe.setFont(new Font("Lucida Grande", Font.PLAIN,
10));
		panel_9.add(lblNumbersMustBe);

		btnSubmit = new JButton("Submit");
		GridBagConstraints gbc_btnSubmit = new GridBagConstraints();
		gbc_btnSubmit.insets = new Insets(0, 0, 5, 5);
		gbc_btnSubmit.gridx = 0;
		gbc_btnSubmit.gridy = 6;
		gbc_btnSubmit.gridwidth = 2;
		panel.add(btnSubmit, gbc_btnSubmit);
		btnSubmit.setEnabled(false);
		hydrophilicityTextField.setText("2");
		accessibilityTextField.setText("2");
		exposedSurfaceTextField.setText("2.4");
		antegenicPropensityTextField.setText("1.8");
		flexibilityTextField.setText("1.9");
		turnsTextField.setText("1.9");
		polarityTextField.setText("2.3");
		combinedTextField.setText("1.9");

		setVisible(true);
	}

	@Override
	public JTextField hydrophilicityTextField() {
		// TODO Auto-generated method stub
		return hydrophilicityTextField;
	}

	@Override
	public JTextField accessibilityTextField() {
		// TODO Auto-generated method stub
		return accessibilityTextField;
```

```
}

@Override
public JTextField exposedSurfaceTextField() {
        // TODO Auto-generated method stub
        return exposedSurfaceTextField;
}

@Override
public JTextField antegenicPropensityTextField() {
        // TODO Auto-generated method stub
        return antegenicPropensityTextField;
}

@Override
public JTextField flexibilityTextField() {
        // TODO Auto-generated method stub
        return flexibilityTextField;
}

@Override
public JTextField turnsTextField() {
        // TODO Auto-generated method stub
```

```
		return turnsTextField;
	}

	@Override
	public JTextField polarityTextField() {
		// TODO Auto-generated method stub
		return polarityTextField;
	}

	@Override
	public JCheckBox hydrophilicityCheckBox() {
		// TODO Auto-generated method stub
		return hydrophilicityCheckBox;
	}

	@Override
	public JCheckBox accessibilityCheckBox() {
		// TODO Auto-generated method stub
		return accessibilityCheckBox;
	}

	@Override
	public JCheckBox exposedSurfaceCheckBox() {
        // TODO Auto-generated method stub
        return exposedSurfaceCheckBox;
}

@Override
public JCheckBox antegenicPropensityCheckBox() {
        // TODO Auto-generated method stub
        return antegenicPropensityCheckBox;
}

@Override
public JCheckBox flexibilityCheckBox() {
        // TODO Auto-generated method stub
        return flexibilityCheckBox;
}

@Override
public JCheckBox turnsCheckBox() {
        // TODO Auto-generated method stub
        return turnsCheckBox;
}

@Override
```

```
public JCheckBox polarityCheckBox() {
	// TODO Auto-generated method stub
	return polarityCheckBox;
}

@Override
public JButton btnSubmit() {
	// TODO Auto-generated method stub
	return btnSubmit;
}

@Override
public JCheckBox chckbxSelectAll() {
	// TODO Auto-generated method stub
	return chckbxSelectAll;
}

@Override
public JTextField combinedTextField() {
	// TODO Auto-generated method stub
	return combinedTextField;
}
```

```
	@Override
	public JCheckBox combinedCheckBox() {
		// TODO Auto-generated method stub
		return combinedCheckBox;
	}
}
package epiprot.services.epitopePrediction;

public class BepipredPredictionService extends IedbEpitopePredictionService {
	public BepipredPredictionService(String sequence) {
		// TODO Auto-generated constructor stub
		super(sequence,"Bepipred");
	}
}
package epiprot.services.blast;

import static
org.biojava.nbio.ws.alignment.qblast.BlastAlignmentParameterEnum.ENTREZ_QUERY
;
import java.io.*;
import org.biojava.nbio.core.sequence.io.util.IOUtils;
import org.biojava.nbio.ws.alignment.qblast.*;
```

```
public class BioJavaBlastService {

	private static final String BLAST_OUTPUT_FILE = "blastOutput2.xml";    // file
to save blast results to
	private static final String SEQUENCE = "TSDDRGIVYSKSLD";     // Blast
query sequence

	public static void main(String[] args) {
		NCBIQBlastService service = new NCBIQBlastService();

		// set alignment options
		NCBIQBlastAlignmentProperties props = new
NCBIQBlastAlignmentProperties();
		props.setBlastProgram(BlastProgramEnum.blastp);
		props.setBlastDatabase("swissprot");
		props.setAlignmentOption(ENTREZ_QUERY, "human[Organism]");
		props.setBlastExpect(200000);
		props.setBlastWordSize(2);
		props.setBlastGapCosts(9, 1);
		props.setBlastMatrix(BlastMatrixEnum.PAM30);

	props.setAlignmentOption(BlastAlignmentParameterEnum.HITLIST_SIZE,
"100");
```

```
		props.setAlignmentOption(BlastAlignmentParameterEnum.THRESHOLD, "11");



			// set output options
			NCBIQBlastOutputProperties outputProps = new
NCBIQBlastOutputProperties();
			// in this example we use default values set by constructor (XML format,
pairwise alignment, 100 descriptions and alignments)

			// Example of two possible ways of setting output options
//			outputProps.setAlignmentNumber(200);
//
		outputProps.setOutputOption(BlastOutputParameterEnum.ALIGNMENTS,
"200");

			String rid = null;		// blast request ID
			FileWriter writer = null;
			BufferedReader reader = null;
			try {
				// send blast request and save request id
				rid = service.sendAlignmentRequest(SEQUENCE, props);
```

```
                        // wait until results become available. Alternatively, one can do
other computations/send other alignment requests
                        while (!service.isReady(rid)) {
                                System.out.println("Waiting for results. Sleeping for 5
seconds");
                                Thread.sleep(5000);
                        }

                        // read results when they are ready
                        InputStream in = service.getAlignmentResults(rid, outputProps);
                        reader = new BufferedReader(new InputStreamReader(in));

                        // write blast output to specified file
                        File f = new File(BLAST_OUTPUT_FILE);
                        System.out.println("Saving query results in file " +
f.getAbsolutePath());
                        writer = new FileWriter(f);

                        String line;
                        while ((line = reader.readLine()) != null) {
                                writer.write(line + System.getProperty("line.separator"));
                        }
                } catch (Exception e) {
```

```
				System.out.println(e.getMessage());
				e.printStackTrace();
			} finally {
				// clean up
				IOUtils.close(writer);
				IOUtils.close(reader);

				// delete given alignment results from blast server (optional
operation)
				service.sendDeleteRequest(rid);
			}
		}

}
package epiprot.services.views;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JComboBox;
import javax.swing.JTextField;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;


import epiprot.Presenter;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.DatabaseOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.DropOffOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.ExpectationOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.FilterOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.MatrixOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.ScoreCutoffOption;


public class BlastPresenter {


        Presenter presenter;


        interface View {

                JComboBox databaseOptionComboBox();

                JComboBox expectationOptionComboBox();

                JComboBox filterOptionComboBox();

                JComboBox dropOffOptionComboBox();

                JComboBox isGapAlignComboBox();

                JTextField gapExtTextField();

                JTextField gapOpenTextField();

                JComboBox matrixOptionComboBox();
```

```
		//JComboBox alignmentCutoffOption = new JComboBox();
		JComboBox scoreCutoffOptionComboBox();
		JCheckBox chckbxLimitToProteins();
		JCheckBox chckbxLimitToSwissProt();


		JButton submitButton();


	}


	View view = new BlastView();


	public BlastPresenter(Presenter presenter) {
		// TODO Auto-generated constructor stub
		this.presenter= presenter;
		bindHandlers();
	}


	void bindHandlers() {
		view.submitButton().addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				// TODO Auto-generated method stub
				//System.out.println("BlastPresenter bindHandlers");
```

```
                        DatabaseOption databaseOption = (DatabaseOption)
view.databaseOptionComboBox().getSelectedItem();
                        System.out.println(databaseOption.toString());

                        ExpectationOption expectationOption =
(ExpectationOption) view.expectationOptionComboBox().getSelectedItem();
                        FilterOption filterOption = (FilterOption)
view.filterOptionComboBox().getSelectedItem();
                        DropOffOption dropOffOption = (DropOffOption)
view.dropOffOptionComboBox().getSelectedItem();
                    MatrixOption matrixOption = (MatrixOption)
view.matrixOptionComboBox().getSelectedItem();
                        ScoreCutoffOption scoreCutoffOption =
(ScoreCutoffOption) view.scoreCutoffOptionComboBox().getSelectedItem();

                        boolean isGapAlign = (boolean)
view.isGapAlignComboBox().getSelectedItem();
                    int gapExt = Integer.parseInt(view.gapExtTextField().getText());
                    int gapOpen =
Integer.parseInt(view.gapOpenTextField().getText());

                        boolean limitToTargetSpecies =
view.chckbxLimitToProteins().isSelected();
```

```
				boolean limitToSwissProtDB =
view.chckbxLimitToSwissProt().isSelected();


				presenter.setBlastHits(databaseOption, expectationOption,
filterOption, dropOffOption, matrixOption, scoreCutoffOption, isGapAlign, gapExt,
gapOpen, limitToTargetSpecies, limitToSwissProtDB);

			}
		});

		view.gapExtTextField().getDocument().addDocumentListener(new
DocumentListener() {
			@Override
			public void insertUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
			@Override
			public void removeUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
```

```
                        @Override
                        public void changedUpdate(DocumentEvent e) {
                                // TODO Auto-generated method stub
                                checkTextField();
                        }
                        public void checkTextField() {
                                String text = view.gapExtTextField().getText();
                                if(isInteger(text) && (Integer.parseInt(text) == -1 ||
(Integer.parseInt(text) > 0 && Integer.parseInt(text) < 5))) {
                                        view.submitButton().setEnabled(true);
                                }
                                else {
                                        view.submitButton().setEnabled(false);
                                }
                        }
                });

                view.gapOpenTextField().getDocument().addDocumentListener(new
DocumentListener() {
                        @Override
                        public void insertUpdate(DocumentEvent e) {
                                // TODO Auto-generated method stub
                                checkTextField();
```

```
					}
					@Override
					public void removeUpdate(DocumentEvent e) {
						// TODO Auto-generated method stub
						checkTextField();
					}
					@Override
					public void changedUpdate(DocumentEvent e) {
						// TODO Auto-generated method stub
						checkTextField();
					}
					public void checkTextField() {
						String text = view.gapOpenTextField().getText();
						if(isInteger(text) && (Integer.parseInt(text) == -1 ||
(Integer.parseInt(text) > 0 && Integer.parseInt(text) < 5))) {
							view.submitButton().setEnabled(true);
						}
						else {
							view.submitButton().setEnabled(false);
						}
					}
				});
			}
   private DatabaseOption getDatabaseOption(String db) {
            switch(db){
case "UniProtKB/Swiss-Prot" :
   return DatabaseOption.SWISSPROT;
case "UniProtKB" :
   return DatabaseOption.UNIPROTKB;
case "Archaea" :
   return DatabaseOption.UNIPROT_ARCHAEA;
case "Bacteria" :
   return DatabaseOption.UNIPROT_BACTERIA;
case "Eukaryota" :
   return DatabaseOption.UNIPROT_EUKARYOTA;
case "Arthropoda" :
   return DatabaseOption.UNIPROT_ARTHROPODA;
case "Fungi" :
   return DatabaseOption.UNIPROT_FUNGI;
case "Human" :
   return DatabaseOption.UNIPROT_HUMAN;
case "Mammals" :
   return DatabaseOption.UNIPROT_MAMMALS;
case "Plants" :
   return DatabaseOption.UNIPROT_VIRIDIPLANTAE;
case "Rodents" :
   return DatabaseOption.UNIPROT_RODENTS;
case "Vertebrates" :
   return DatabaseOption.UNIPROT_VERTEBRATES;
case "Viruses" :
   return DatabaseOption.UNIPROT_VIRUSES;
case "with 3D structure (PDB)" :
   return DatabaseOption.UNIPROT_PDB;
case "Microbial proteomes" :
   return DatabaseOption.UNIPROT_COMPLETE_MICROBIAL_PROTEOMES;
            case "UniRef100" :
         return DatabaseOption.UNIREF_100;
            case "UniRef90" :
               return DatabaseOption.UNIREF_90;
            case "UniRef50" :
               return DatabaseOption.UNIREF_50;
            case "UniParc" :
               return DatabaseOption.UNIPARC;
            case "Trembl" :
               return DatabaseOption.TREMBL;
            }
            return null;
   }
```

```
        private boolean isInteger(String s) {
                try {
            Integer.parseInt(s);
          } catch(NumberFormatException e) {
            return false;
          } catch(NullPointerException e) {
            return false;
          }
          // only got here if we didn't return false
          return true;
        }

        public static void main (String[] args) {
                System.out.println(DatabaseOption.SWISSPROT);
        }

}
package epiprot.services.views;

import javax.swing.JFrame;
import java.awt.Dimension;
import javax.swing.JComboBox;

import java.awt.BorderLayout;

import javax.swing.JLabel;

import javax.swing.JPanel;

import javax.swing.JScrollPane;

import javax.swing.JTextField;


import com.jgoodies.forms.layout.FormLayout;

import com.jgoodies.forms.layout.ColumnSpec;

import com.jgoodies.forms.layout.FormSpecs;

import com.jgoodies.forms.layout.RowSpec;


import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.DatabaseOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.DropOffOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.ExpectationOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.FilterOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.MatrixOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.ScoreCutoffOption;


import javax.swing.JButton;

import javax.swing.JCheckBox;


public class BlastView extends JFrame implements BlastPresenter.View{
	JComboBox databaseOptionComboBox = new JComboBox();
	JComboBox expectationOptionComboBox = new JComboBox();
	JComboBox filterOptionComboBox = new JComboBox();
	JComboBox dropOffOptionComboBox = new JComboBox();
	JComboBox isGapAlignComboBox = new JComboBox();
	JTextField gapExtTextField = new JTextField("-1");
	JTextField gapOpenTextField = new JTextField("-1");
	JComboBox matrixOptionComboBox = new JComboBox();
	//JComboBox alignmentCutoffOption = new JComboBox();
	JComboBox scoreCutoffOptionComboBox = new JComboBox();
	JCheckBox chckbxLimitToProteins = new JCheckBox("Limit to proteins of the
target protein species");
	JCheckBox chckbxLimitToSwissProt = new JCheckBox("Limit to proteins from
Swiss-Prot Database");
	JButton submitButton = new JButton("Submit");

	public BlastView() {
		getContentPane().setPreferredSize(new Dimension(400, 500));
		setTitle("BLAST");
		setPreferredSize(new Dimension(700, 500));
		pack();
		//setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
```

```
getContentPane().setLayout(new BorderLayout());


JScrollPane scrollPane = new JScrollPane();

getContentPane().add(scrollPane, BorderLayout.CENTER);


JPanel panel = new JPanel();

scrollPane.setViewportView(panel);

panel.setLayout(new FormLayout(new ColumnSpec[] {

                FormSpecs.RELATED_GAP_COLSPEC,

                ColumnSpec.decode("max(75dlu;default):grow"),

                FormSpecs.RELATED_GAP_COLSPEC,

                ColumnSpec.decode("max(75dlu;default):grow"),

                FormSpecs.RELATED_GAP_COLSPEC,},

        new RowSpec[] {

                FormSpecs.RELATED_GAP_ROWSPEC,

                FormSpecs.DEFAULT_ROWSPEC,

                FormSpecs.RELATED_GAP_ROWSPEC,

                FormSpecs.DEFAULT_ROWSPEC,

                FormSpecs.RELATED_GAP_ROWSPEC,

                FormSpecs.DEFAULT_ROWSPEC,

                FormSpecs.RELATED_GAP_ROWSPEC,

                FormSpecs.DEFAULT_ROWSPEC,

                FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,
FormSpecs.RELATED_GAP_ROWSPEC,
FormSpecs.DEFAULT_ROWSPEC,}));
```

```
                JLabel databaseOptionLabel = new JLabel("Target Database");
                panel.add(databaseOptionLabel, "2, 2");

                databaseOptionComboBox.addItem(DatabaseOption.UNIPROTKB);

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_ARCHAEA);

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_BACTERIA);

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_EUKARYOTA)
;

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_ARTHROPOD
A);
                databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_FUNGI);

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_HUMAN);

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_MAMMALS);

        databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_VIRIDIPLANT
AE);
```

```
	databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_RODENTS);

	databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_VERTEBRATE
S);

	databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_VIRUSES);
		databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_PDB);
		//databaseOptionComboBox.addItem("Reference proteomes");

	databaseOptionComboBox.addItem(DatabaseOption.UNIPROT_COMPLETE_M
ICROBIAL_PROTEOMES);
		databaseOptionComboBox.addItem(DatabaseOption.SWISSPROT);
		databaseOptionComboBox.addItem(DatabaseOption.UNIREF_100);
		databaseOptionComboBox.addItem(DatabaseOption.UNIREF_90);
		databaseOptionComboBox.addItem(DatabaseOption.UNIREF_50);
		databaseOptionComboBox.addItem(DatabaseOption.UNIPARC);
		databaseOptionComboBox.addItem(DatabaseOption.TREMBL);

	databaseOptionComboBox.setSelectedItem(DatabaseOption.SWISSPROT);
		panel.add(databaseOptionComboBox, "2, 4, fill, default");

		JLabel expectationOptionLabel = new JLabel("E-Threshold");
		panel.add(expectationOptionLabel, "4, 2");
```

```
        expectationOptionComboBox.addItem(ExpectationOption.ONE_EXPONENT_M
INUS_ONE_HUNDRED);

        expectationOptionComboBox.addItem(ExpectationOption.ONE_EXPONENT_M
INUS_TEN);

        expectationOptionComboBox.addItem(ExpectationOption.ONE_EXPONENT_M
INUS_ONE);
                expectationOptionComboBox.addItem(ExpectationOption.ONE);
                expectationOptionComboBox.addItem(ExpectationOption.TEN);

        expectationOptionComboBox.addItem(ExpectationOption.ONE_HUNDRED);

        expectationOptionComboBox.addItem(ExpectationOption.ONE_THOUSAND);
                expectationOptionComboBox.setSelectedItem(ExpectationOption.TEN);
                panel.add(expectationOptionComboBox, "4, 4, fill, default");

                JLabel matrixLabel = new JLabel("Matrix");
                panel.add(matrixLabel, "2, 6");
                matrixOptionComboBox.addItem(MatrixOption.BLOSUM_45);
                matrixOptionComboBox.addItem(MatrixOption.BLOSUM_62);
                matrixOptionComboBox.addItem(MatrixOption.BLOSUM_80);
                matrixOptionComboBox.addItem(MatrixOption.PAM_70);
                matrixOptionComboBox.addItem(MatrixOption.PAM_30);
                matrixOptionComboBox.setSelectedItem(MatrixOption.BLOSUM_62);
                panel.add(matrixOptionComboBox, "2, 8, fill, default");

                JLabel isGapAlignLabel = new JLabel("Gapped");
                panel.add(isGapAlignLabel, "4, 6");
                isGapAlignComboBox.addItem(true);
                isGapAlignComboBox.addItem(false);
                isGapAlignComboBox.setSelectedItem(true);
                panel.add(isGapAlignComboBox, "4, 8, fill, default");

                JLabel scoreCutoffeOptionsLabel = new JLabel("Number of hits");
                panel.add(scoreCutoffeOptionsLabel, "2, 10");

                scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.FIVE);
                scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.TEN);
                scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.TWENTY);
                scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.FIFTY);

        scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.ONE_HUNDRED_FI
FTY);
```

```
        scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.TWO_HUNDRED);

        scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.TWO_HUNDRED_FI
FTY);

        scoreCutoffOptionComboBox.addItem(ScoreCutoffOption.FIVE_HUNDRED);

        scoreCutoffOptionComboBox.setSelectedItem(ScoreCutoffOption.TWO_HUND
RED_FIFTY);
                panel.add(scoreCutoffOptionComboBox, "2, 12, fill, default");

                JLabel filterOptionLabel = new JLabel("Filter Options");
                panel.add(filterOptionLabel, "4, 10");

                filterOptionComboBox.addItem(FilterOption.YES);
                filterOptionComboBox.addItem(FilterOption.NO);
                filterOptionComboBox.setSelectedItem(FilterOption.NO);
                panel.add(filterOptionComboBox, "4, 12, fill, default");

                JLabel dropOffOptionLabel = new JLabel("Drop Off Option");
                panel.add(dropOffOptionLabel, "2, 14");
```

```
		dropOffOptionComboBox.addItem(DropOffOption.ZERO);
		dropOffOptionComboBox.addItem(DropOffOption.TWO);
		dropOffOptionComboBox.addItem(DropOffOption.FOUR);
		dropOffOptionComboBox.addItem(DropOffOption.SIX);
		dropOffOptionComboBox.addItem(DropOffOption.EIGHT);
		dropOffOptionComboBox.addItem(DropOffOption.TEN);
		dropOffOptionComboBox.setSelectedItem(DropOffOption.ZERO);
		panel.add(dropOffOptionComboBox, "2, 16, fill, default");

		JLabel gapOpenLabel = new JLabel("Gap Open. Default = -1, pick # 1-
4");
		panel.add(gapOpenLabel, "2, 18");
		panel.add(gapOpenTextField, "2, 20, fill, default");

		JLabel gapExtLabel = new JLabel("Gap Extend. Default = -1, pick # 1-
4");
		panel.add(gapExtLabel, "4, 18");
		panel.add(gapExtTextField, "4, 20, fill, default");

		panel.add(chckbxLimitToProteins, "2, 26");

		panel.add(chckbxLimitToSwissProt, "2, 28");
```

```
        panel.add(submitButton, "2, 30, 3, 1");

        JLabel lblNewLabel = new JLabel("");
        panel.add(lblNewLabel, "2, 48");

        JLabel lblNewLabel_1 = new JLabel("");
        panel.add(lblNewLabel_1, "2, 50");

        setVisible(true);
}



public static void main (String[]args) {
        BlastView bv = new BlastView();
}

@Override
public JCheckBox chckbxLimitToProteins() {
        // TODO Auto-generated method stub
        return chckbxLimitToProteins;
}
```

```
@Override
public JCheckBox chckbxLimitToSwissProt() {
	// TODO Auto-generated method stub
	return chckbxLimitToSwissProt;
}



@Override
public JComboBox databaseOptionComboBox() {
	// TODO Auto-generated method stub
	return databaseOptionComboBox;
}



@Override
public JComboBox expectationOptionComboBox() {
	// TODO Auto-generated method stub
	return expectationOptionComboBox;
}
```

```
@Override
public JComboBox filterOptionComboBox() {
        // TODO Auto-generated method stub
        return filterOptionComboBox;
}



@Override
public JComboBox dropOffOptionComboBox() {
        // TODO Auto-generated method stub
        return dropOffOptionComboBox;
}



@Override
public JComboBox isGapAlignComboBox() {
        // TODO Auto-generated method stub
        return isGapAlignComboBox;
}
```

```
@Override
public JTextField gapExtTextField() {
        // TODO Auto-generated method stub
        return gapExtTextField;
}



@Override
public JTextField gapOpenTextField() {
        // TODO Auto-generated method stub
        return gapOpenTextField;
}



@Override
public JComboBox matrixOptionComboBox() {
        // TODO Auto-generated method stub
        return matrixOptionComboBox;
}
```

```
	@Override
	public JComboBox scoreCutoffOptionComboBox() {
		// TODO Auto-generated method stub
		return scoreCutoffOptionComboBox;
	}



	@Override
	public JButton submitButton() {
		// TODO Auto-generated method stub
		return submitButton;
	}
}
package epiprot.services.epitopePrediction;

import epiprot.Protein;

public class ChouPredictionService extends IedbEpitopePredictionService {

	public ChouPredictionService(String sequence) {
		// TODO Auto-generated constructor stub
		super(sequence,"Chou-Fasman");
	}

	public static void main (String[] args) {
		Protein protein = new Protein("Q99523",true);
		ChouPredictionService cps = new
ChouPredictionService(protein.getSequence());
		cps.run();
		System.out.println(cps.getAminoAcids().size());
		for (IedbEpitopePredictionAminoAcid aa : cps.getAminoAcids()) {
			System.out.println(aa.getScore());
		}
	}
}
package epiprot.services.uniprot;

import java.util.ArrayList;

public class DBentry {
	private String id;
	private String type;
private ArrayList<DBproperty> dbPropertiesList;


public DBentry (String id, String type) {
        this.id = id;
        this.type = type;
}
public String getId() {
        return id;
}
public void setId(String id) {
        this.id = id;
}
public String getType() {
        return type;
}
public void setType(String type) {
        this.type = type;
}
public ArrayList<DBproperty> getDbPropertiesList() {
        return dbPropertiesList;
}
public void setDbPropertiesList(ArrayList<DBproperty> dbPropertiesList) {
        this.dbPropertiesList = dbPropertiesList;
        }
        public void addDbProperty(DBproperty dbProperty) {
                this.dbPropertiesList.add(dbProperty);
        }
}
package epiprot.services.uniprot;


public class DBproperty {
        private String type;
        private String value;


        public DBproperty (String type, String value) {
                this.type = type;
                this.value = value;
        }
        public String getValue() {
                return value;
        }
        public void setValue(String value) {
                this.value = value;
        }
        public String getType() {
                return type;
```

```
	}
	public void setType(String type) {
		this.type = type;
	}
}
package epiprot.services.epitopePrediction;

import epiprot.AminoAcid;

public class DiscotopePredictionAminoAcid extends AminoAcid{

	private String chain;
	private int residueNo;
	private int contactNo;
	private double propensityScore;
	private double score;
	private boolean isPredictedEpitope;

	public DiscotopePredictionAminoAcid() {
		// TODO Auto-generated constructor stub
	}

	public String getChain() {
		return chain;
	}

	public void setChain(String chain) {
		this.chain = chain;
	}

	public int getResidueNo() {
		return residueNo;
	}

	public void setResidueNo(int residueNo) {
		this.residueNo = residueNo;
	}

	public int getContactNo() {
		return contactNo;
	}

	public void setContactNo(int contactNo) {
		this.contactNo = contactNo;
	}
```

```
public double getPropensityScore() {

	return propensityScore;

}


public void setPropensityScore(double propensityScore) {

	this.propensityScore = propensityScore;

}


public double getScore() {

	return score;

}


public void setScore(double score) {

	this.score = score;

}


public boolean isPredictedEpitope() {

	return isPredictedEpitope;

}


public void setPredictedEpitope(boolean isPredictedEpitope) {

	this.isPredictedEpitope = isPredictedEpitope;

}
```

```
}
package epiprot.services.epitopePrediction;

import java.io.BufferedReader;
import java.io.File;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;

import org.apache.commons.io.FileUtils;
import epiprot.services.Service;

public class DiscotopePredictionService extends Service{

        private String pdbId;
        private String chain;

        private final String DISCOTOPE_PATH = "/Users/Patrick/Documents/discotope-
1.1/";
```

```
	private File dataFile;

	public DiscotopePredictionService(String pdbId, String chain) {
		// TODO Auto-generated constructor stub
		this.pdbId = pdbId;
		this.chain = chain;
	}

	@Override
	public void run() {
		try {
			URL url = new
URL("http://www.rcsb.org/pdb/files/"+pdbId+".pdb");
			dataFile = new File(SERVICEFILEPATH + pdbId + ".pdb");
			PrintWriter writer = new PrintWriter(dataFile);
			writer.close();
			FileUtils.copyURLToFile(url, dataFile);
		} catch (MalformedURLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
```

```
                        e.printStackTrace();
                }
        }

        public String getOutput() {
      String output = "";

                try {
        // using the Runtime exec method:
        String cmd = DISCOTOPE_PATH + "discotope -f " + dataFile + " -chain " +
chain;
        System.out.println(cmd);
        Process p = Runtime.getRuntime().exec(cmd);
        p.waitFor();
        BufferedReader stdInput = new BufferedReader(new
              InputStreamReader(p.getInputStream()));

        BufferedReader stdError = new BufferedReader(new
            InputStreamReader(p.getErrorStream()));

        // read the output from the command
        //System.out.println("Here is the standard output of the command:\n");
        String line = "";
```

```
         while ((line = stdInput.readLine()) != null) {
            output = output + line + "\n";
         }
       }
       catch (IOException e) {
          e.printStackTrace();
       } catch (InterruptedException e) {
                         // TODO Auto-generated catch block
                         e.printStackTrace();
                }
                return output;
          }


          public ArrayList<DiscotopePredictionAminoAcid> getAminoAcids() {
                ArrayList<DiscotopePredictionAminoAcid> aaList = new
ArrayList<DiscotopePredictionAminoAcid>();
                String predictionString = getOutput();
                String[] lines = predictionString.split("\n");
                for (String line: lines) {
                         String [] lineArray = line.split("\t");
                         if (lineArray.length > 6) {
                                  DiscotopePredictionAminoAcid aa = new
DiscotopePredictionAminoAcid();
```

```
                    aa.setPredictedEpitope(lineArray.length == 7d);
                    aa.setChain(lineArray[0]);
                    aa.setResidueNo(Integer.parseInt(lineArray[1]));
                    aa.setResidue(lineArray[2]);
                    aa.setContactNo(Integer.parseInt(lineArray[3]));
                    aa.setPropensityScore(Double.parseDouble(lineArray[4]));
                    aa.setScore(Double.parseDouble(lineArray[5]));
                    aaList.add(aa);
                }
            }
            return aaList;
        }


        public static void main (String[]args) {
            DiscotopePredictionService dps = new
DiscotopePredictionService("4hhb","A");
            dps.run();
            System.out.println(dps.getOutput());
            dps.getAminoAcids();
        }
}
package epiprot.services.views;
```

```
public class DiscotopeView {


        public DiscotopeView() {
                // TODO Auto-generated constructor stub
        }


}
package epiprot.services;


public class DocumentNullException extends Exception {
        private String message = null;


   public DocumentNullException() {
      super();
   }


   public DocumentNullException(String message) {
      super(message);
      this.message = message;
   }


   public DocumentNullException(Throwable cause) {
      super(cause);
```

```
    }

    @Override
    public String toString() {
        return message;
    }

    @Override
    public String getMessage() {
        return message;
    }
}
package epiprot.services.epitopePrediction;

import epiprot.AminoAcid;

public class ElliproPredictionAminoAcid extends AminoAcid{

            private int number;
            private String pdbId;
            private int start;
            private int end;
            private String peptide;
            private double score;

            private boolean isLinear;

            private String chain;


            public ElliproPredictionAminoAcid() {

                    // TODO Auto-generated constructor stub

            }


            public int getNumber() {

                    return number;

            }


            public void setNumber(int number) {

                    this.number = number;

            }


            public String getPdbId() {

                    return pdbId;

            }


            public void setPdbId(String pdbId) {

                    this.pdbId = pdbId;

            }
```

```
public int getStart() {
        return start;
}

public void setStart(int start) {
        this.start = start;
}

public int getEnd() {
        return end;
}

public void setEnd(int end) {
        this.end = end;
}

public String getPeptide() {
        return peptide;
}

public void setPeptide(String peptide) {
        this.peptide = peptide;
```

```
}

public double getScore() {
	return score;
}

public void setScore(double score) {
	this.score = score;
}

public boolean isLinear() {
	return isLinear;
}

public void setLinear(boolean isLinear) {
	this.isLinear = isLinear;
}

public String getChain() {
	return chain;
}

public void setChain(String chain) {
		this.chain = chain;
	}
}
package epiprot.services.epitopePrediction;


import epiprot.Epitope;


public class ElliproPredictionEpitope extends Epitope{


	private int number;
	private String pdbId;
	private String chain;
	private double score;
	private boolean isLinear;


	public ElliproPredictionEpitope() {
		// TODO Auto-generated constructor stub
	}


	public int getNumber() {
		return number;
	}
```

```
public void setNumber(int number) {

        this.number = number;

}


public String getPdbId() {

        return pdbId;

}


public void setPdbId(String pdbId) {

        this.pdbId = pdbId;

}


public String getChain() {

        return chain;

}


public void setChain(String chain) {

        this.chain = chain;

}


public double getScore() {

        return score;

}
```

```
		public void setScore(double score) {

			this.score = score;

		}


		public boolean isLinear() {

			return isLinear;

		}


		public void setLinear(boolean isLinear) {

			this.isLinear = isLinear;

		}

}

/**

 * Ellipro.java

 *

 * This service uses an executable file, Ellipro.jar, to predict

 * epitopes. It uses a PDB entry, i.e. 3F6K, to predict epitopes.

 * It produces two tables in a txt file, one table contains continuous

 * epitopes while the other is discontinuous.

 *

 * To start pass the PDB entry id and chain to the constructor, create

 * the file then retrieve the results.
 */
package epiprot.services.epitopePrediction;


import java.io.File;
import java.io.IOException;
import java.util.ArrayList;
import java.util.Scanner;
import epiprot.AminoAcid;
import epiprot.services.Service;


public class ElliproPredictionService extends Service{


        private String pdbId;
        private String chain;


        public ElliproPredictionService(String pdbId, String chain) {
                // TODO Auto-generated constructor stub
                this.pdbId = pdbId;
                this.chain = chain;
        }


        //starts the jar file
   @Override
```

```
        public void run() {


      try {
        // using the Runtime exec method:
        String cmd = "java -jar " + SERVICEFILEPATH + "lib/Ellipro.jar -i " + pdbId +
" -c " + chain;
        System.out.println(cmd);
        Process p = Runtime.getRuntime().exec(cmd);
        p.waitFor();
      }
      catch (IOException e) {
        System.out.println("exception happened - here's what I know: ");
        e.printStackTrace();
      } catch (InterruptedException e) {
                              // TODO Auto-generated catch block
                              e.printStackTrace();
                  }
    }


    public ArrayList<ElliproPredictionEpitope> getEpitopeList() {
        ArrayList<ElliproPredictionEpitope> epitopeList = new
ArrayList<ElliproPredictionEpitope>();
        try (Scanner scanner = new Scanner(new
File(SERVICEFILEPATH+"output.txt"))) {
                        while (scanner.hasNext()){
                                String line = scanner.nextLine();
                                System.out.println(line);
                                if (!line.contains("Structure") && line.length() > 1) {
                                        line.replaceAll("\\s","");
                                        String[] lineArray = line.split(",");
                                        ElliproPredictionEpitope epe = new
ElliproPredictionEpitope();
                                        epe.setNumber(Integer.parseInt(lineArray[0]));
                                        epe.setPdbId(lineArray[1]);
                                        ArrayList<AminoAcid> aaList = new
ArrayList<AminoAcid>();
                                        if (lineArray[lineArray.length-1].equals("Linear"))
{
                                                String residues = lineArray[5];
                                                int start = Integer.parseInt(lineArray[3]);
                                                epe.setChain(lineArray[2]);
                                                epe.setStart(start);
                                                epe.setEnd(Integer.parseInt(lineArray[4]));
                                                epe.setResidues(residues);

        epe.setScore(Double.parseDouble(lineArray[6]));
```

```
                                        epe.setLinear(true);
                                        for (int i = 0; i < residues.length(); i++) {
                                                EllipropredictionAminoAcid aa =
new ElliproPredictionAminoAcid();

        aa.setResidue(residues.substring(i,i+1));
                                                aa.setPosition(start+i);
                                                aa.setChain(lineArray[2]);

        aa.setScore(Double.parseDouble(lineArray[6]));
                                                aa.setLinear(true);

        aa.setNumber(Integer.parseInt(lineArray[0]));
                                                aaList.add(aa);
                                        }
                                }
                                else if (lineArray[lineArray.length-
1].equals("Discontinous")) {
                                        epe.setLinear(false);

        epe.setScore(Double.parseDouble(lineArray[lineArray.length-2]));
                                        String residues = "";
                                        for (int i = 2; i < lineArray.length-3; i++) {
```

```
                                                  String residueArray = lineArray[i];
                                                  int colonPos =
residueArray.indexOf(":");
                                                  String residue =
residueArray.substring(colonPos+1,colonPos+2);
                                                  residues = residues + residue;
                                                  ElliproPredictionAminoAcid aa =
new ElliproPredictionAminoAcid();

     aa.setChain(residueArray.substring(0, colonPos));
                                                  aa.setResidue(residue);

     aa.setPosition(Integer.parseInt(residueArray.substring(colonPos+2)));

     aa.setScore(Double.parseDouble(lineArray[lineArray.length-2]));
                                                  aa.setLinear(false);

     aa.setNumber(Integer.parseInt(lineArray[0]));
                                                  aaList.add(aa);
                                             }
                                             epe.setResidues(residues);
                                        }
                                        epe.setAminoAcidList(aaList);
```

```
                                    epitopeList.add(epe);
                               }
                          }
                     } catch (IOException e) {
                          e.printStackTrace();
                     }
                return epitopeList;
        }

        public static void main (String []args) {
                ElliproPredictionService eps = new ElliproPredictionService("3OW4","A");
                System.out.println(Service.SERVICEFILEPATH);
                eps.run();
                ArrayList<ElliproPredictionEpitope> epitopeList = eps.getEpitopeList();
                for (ElliproPredictionEpitope epitope : epitopeList) {
                        System.out.println(epitope.getNumber() + ". " + epitope.toString());
                }
        }
}
package epiprot.services.views;

public class ElliproView {
	public ElliproView() {
		// TODO Auto-generated constructor stub
	}

}
package epiprot.services.epitopePrediction;

import java.util.ArrayList;

import epiprot.Protein;

public class EminiPredictionService extends IedbEpitopePredictionService {

	public EminiPredictionService(String sequence) {
		super(sequence, "Emini");
		// TODO Auto-generated constructor stub
	}

	public static void main (String[]args) {
		Protein protein = new Protein("Q99523",true);
		System.out.println(protein.getSequence());
		EminiPredictionService eps = new
EminiPredictionService(protein.getSequence());
```

```
		eps.run();
		System.out.println(eps.chart);
		ArrayList<IedbEpitopePredictionAminoAcid> aaList =
eps.getAminoAcids();
		for(IedbEpitopePredictionAminoAcid aa: aaList) {
			System.out.println(aa.toString());
		}
	}
}
package epiprot.services.epitopePrediction;

import epiprot.AminoAcid;

public class IedbEpitopePredictionAminoAcid extends AminoAcid {

	private int start;
	private int end;
	private String peptide;
	private double score;
	private int relativeScore;

	public IedbEpitopePredictionAminoAcid(){}
public int getStart() {
        return start;
}
public void setStart(int start) {
        this.start = start;
}
public int getEnd() {
        return end;
}
public void setEnd(int end) {
        this.end = end;
}
public String getPeptide() {
        return peptide;
}
public void setPeptide(String peptide) {
        this.peptide = peptide;
}
public double getScore() {
        return score;
}
public void setScore(double score) {
        this.score = score;
```

```
	}

	public int getRelativeScore() {
		return relativeScore;
	}

	public void setRelativeScore(int relativeScore) {
		this.relativeScore = relativeScore;
	}
}
package epiprot.services.epitopePrediction;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.ArrayList;
import java.util.Collections;
import java.util.Comparator;
import java.util.Iterator;
import java.util.stream.Stream;

import epiprot.Protein;
import epiprot.services.Service;
```

```
public class IedbEpitopePredictionService extends Service {

	private String sequence;
	private String epitopePredictionMethod;
	private int windowSize;

	public String chart;

	//sets windowSize automatically to 7
	public IedbEpitopePredictionService(String sequence, String
epitopePredictionMethod) {
		this.sequence = sequence;
		this.epitopePredictionMethod = epitopePredictionMethod;
		this.windowSize = 7;
	}

	public IedbEpitopePredictionService(String sequence, String
epitopePredictionMethod, int windowSize) {
		this.sequence = sequence;
		this.epitopePredictionMethod = epitopePredictionMethod;
		System.out.println(sequence);
		this.windowSize = windowSize;
```

```
    }

    //wrapper for getPredictionChart
    @Override
    public void run() {
    String s = null;
    chart = epitopePredictionMethod + " Prediction Chart\n";

  try {
    // using the Runtime exec method:
    String cmd = "";

    cmd = "curl http://tools-api.iedb.org/tools_api/bcell/ --data method=" +
epitopePredictionMethod + "&sequence_text=" + sequence +"&window_size=" +
windowSize;

    Process p = Runtime.getRuntime().exec(cmd);
    p.waitFor();

    BufferedReader stdInput = new BufferedReader(new
        InputStreamReader(p.getInputStream()));

    BufferedReader stdError = new BufferedReader(new
         InputStreamReader(p.getErrorStream()));



    Stream<String> lines = stdInput.lines();
    Iterator it = lines.iterator();
    while(it.hasNext()) {
            //System.out.println(it.next());
            chart = chart + it.next() + "\n";
    }
    System.out.println(chart);

    /* read any errors from the attempted command
    System.out.println("Here is the standard error of the command (if any):\n");
    while ((s = stdError.readLine()) != null) {
       System.out.println(s);
    }*/
}
catch (IOException e) {
    System.out.println("exception happened - here's what I know: ");
    e.printStackTrace();
} catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
```

```
                }
    }


    //returns an arraylist of IedbEpiprotPredictionAminoAcid
    public ArrayList<IedbEpitopePredictionAminoAcid> getAminoAcids() {
            ArrayList<IedbEpitopePredictionAminoAcid> aaList = new
ArrayList<IedbEpitopePredictionAminoAcid>();
            String[] chartArray = chart.split("\n");
            boolean isLine = false;
            for (String chartLine: chartArray) {
                    if(isLine) {
                            if (chartLine != null && !chartLine.equals("") &&
chartLine.length() > 1) {
                                    String [] chartLineArray = chartLine.split("\\s+");
                                    IedbEpitopePredictionAminoAcid aa = new
IedbEpitopePredictionAminoAcid();
                            if (chartLineArray != null && chartLineArray[1] != null) {
                                    aa.setPosition(Integer.parseInt(chartLineArray[0]));
                                    aa.setResidue(chartLineArray[1]);
                                    if (epitopePredictionMethod.equals("Bepipred")) {

            aa.setScore(Double.parseDouble(chartLineArray[3]));
                                    }
```

```
                              else {
                                        aa.setStart(Integer.parseInt(chartLineArray[2]));
                                        aa.setEnd(Integer.parseInt(chartLineArray[3]));
                                        aa.setPeptide(chartLineArray[4]);

        aa.setScore(Double.parseDouble(chartLineArray[5]));
                              }
                    }
                    aaList.add(aa);
                    }
          }
          if(chartLine.contains("Position")) {
                    isLine = true;
          }
        }
        return calcDisplayedScores(aaList);
  }

  private ArrayList<IedbEpitopePredictionAminoAcid>
calcDisplayedScores(ArrayList<IedbEpitopePredictionAminoAcid> aaList) {
        ArrayList<IedbEpitopePredictionAminoAcid> sortedAAList = new
ArrayList<IedbEpitopePredictionAminoAcid>(aaList);
        Collections.sort(sortedAAList, new
Comparator<IedbEpitopePredictionAminoAcid>(){
            @Override
                        public int compare(IedbEpitopePredictionAminoAcid o1,
IedbEpitopePredictionAminoAcid o2){
              if(o1.getScore() == o2.getScore())
                return 0;
              return o1.getScore() < o2.getScore() ? -1 : 1;
            }
        });

        int tenth = aaList.size()/10;
        int remainder = aaList.size()%10;

        for(int i = 0; i < 10; i++) {
                for (int j = i*tenth; j < (i*tenth)+tenth; j++) {
                        sortedAAList.get(j).setRelativeScore(i);
                }
        }

        for (int i = aaList.size()-1; i >= aaList.size()-remainder; i--) {
                sortedAAList.get(i).setRelativeScore(9);
        }
        Collections.sort(sortedAAList, new
Comparator<IedbEpitopePredictionAminoAcid>(){
            @Override
                public int compare(IedbEpitopePredictionAminoAcid o1,
IedbEpitopePredictionAminoAcid o2){
                if(o1.getPosition() == o2.getPosition())
                    return 0;
                return o1.getPosition() < o2.getPosition() ? -1 : 1;
            }
        });

        return sortedAAList;
    }

    public static void main (String[]args) {
        Protein protein = new Protein("Q99523", true);
        IedbEpitopePredictionService ieps = new
IedbEpitopePredictionService(protein.getSequence(),"Emini");
        ieps.run();
        ArrayList<IedbEpitopePredictionAminoAcid> aminoAcids =
ieps.getAminoAcids();
        for (IedbEpitopePredictionAminoAcid aa: aminoAcids) {
            System.out.println(aa.toString()+"|"+aa.getRelativeScore()+"|"+aa.getScore());
        }
    }
}
package epiprot.services.views;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;

import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

import epiprot.Presenter;
import epiprot.services.epitopePrediction.ChouPredictionService;
import epiprot.services.epitopePrediction.EminiPredictionService;
import epiprot.services.epitopePrediction.IedbEpitopePredictionAminoAcid;
import epiprot.services.epitopePrediction.KarplusPredictionService;
import epiprot.services.epitopePrediction.KolaskarPredictionService;
import epiprot.services.epitopePrediction.ParkerPredictionService;
```

```
public class IEDBPredPresenter {

	Presenter presenter;
	int proteinSeqLength;

	interface View {
		JTextField windowSizeTextField();
		//JCheckBox bepipredCheckBox();
		JCheckBox chouCheckBox();
		JCheckBox eminiNewCheckBox();
		JCheckBox karplusCheckBox();
		JCheckBox kolaskarCheckBox();
		JCheckBox parkerCheckBox();
		JButton btnSubmit();

	}

	public IEDBPredPresenter(Presenter presenter) {
		// TODO Auto-generated constructor stub
		this.presenter = presenter;
		this.proteinSeqLength = presenter.protein.getSequence().length();
		bindHandlers();
	}

	View view = new IEDBPredView();

	void bindHandlers() {
		view.btnSubmit().addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				// TODO Auto-generated method stub/
				/*
				if (view.bepipredCheckBox().isSelected()) {
					BepipredPredictionService ps = new
BepipredPredictionService(presenter.protein.getSequence());
					ps.run();
					ArrayList<IedbEpitopePredictionAminoAcid>
aminoAcids = ps.getAminoAcids();
					insertPredictionLine("Bepipred Epitope
Prediction",presenter.getMainLine(), aminoAcids);
				}*/
				if (view.chouCheckBox().isSelected()) {
					ChouPredictionService ps = new
ChouPredictionService(presenter.protein.getSequence());
					ps.run();
				ArrayList<IedbEpitopePredictionAminoAcid>
aminoAcids = ps.getAminoAcids();
				insertPredictionLine("Chou Epitope
Prediction",presenter.getMainLine(), aminoAcids);
			}
			if (view.eminiNewCheckBox().isSelected()) {
				EminiPredictionService ps = new
EminiPredictionService(presenter.protein.getSequence());
				ps.run();
				ArrayList<IedbEpitopePredictionAminoAcid>
aminoAcids = ps.getAminoAcids();
				insertPredictionLine("Emini Epitope
Prediction",presenter.getMainLine(), aminoAcids);
			}
			if (view.karplusCheckBox().isSelected()) {
				KarplusPredictionService ps = new
KarplusPredictionService(presenter.protein.getSequence());
				ps.run();
				ArrayList<IedbEpitopePredictionAminoAcid>
aminoAcids = ps.getAminoAcids();
				insertPredictionLine("Karplus Epitope
Prediction",presenter.getMainLine(), aminoAcids);
			}
```

```
                    if (view.kolaskarCheckBox().isSelected()) {
                              KolaskarPredictionService ps = new
KolaskarPredictionService(presenter.protein.getSequence());
                              ps.run();
                              ArrayList<IedbEpitopePredictionAminoAcid>
aminoAcids = ps.getAminoAcids();
                              insertPredictionLine("Kolaskar Epitope
Prediction",presenter.getMainLine(), aminoAcids);
                    }
                    if (view.parkerCheckBox().isSelected()) {
                              ParkerPredictionService ps = new
ParkerPredictionService(presenter.protein.getSequence());
                              ps.run();
                              ArrayList<IedbEpitopePredictionAminoAcid>
aminoAcids = ps.getAminoAcids();
                              insertPredictionLine("Parker Epitope
Prediction",presenter.getMainLine(), aminoAcids);
                    }
          }

          public void insertPredictionLine(String header, String mainLine,
ArrayList<IedbEpitopePredictionAminoAcid> aminoAcids) {
                    String aaListSequence ="";
```

```
                                for (int i = 0; i < aminoAcids.size(); i++) {
                                        aaListSequence = aaListSequence +
aminoAcids.get(i).getResidue();
                                }
                                String insertLine = "";
                                int mainIndex = 0;
                                for(int i = 0; i < aminoAcids.size(); i++) {
                                        if(Character.isLetter(mainLine.charAt(mainIndex)))
{
                                                int score =
aminoAcids.get(i).getRelativeScore();
                                                insertLine = insertLine + score;
                                        }
                                        else {
                                                insertLine = insertLine + "-";
                                                i--;
                                        }
                                        mainIndex++;
                                }
                                presenter.insertLineAboveTarget(header, insertLine);
                        }
                });
```

```
		view.windowSizeTextField().getDocument().addDocumentListener(new
DocumentListener() {
			@Override
			public void insertUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
			@Override
			public void removeUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
			@Override
			public void changedUpdate(DocumentEvent e) {
				// TODO Auto-generated method stub
				checkTextField();
			}
			public void checkTextField() {
				String text = view.windowSizeTextField().getText();
				if(isInteger(text) && Integer.parseInt(text) <
proteinSeqLength && Integer.parseInt(text) > 0) {
					view.btnSubmit().setEnabled(true);
				}
```

```
                        else {
                                view.btnSubmit().setEnabled(false);
                        }
                }
        });
    }

    private boolean isInteger(String s) {
        try {
           Integer.parseInt(s);
       } catch(NumberFormatException e) {
           return false;
       } catch(NullPointerException e) {
           return false;
       }
       // only got here if we didn't return false
       return true;
    }
}
package epiprot.services.views;


import javax.swing.JFrame;
import javax.swing.JLabel;
import java.awt.Dimension;

import java.awt.GridBagLayout;


import javax.swing.JScrollPane;

import javax.swing.JTextField;


import java.awt.BorderLayout;

import javax.swing.JPanel;

import javax.swing.BorderFactory;

import javax.swing.JCheckBox;

import java.awt.GridBagConstraints;

import java.awt.Insets;

import javax.swing.JButton;


public class IEDBPredView extends JFrame implements IEDBPredPresenter.View {


        JTextField windowSizeTextField = new JTextField("6");

        //JCheckBox bepipredCheckBox = new JCheckBox("Bepipred Linear Epitope
Prediction");

        JCheckBox chouCheckBox = new JCheckBox("Chou & Fasman Beta-Turn
Prediction");

        JCheckBox eminiNewCheckBox = new JCheckBox("Emini Surface Accessibility
Prediction");
	JCheckBox karplusCheckBox = new JCheckBox("Karplus & Schulz Flexibility
Prediction");
	JCheckBox kolaskarCheckBox = new JCheckBox("Kolaskar & Tongaonkar
Antigenicity");
	JCheckBox parkerCheckBox = new JCheckBox("Parker Hydrophilicity
Prediction");
	JButton btnSubmit = new JButton("Submit");
	private final JLabel lblNewLabel = new JLabel("Enter a number between 0 and
the length of the protein");

	public IEDBPredView() {
		setSize(new Dimension(700, 300));

		JScrollPane scrollPane = new JScrollPane();
		getContentPane().add(scrollPane, BorderLayout.CENTER);

		JPanel panel = new JPanel();
		scrollPane.setViewportView(panel);
		panel.setLayout(new GridBagLayout());
		/*

	bepipredCheckBox.setHorizontalAlignment(SwingConstants.TRAILING);
```

```
		GridBagConstraints gbc_chckbxNewCheckBox = new
GridBagConstraints();
		gbc_chckbxNewCheckBox.insets = new Insets(5, 5, 5, 5);
		gbc_chckbxNewCheckBox.anchor = GridBagConstraints.WEST;
		gbc_chckbxNewCheckBox.weightx = 1.0;
		gbc_chckbxNewCheckBox.weighty = 1.0;
		gbc_chckbxNewCheckBox.gridx = 0;
		gbc_chckbxNewCheckBox.gridy = 0;
		panel.add(bepipredCheckBox, gbc_chckbxNewCheckBox);
		*/
		GridBagConstraints gbc_chckbxNewCheckBox_1 = new
GridBagConstraints();
		gbc_chckbxNewCheckBox_1.insets = new Insets(5, 5, 5, 0);
		gbc_chckbxNewCheckBox_1.anchor = GridBagConstraints.WEST;
		gbc_chckbxNewCheckBox_1.weightx = 1.0;
		gbc_chckbxNewCheckBox_1.weighty = 1.0;
		gbc_chckbxNewCheckBox_1.gridx = 1;
		gbc_chckbxNewCheckBox_1.gridy = 0;
		panel.add(chouCheckBox, gbc_chckbxNewCheckBox_1);

		GridBagConstraints gbc_chckbxNewCheckBox_2 = new
GridBagConstraints();
		gbc_chckbxNewCheckBox_2.insets = new Insets(5, 5, 5, 5);
		gbc_chckbxNewCheckBox_2.anchor = GridBagConstraints.WEST;
		gbc_chckbxNewCheckBox_2.weightx = 1.0;
		gbc_chckbxNewCheckBox_2.weighty = 1.0;
		gbc_chckbxNewCheckBox_2.gridx = 0;
		gbc_chckbxNewCheckBox_2.gridy = 1;
		panel.add(eminiNewCheckBox, gbc_chckbxNewCheckBox_2);

		GridBagConstraints gbc_chckbxNewCheckBox_3 = new
GridBagConstraints();
		gbc_chckbxNewCheckBox_3.insets = new Insets(5, 5, 5, 0);
		gbc_chckbxNewCheckBox_3.anchor = GridBagConstraints.WEST;
		gbc_chckbxNewCheckBox_2.weightx = 1.0;
		gbc_chckbxNewCheckBox_2.weighty = 1.0;
		gbc_chckbxNewCheckBox_3.gridx = 1;
		gbc_chckbxNewCheckBox_3.gridy = 1;
		panel.add(karplusCheckBox, gbc_chckbxNewCheckBox_3);

		GridBagConstraints gbc_chckbxNewCheckBox_4 = new
GridBagConstraints();
		gbc_chckbxNewCheckBox_4.insets = new Insets(5, 5, 5, 5);
		gbc_chckbxNewCheckBox_4.anchor = GridBagConstraints.WEST;
		gbc_chckbxNewCheckBox_4.weightx = 1.0;
		gbc_chckbxNewCheckBox_4.weighty = 1.0;
		gbc_chckbxNewCheckBox_4.gridx = 0;
		gbc_chckbxNewCheckBox_4.gridy = 2;
		panel.add(kolaskarCheckBox, gbc_chckbxNewCheckBox_4);

		GridBagConstraints gbc_chckbxNewCheckBox_5 = new
GridBagConstraints();
		gbc_chckbxNewCheckBox_5.insets = new Insets(5, 5, 5, 0);
		gbc_chckbxNewCheckBox_5.anchor = GridBagConstraints.WEST;
		gbc_chckbxNewCheckBox_5.weightx = 1.0;
		gbc_chckbxNewCheckBox_5.weighty = 1.0;
		gbc_chckbxNewCheckBox_5.gridx = 1;
		gbc_chckbxNewCheckBox_5.gridy = 2;
		panel.add(parkerCheckBox, gbc_chckbxNewCheckBox_5);

		JPanel windowPanel = new JPanel();
		windowPanel.setOpaque(true);
		//windowPanel.setBackground(Color.WHITE);
		windowPanel.setBorder(
    BorderFactory.createTitledBorder("Window Size"));
      GridBagConstraints gbc_windowSizePanel = new GridBagConstraints();
      gbc_windowSizePanel.fill = GridBagConstraints.BOTH;
      gbc_windowSizePanel.insets = new Insets(5, 5, 5, 5);
      gbc_windowSizePanel.anchor = GridBagConstraints.WEST;
gbc_windowSizePanel.weightx = 1.0;
gbc_windowSizePanel.weighty = 1.0;
gbc_windowSizePanel.gridx = 0;
gbc_windowSizePanel.gridy = 3;
gbc_windowSizePanel.gridheight = 2;
        panel.add(windowPanel, gbc_windowSizePanel);
        windowPanel.setLayout(new BorderLayout(0, 0));

        windowPanel.add(windowSizeTextField, BorderLayout.CENTER);
        windowSizeTextField.setColumns(10);

        GridBagConstraints gbc_notificationLabel = new GridBagConstraints();
        gbc_notificationLabel.insets = new Insets(0, 0, 5, 0);
        gbc_notificationLabel.gridx = 1;
        gbc_notificationLabel.gridy = 4;
        panel.add(lblNewLabel, gbc_notificationLabel);

        GridBagConstraints gbc_btnSubmit = new GridBagConstraints();
        gbc_btnSubmit.gridx = 1;
        gbc_btnSubmit.gridy = 5;
        panel.add(btnSubmit, gbc_btnSubmit);
```

```
		// TODO Auto-generated constructor stub
		setVisible(true);


	}


	@Override
	public JTextField windowSizeTextField() {
		// TODO Auto-generated method stub
		return windowSizeTextField;
	}
/*
	@Override
	public JCheckBox bepipredCheckBox() {
		// TODO Auto-generated method stub
		return bepipredCheckBox;
	}
*/
	@Override
	public JCheckBox chouCheckBox() {
		// TODO Auto-generated method stub
		return chouCheckBox;
	}
```

```
@Override
public JCheckBox eminiNewCheckBox() {
	// TODO Auto-generated method stub
	return eminiNewCheckBox;
}

@Override
public JCheckBox karplusCheckBox() {
	// TODO Auto-generated method stub
	return karplusCheckBox;
}

@Override
public JCheckBox kolaskarCheckBox() {
	// TODO Auto-generated method stub
	return kolaskarCheckBox;
}

@Override
public JCheckBox parkerCheckBox() {
	// TODO Auto-generated method stub
	return parkerCheckBox;
}
```

```
	@Override
	public JButton btnSubmit() {
		// TODO Auto-generated method stub
		return btnSubmit;
	}

}
package epiprot.services.jabaws;

public interface JabawsConstants {

	String MUSCLE = "MuscleWS";
	String CLUSTALOMEGA = "ClustalOWS";
	String CLUSTAL = "ClustalWS";
	String TCOFFEE = "TcoffeeWS";
	String MAFFT = "MafftWS";
	String PROBCONS = "ProbconsWS";
	String MSAPROBS = "MSAprobsWS";
	String GLPROBS = "GLprobsWS";

}
package epiprot.services.jabaws;
```

```
import java.io.BufferedReader;
import java.io.BufferedWriter;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import java.util.List;
import org.apache.commons.io.FileUtils;

import epiprot.Protein;
import epiprot.services.Service;

public class JabawsService extends Service implements JabawsConstants {

	private String msa;
	private File msaInputFile;
	private ArrayList<String> proteinAccs;
	private String cmd = "java -jar "+SERVICEFILEPATH+"lib/Jabaws.jar -
h=http://www.compbio.dundee.ac.uk/jabaws -s=";
	private LinkedHashMap<String,String> proteinMap = new
LinkedHashMap<String,String>();

	public JabawsService(String msa, File inputFile) {
		// TODO Auto-generated constructor stub
		this.msa = msa;
		this.msaInputFile = msaInputFile;
		//setProteinMap();
	}

	public JabawsService(String msa, ArrayList<String> proteinAccs) {
		// TODO Auto-generated constructor stub
		this.msa = msa;
		this.proteinAccs = proteinAccs;
		setFile();
		//setProteinMap();
	}

	@Override
	public void run() {
				// TODO Auto-generated method stub
				try {
					cmd = cmd + msa + " -i=" + msaInputFile.getAbsolutePath();
			Process p = Runtime.getRuntime().exec(cmd);
			p.waitFor();

			BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));
			BufferedReader stdError = new BufferedReader(new
InputStreamReader(p.getErrorStream()));
			String line = "";
			while ((line = stdInput.readLine()) != null) {
				if (line.contains("|")) {
					System.out.println(line);
					String[] a1 = line.split("\\s+");
					//String[] a2 = a1[0].split("\\|");
					String key = a1[0];

					String value = "";
					if (proteinMap.containsKey(key)) {
						value = proteinMap.get(key);
					}
					value = value + a1[1];
```

```
                        proteinMap.put(key, value);
                    }
                }
                msaInputFile.delete();
                    } catch (IOException e) {
                System.out.println("exception happened - here's what I know: ");
                e.printStackTrace();
            } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
            }
                }


                public LinkedHashMap<String,String> getAlignment() {
                    return proteinMap;
                }


                private void setProteinMap() {
                    try {
                        BufferedReader br = new BufferedReader(new
FileReader(msaInputFile));
                        String line;
                         while ((line = br.readLine()) != null) {
```

```
                    if (line.contains(">")) {
                        String[] a1 = line.split("\\|");
                        proteinMap.put(a1[1],"");


                    }
                }
            } catch (FileNotFoundException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
        }


        private void setFile() {
            for(String acc: proteinAccs) {
                msaInputFile = new
File(SERVICEFILEPATH+"msaInputFile.txt");
                Protein protein = new Protein(acc,true);
                try {
                    File proteinFile = protein.getFastaFile();
                    String str = FileUtils.readFileToString(proteinFile);
```

```
                        PrintWriter out = new PrintWriter(new
BufferedWriter(new FileWriter(msaInputFile, true)));
                        out.println(str);
                        proteinFile.delete();
                    out.close();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
            }
        }

        public static void main (String[]args) {
            ArrayList<String> proteinAccs = new ArrayList<String>();
            proteinAccs.add("P31749");
            proteinAccs.add("P31751");
            proteinAccs.add("Q9Y243");
            proteinAccs.add("Q9Y243-2");
            JabawsService js = new
JabawsService(JabawsConstants.CLUSTALOMEGA,proteinAccs);
            js.run();
            LinkedHashMap<String,String> proteinMap = js.getAlignment();
            List<String> sequences = new ArrayList<String>(proteinMap.values());
```

```
		List<String> headers = new ArrayList<String>(proteinMap.keySet());
		for(int i = 0; i < sequences.size(); i++) {
			System.out.println(headers.get(i)+" "+sequences.get(i));
		}
	}
}
package epiprot.services.ssp;

import epiprot.AminoAcid;

public class JPredAminoAcid extends AminoAcid {

	private String prediction;
	private int confidenceScore;

	public JPredAminoAcid() {
		// TODO Auto-generated constructor stub
	}

	public String getPrediction() {
		return prediction;
	}
	public void setPrediction(String prediction) {
		this.prediction = prediction;
	}

	public int getConfidenceScore() {
		return confidenceScore;
	}

	public void setConfidenceScore(int confidenceScore) {
		this.confidenceScore = confidenceScore;
	}

	@Override
	public String toString() {
		return "Conf: " + confidenceScore + " Pred: " + prediction + " AA: " +
super.getResidue();
	}

}
package epiprot.services.ssp;

public class JPredSequenceTooLongException extends Exception {
    private String message = null;

public JPredSequenceTooLongException() {
  super();
}

public JPredSequenceTooLongException(String message) {
  super(message);
  this.message = message;
}

public JPredSequenceTooLongException(Throwable cause) {
  super(cause);
}

@Override
public String toString() {
  return message;
}

@Override
public String getMessage() {
  return message;
```

```
    }


}
package epiprot.services.ssp;


import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.MalformedURLException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Iterator;
import java.util.stream.Stream;


import epiprot.Protein;
import epiprot.services.Service;


public class JPredService extends Service {


        private File fastaFile;
```

```
	private String sequence;

	// usr/local/jpred/
	final private static String JPREDPATH = "/usr/local/jpred/";

	private ArrayList<JPredAminoAcid> aminoAcids = new
ArrayList<JPredAminoAcid>();

	public JPredService(File fastaFile) throws JPredSequenceTooLongException {
		// TODO Auto-generated constructor stub
		this.fastaFile = fastaFile;
		checkSequenceLength();
	}

	private JPredService(String sequence) {
		this.sequence = sequence;
	}

	@Override
	public void run() {
		// TODO Auto-generated method stub
		String s;
		String urlString = "";
		try {
	// using the Runtime exec method:
	String cmd = "perl "+JPREDPATH+"jpredapi submit mode=single format=fasta
email=name@domain.com file="+fastaFile.getAbsolutePath()+" name=TestJob";
	System.out.println(cmd);
	Process p = Runtime.getRuntime().exec(cmd);
	p.waitFor();

	BufferedReader stdInput = new BufferedReader(new
InputStreamReader(p.getInputStream()));
	BufferedReader stdError = new BufferedReader(new
InputStreamReader(p.getErrorStream()));

	// read the output from the command
	String jobid = "";
	String cmd2 = "";
	while ((s = stdInput.readLine()) != null) {
		System.out.println(s);
		if (s.contains("Created JPred job with jobid:")) {
			jobid = s.substring(30);
		}
		else if (s.contains("...or using 'perl")) {
			String [] array = s.split("'");
```

```
                        cmd2 = array[1];
                        cmd2 = cmd2.replace("jpredapi", JPREDPATH+"jpredapi");
                        cmd2 = cmd2.replace("getResults=yes", "getResults=no");
                                cmd2 = cmd2.replace("checkEvery=60",
"checkEvery=10");
                }
        }
        // read any errors from the attempted command
        while ((s = stdError.readLine()) != null) {
           System.out.println(s);
        }
        System.out.println(jobid);
        System.out.println(cmd2);

        //second part of JPred
        Process p2 = Runtime.getRuntime().exec(cmd2);

        stdInput = new BufferedReader(new InputStreamReader(p2.getInputStream()));
        stdError = new BufferedReader(new InputStreamReader(p2.getErrorStream()));

        // read the output from the command

        while ((s = stdInput.readLine()) != null) {
                System.out.println(s);
                if (s.contains("http://www.compbio.dundee.ac.uk")) {
                        urlString = s;
                }
        }
        System.out.println(urlString);


        // read any errors from the attempted command
        while ((s = stdError.readLine()) != null) {
           System.out.println(s);
        }
    }
    catch (IOException e) {
        System.out.println("exception happened - here's what I know: ");
        e.printStackTrace();
        //System.exit(-1);
    } catch (InterruptedException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }


                //change urlString to .jnet
                urlString = urlString.replace(".results.html", ".jnet");
                try {
                      URL url = new URL(urlString);
                      BufferedReader in = new BufferedReader(new
    InputStreamReader(url.openStream()));

                  String inputLine;
                  String jpredline = "";
                  String jpredConf = "";
                  while ((inputLine = in.readLine()) != null) {
                     System.out.println(inputLine);
                     if (inputLine.contains("jnetpred:")) {
                        jpredline = inputLine.substring(9);
                     }
                     else if (inputLine.contains("JNETCONF:")) {
                        jpredConf = inputLine.substring(9);
                     }
                  }
                  String[] jpredlineArray = jpredline.split(",");
                  String[] jpredConfArray = jpredConf.split(",");

                  for(int i = 0; i < jpredlineArray.length; i++) {
                        JPredAminoAcid aa = new JPredAminoAcid();
```

```
                        aa.setPosition(i+1);

                        aa.setResidue(sequence.substring(i, i+1));

                        aa.setPrediction(jpredlineArray[i]);

                        aa.setConfidenceScore(Integer.parseInt(jpredConfArray[i]));

                        aminoAcids.add(aa);

                    }


                    in.close();

                } catch (MalformedURLException e) {

                        // TODO Auto-generated catch block

                        e.printStackTrace();

                } catch (IOException e) {

                        // TODO Auto-generated catch block

                        e.printStackTrace();

                }

        }


        private void checkSequenceLength() throws JPredSequenceTooLongException {

                String urlString = "";

                try {

                        BufferedReader br = new BufferedReader(new
FileReader(fastaFile));

                        Stream<String> lineStream = br.lines();
```

```
			Iterator<String> it = lineStream.iterator();
			StringBuilder sb = new StringBuilder();
			it.next();
			while(it.hasNext()) {
				String line = it.next();
				line = line.replace("\n", "");
				sb.append(line);
			}
			br.close();

			sequence = sb.toString();
			if (sequence.length() > 800) {
				throw new JPredSequenceTooLongException("Protein
sequence is too long. Must be under 800 amino acids.");
			}
		} catch (FileNotFoundException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
```

```
public ArrayList<JPredAminoAcid> getAminoAcids() {
        return aminoAcids;
}

public void setAminoAcids(ArrayList<JPredAminoAcid> aminoAcids) {
        this.aminoAcids = aminoAcids;
}

public static void main(String[]args) {
        Protein protein = new Protein("P31749");
        System.out.println("test1");
        JPredService pps;
        try {
                pps = new JPredService(protein.getFastaFile());
                pps.run();
                System.out.println("test2");
                ArrayList<JPredAminoAcid> aaList = pps.getAminoAcids();
                System.out.println(aaList.size());
                for (JPredAminoAcid aa : aaList) {
                        System.out.println(aa.toString());
                }
        } catch (JPredSequenceTooLongException e) {
```

```
                    // TODO Auto-generated catch block
                    e.printStackTrace();
                }
        }
}
package epiprot.services.epitopePrediction;

public class KarplusPredictionService extends IedbEpitopePredictionService{

        public KarplusPredictionService(String sequence) {
                // TODO Auto-generated constructor stub
                super(sequence,"Karplus-Schulz");
        }
}
package epiprot.services.epitopePrediction;

public class KolaskarPredictionService extends IedbEpitopePredictionService{

        public KolaskarPredictionService(String sequence) {
                // TODO Auto-generated constructor stub
                super(sequence,"Kolaskar-Tongaonkar");
        }
```

```
}

package epiprot.services.views;


import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.Enumeration;


import javax.swing.AbstractButton;

import javax.swing.ButtonGroup;

import javax.swing.JButton;

import javax.swing.JRadioButton;


import epiprot.Presenter;

import epiprot.services.jabaws.JabawsConstants;


public class MSAPresenter {


        Presenter presenter;


        interface View {

                JRadioButton muscleWSRadioButton();

                JRadioButton clustalOWSRadioButton();

                JRadioButton clustalWSRadioButton();
                JRadioButton tCoffeeWSRadioButton();

                JRadioButton mafftWSRadioButton();

                JRadioButton probconsWSRadioButton();

                JRadioButton msaProbsWSRadioButton();

                JRadioButton glProbsWSRadioButton();


                JButton btnSubmit();


                ButtonGroup buttonGroup();

        }


        View view = new MSAView();


        public MSAPresenter(Presenter presenter) {

                // TODO Auto-generated constructor stub

                this.presenter = presenter;

                bindHandlers();

        }


        void bindHandlers() {

                view.btnSubmit().addActionListener(new ActionListener() {

                        @Override

                        public void actionPerformed(ActionEvent e) {
                                // TODO Auto-generated method stub
                                Enumeration<AbstractButton> allRadioButton=
        view.buttonGroup().getElements();
                        while(allRadioButton.hasMoreElements()) {
                           JRadioButton temp = (JRadioButton)
        allRadioButton.nextElement();
                           if(temp.isSelected()) {
                                String msaProgram = JabawsConstants.MUSCLE;

                                if(temp.getText().equals("Clustal Omega")) {msaProgram
        = JabawsConstants.CLUSTALOMEGA;}
                                else if(temp.getText().equals("Clustal W")) {msaProgram =
        JabawsConstants.CLUSTAL;}
                                else if(temp.getText().equals("T-Coffee")) {msaProgram =
        JabawsConstants.TCOFFEE;}
                                else if(temp.getText().equals("Mafft")) {msaProgram =
        JabawsConstants.MAFFT;}
                                else if(temp.getText().equals("ProbCons")) {msaProgram =
        JabawsConstants.PROBCONS;}
                                else if(temp.getText().equals("MSAProbs")) {msaProgram
        = JabawsConstants.MSAPROBS;}
                                else if(temp.getText().equals("GLProbs")) {msaProgram =
        JabawsConstants.GLPROBS;}
```

```
                        presenter.setAlignment(msaProgram);
                        Presenter.msaProteinList.clear();
                        break;
                    }
                }
                    }
            });
        }


}
package epiprot.services.views;


import javax.swing.JFrame;
import java.awt.Dimension;
import javax.swing.JPanel;
import java.awt.BorderLayout;
import java.awt.GridLayout;
import javax.swing.JRadioButton;
import javax.swing.ButtonGroup;
import javax.swing.JButton;


public class MSAView extends JFrame implements MSAPresenter.View {
```

```
JRadioButton muscleWSRadioButton = new JRadioButton("MUSCLE");
JRadioButton clustalOWSRadioButton = new JRadioButton("Clustal Omega");
JRadioButton clustalWSRadioButton = new JRadioButton("Clustal W");
JRadioButton tCoffeeWSRadioButton = new JRadioButton("T-Coffee");
JRadioButton mafftWSRadioButton = new JRadioButton("Mafft");
JRadioButton probconsWSRadioButton = new JRadioButton("ProbCons");
JRadioButton msaProbsWSRadioButton = new JRadioButton("MSAProbs");
JRadioButton glProbsWSRadioButton = new JRadioButton("GLProbs");

JButton btnSubmit = new JButton("Submit");

ButtonGroup buttonGroup = new ButtonGroup();

public MSAView() {
        setSize(new Dimension(300, 200));
        // TODO Auto-generated constructor stub
        setTitle("Multiple Sequence Alignment");
        getContentPane().setPreferredSize(new Dimension(400, 400));

        JPanel panel = new JPanel();
        getContentPane().add(panel, BorderLayout.CENTER);
        panel.setLayout(new GridLayout(6, 2, 0, 0));
        panel.add(muscleWSRadioButton);
        muscleWSRadioButton.setSelected(true);
        panel.add(clustalOWSRadioButton);
        panel.add(clustalWSRadioButton);
        panel.add(tCoffeeWSRadioButton);
        panel.add(mafftWSRadioButton);
        panel.add(probconsWSRadioButton);
        panel.add(msaProbsWSRadioButton);
        panel.add(glProbsWSRadioButton);

        buttonGroup.add(muscleWSRadioButton);
        buttonGroup.add(glProbsWSRadioButton);;
        buttonGroup.add(msaProbsWSRadioButton);
        buttonGroup.add(probconsWSRadioButton);
        buttonGroup.add(mafftWSRadioButton);
        buttonGroup.add(tCoffeeWSRadioButton);
        buttonGroup.add(clustalWSRadioButton);
        buttonGroup.add(clustalOWSRadioButton);

        panel.add(btnSubmit);
        setVisible(true);
    }
```

```
@Override
public JRadioButton muscleWSRadioButton() {
        // TODO Auto-generated method stub
        return muscleWSRadioButton;
}

@Override
public JRadioButton clustalOWSRadioButton() {
        // TODO Auto-generated method stub
        return clustalOWSRadioButton;
}

@Override
public JRadioButton clustalWSRadioButton() {
        // TODO Auto-generated method stub
        return clustalWSRadioButton;
}

@Override
public JRadioButton tCoffeeWSRadioButton() {
        // TODO Auto-generated method stub
        return tCoffeeWSRadioButton;
}
```

```
@Override
public JRadioButton mafftWSRadioButton() {
	// TODO Auto-generated method stub
	return mafftWSRadioButton;
}

@Override
public JRadioButton probconsWSRadioButton() {
	// TODO Auto-generated method stub
	return probconsWSRadioButton;
}

@Override
public JRadioButton msaProbsWSRadioButton() {
	// TODO Auto-generated method stub
	return msaProbsWSRadioButton;
}

@Override
public JRadioButton glProbsWSRadioButton() {
	// TODO Auto-generated method stub
	return glProbsWSRadioButton;
	}

	@Override
	public JButton btnSubmit() {
		// TODO Auto-generated method stub
		return btnSubmit;
	}

	@Override
	public ButtonGroup buttonGroup() {
		// TODO Auto-generated method stub
		return buttonGroup;
	}

}
package epiprot.services.epitopePrediction;

public class ParkerPredictionService extends IedbEpitopePredictionService{

	public ParkerPredictionService(String sequence) {
		// TODO Auto-generated constructor stub
		super(sequence,"Parker");
	}
```

```
}
package epiprot.services.uniprot;


public class PDBchain {
        private String chain;
        private int begin;
        private int end;
        public String getChain() {
                return chain;
        }
        public void setChain(String chain) {
                this.chain = chain;
        }
        public int getBegin() {
                return begin;
        }
        public void setBegin(int begin) {
                this.begin = begin;
        }
        public int getEnd() {
                return end;
        }
```

```
	public void setEnd(int end) {
		this.end = end;
	}

}
package epiprot.services.uniprot;

import java.util.ArrayList;

import org.springframework.util.StringUtils;

public class PDBentry extends DBentry {
	private String method;
	private double resolution;
	private String positions;

	public PDBentry(String id, String type, String method, double resolution, String
positions){
		super(id,type);
		this.method = method;
		this.resolution = resolution;
		this.positions = positions;
	}
```

```
public String getMethod() {
        return method;
}

public void setMethod(String method) {
        this.method = method;
}

public double getResolution() {
        return resolution;
}

public void setResolution(double resolution) {
        this.resolution = resolution;
}

public String getPositions() {
        return positions;
}

public void setPositions(String positions) {
        this.positions = positions;
```

```
}

public ArrayList<PDBchain> getPdbChainList() {
        ArrayList<PDBchain> pdbChainList = new ArrayList<PDBchain>();
        String positions = getPositions();
        int numCommas = StringUtils.countOccurrencesOf(positions, ",");
        int count = 0;
        do {
                PDBchain pdbChain = new PDBchain();
                int equalPos = 0;
        int dashPos = 0;
        int commaPos = 0;

        for (int i = 0; i < positions.length(); i++) {
                if (positions.charAt(i) == '=') {
                        equalPos = i;
                }
                else if (positions.charAt(i) == '-') {
                        dashPos = i;
                }
                else if (positions.charAt(i) == ',') {
                        commaPos = i;
                        break;
```

```
                    }
                }
                    pdbChain.setChain(positions.substring(0,equalPos));

            pdbChain.setBegin(Integer.parseInt(positions.substring(equalPos+1,dashPos)));

            pdbChain.setEnd(Integer.parseInt(positions.substring(dashPos+1,commaPos)));
                    pdbChainList.add(pdbChain);
                    if (commaPos != 0) {
                        positions = positions.substring(commaPos+2);
                    }
                    count++;
                } while (count <= numCommas);
                return pdbChainList;
            }
        }
package epiprot.services.sifts;

import javax.swing.JCheckBox;

public class PdbEntry {

    private String proteinAcc;
private String pdbId;

private double resolution;

private String method;

private String chain;

private JCheckBox checkBox;

private String positions;


public PdbEntry() {

        // TODO Auto-generated constructor stub

}


public String getProteinAcc() {

        return proteinAcc;

}


public void setProteinAcc(String proteinAcc) {

        this.proteinAcc = proteinAcc;

}


public String getPdbId() {

        return pdbId;

}
```

```
public void setPdbId(String pdbId) {
	this.pdbId = pdbId;
}

public double getResolution() {
	return resolution;
}

public void setResolution(double resolution) {
	this.resolution = resolution;
}

public String getMethod() {
	return method;
}

public void setMethod(String method) {
	this.method = method;
}

public String getChain() {
	return chain;
}
```

```
	public void setChain(String chain) {
		this.chain = chain;
	}

	public JCheckBox getCheckBox() {
		return checkBox;
	}

	public void setCheckBox(JCheckBox checkBox) {
		this.checkBox = checkBox;
	}

	public String getPositions() {
		return positions;
	}

	public void setPositions(String positions) {
		this.positions = positions;
	}

}
package epiprot.services.phosphosite;
```

```
import java.util.ArrayList;

import epiprot.AminoAcid;

public class PhosphoSiteAminoAcid extends AminoAcid{

	private boolean isPhosphorylated;
	private boolean isAcetylated;
	private boolean isMethylated;
	private boolean isOGlycosylated;
	private boolean isOGalNAc;
	private boolean isOGlcNAc;
	private boolean isSumoylated;
	private boolean isUbiquitinated;
	private boolean isPTM;

	private String peptide;

	private ArrayList<PhosphoSitePTM> ptmList = new
ArrayList<PhosphoSitePTM>();

	public PhosphoSiteAminoAcid() {
		// TODO Auto-generated constructor stub
	}

	public boolean isAcetylated() {
		return isAcetylated;
	}

	public void setAcetylated(boolean isAcetylated) {
		this.isAcetylated = isAcetylated;
	}

	public boolean isPhosphorylated() {
		return isPhosphorylated;
	}

	public void setPhosphorylated(boolean isPhosphorylated) {
		this.isPhosphorylated = isPhosphorylated;
	}

	public boolean isMethylated() {
		return isMethylated;
	}
```

```
public void setMethylated(boolean isMethylated) {
        this.isMethylated = isMethylated;
}

public boolean isOGlycosylated() {
        return isOGlycosylated;
}

public void setOGlycosylated(boolean isOGlycosylated) {
        this.isOGlycosylated = isOGlycosylated;
}

public boolean isOGalNAc() {
        return isOGalNAc;
}

public void setOGalNAc(boolean isOGalNAc) {
        this.isOGalNAc = isOGalNAc;
}

public boolean isOGlcNAc() {
        return isOGlcNAc;
}
```

```
public void setOGlcNAc(boolean isOGlcNAc) {
        this.isOGlcNAc = isOGlcNAc;
}

public boolean isSumoylated() {
        return isSumoylated;
}

public void setSumoylated(boolean isSumoylated) {
        this.isSumoylated = isSumoylated;
}

public boolean isUbiquitinated() {
        return isUbiquitinated;
}

public void setUbiquitinated(boolean isUbiquitinated) {
        this.isUbiquitinated = isUbiquitinated;
}

public String getPeptide() {
        return peptide;
```

```
}

public void setPeptide(String peptide) {
        this.peptide = peptide;
}

public PhosphoSiteAminoAcid(PhosphoSiteAminoAcid aminoAcidToCopy) {
        this.isAcetylated = aminoAcidToCopy.isAcetylated;
        this.isMethylated = aminoAcidToCopy.isMethylated;
        this.isOGalNAc = aminoAcidToCopy.isOGalNAc;
        this.isOGlcNAc = aminoAcidToCopy.isOGlcNAc;
        this.isOGlycosylated = aminoAcidToCopy.isOGlycosylated;
        this.isPhosphorylated = aminoAcidToCopy.isPhosphorylated;
        this.isSumoylated = aminoAcidToCopy.isSumoylated;
        this.isUbiquitinated = aminoAcidToCopy.isUbiquitinated;
        this.isPTM = aminoAcidToCopy.isPTM;
        this.peptide = aminoAcidToCopy.peptide;
        aminoAcidToCopy.setPosition(super.getPosition());
        aminoAcidToCopy.setResidue(super.getResidue());
}

public boolean isPTM() {
        return isPTM;
```

```
	}

	public void setPTM(boolean isPTM) {
		this.isPTM = isPTM;
	}

	public void addPTM(PhosphoSitePTM phosphoSitePLM) {
		ptmList.add(phosphoSitePLM);
	}

	public ArrayList<PhosphoSitePTM> getPhosphoSitePTMs() {
		return ptmList;
	}

	@Override
	public String toString() {
		String aa = this.getResidue()+"|"+this.getPosition();
		for(PhosphoSitePTM ptm : this.ptmList) {
			aa = aa + "|" + ptm.toString();
		}
		return aa;
	}
}
```

```
package epiprot.services.views;

import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;
import java.util.LinkedHashMap;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JTextField;
import javax.swing.JTextPane;
import javax.swing.event.DocumentEvent;
import javax.swing.event.DocumentListener;

import epiprot.Presenter;
import epiprot.services.phosphosite.PhosphoSiteAminoAcid;
import epiprot.services.phosphosite.PhosphoSitePTM;
import epiprot.services.phosphosite.PhosphoSiteService;
import epiprot.services.uniprot.UniProtAminoAcid;
import epiprot.services.uniprot.UniProtPTM;

public class PhosphoSitePresenter {

	Presenter presenter;
```

```
String phosphoLine = "";

String acetylLine = "";

String methylLine = "";

String sumoLine = "";

String ubiquitLine = "";

String ogalnacLine = "";

String oglcnacLine = "";


interface View {

        JTextField htpPhosphoTextField();

        JTextField ltpPhosphoTextField();

        JTextField htpAcetylTextField();

        JTextField ltpAcetylTextField();

        JTextField htpMethylTextField();

        JTextField ltpMethylTextField();

        JTextField htpOgalnacTextField();

        JTextField ltpOgalnacTextField();

        JTextField htpOglcnacTextField();

        JTextField ltpOglcnacTextField();

        JTextField htpSumoylTextField();

        JTextField ltpSumoylTextField();

        JTextField htpUbiquitinTextField();
```

```
		JTextField ltpUbiquitinTextField();

		JCheckBox phosphoCheckBox();

		JCheckBox acetylCheckBox();

		JCheckBox methylCheckBox();

		JCheckBox ogalnacCheckBox();

		JCheckBox oglcnacCheckBox();

		JCheckBox sumoylCheckBox();

		JCheckBox ubiquitinCheckBox();

		JCheckBox uniprotCheckBox();

		JCheckBox chckbxSelectAll();

		JButton btnSubmit();

	}


	interface SummaryView {

		JTextPane textPane();

		void appendString(String str);

	}


	public PhosphoSitePresenter(Presenter presenter) {

		// TODO Auto-generated constructor stub

		this.presenter = presenter;

		bindHandlers();
```

```
        }


        View view = new PhosphoSiteView();

        SummaryView summaryView = new PhosphoSiteSummaryView();


        public void bindHandlers() {

                view.btnSubmit().addActionListener(new ActionListener() {

                        @Override

                        public void actionPerformed(ActionEvent e) {

                                System.out.println(presenter.proteinAcc);

                                System.out.println(presenter.protein.getSequence());

                                System.out.println(view.phosphoCheckBox().isSelected());


                                String text = String.format("%1$-20s %2$10s %3$10s",
    "PTM", "HTP","LTP");

                                summaryView.appendString(text+"\n");




                                for(int i = 0; i < presenter.protein.getSequence().length();
    i++) {

                                        addDashToAllLines();

                                }

                                PhosphoSiteService phosphoSiteService = new
```

```
PhosphoSiteService(presenter.protein.getSequence(),presenter.proteinAcc,

          view.phosphoCheckBox().isSelected(),view.acetylCheckBox().isSelected(),view.
methylCheckBox().isSelected(),

          view.ogalnacCheckBox().isSelected(),view.oglcnacCheckBox().isSelected(),view
.sumoylCheckBox().isSelected(),
                                                  view.ubiquitinCheckBox().isSelected());

                              phosphoSiteService.run();
                              ArrayList<PhosphoSiteAminoAcid> aaList =
phosphoSiteService.getAminoAcids();

                              for(int i = 0; i < aaList.size(); i++) {
                                        PhosphoSiteAminoAcid aa = aaList.get(i);
                                        if(aa.isPTM() &&
aa.getPhosphoSitePTMs().size()>0) {
                                                  for (PhosphoSitePTM ptm:
aa.getPhosphoSitePTMs()) {
                                                            String ptmSummary =
aa.getResidue()+aa.getPosition()+" ";

          if(ptm.getType().equals(PhosphoSiteService.PHOSPHOYLATED)) {
```

```
                                                            ptmSummary = ptmSummary
+ "-p";
                                                            if
(ptm.getHtpHits()>=Integer.parseInt(view.htpPhosphoTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpPhosphoTextField().getText())){
                                                                  StringBuilder sb =
new StringBuilder(phosphoLine);
                                                                  phosphoLine =
sb.replace(i, i+1, "p").toString();
                                                            }
                                                      }
                                                      else
if(ptm.getType().equals(PhosphoSiteService.ACETYLATED)) {
                                                            ptmSummary = ptmSummary
+ "-a";
                                                            if
(ptm.getHtpHits()>=Integer.parseInt(view.htpAcetylTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpAcetylTextField().getText())) {
                                                                  StringBuilder sb =
new StringBuilder(acetylLine);
                                                                  acetylLine =
sb.replace(i, i+1, "a").toString();
                                                            }
```

```
                                        }
                                        else
if(ptm.getType().equals(PhosphoSiteService.METHYLATED)) {
                                            ptmSummary = ptmSummary
+ "-m";
                                            if
(ptm.getHtpHits()>=Integer.parseInt(view.htpMethylTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpMethylTextField().getText())) {
                                                StringBuilder sb =
new StringBuilder(methylLine);
                                                methylLine =
sb.replace(i, i+1, "m").toString();
                                            }
                                        }
                                        else
if(ptm.getType().equals(PhosphoSiteService.SUMOYLATED)) {
                                            ptmSummary = ptmSummary
+ "-s";

    if(ptm.getHtpHits()>=Integer.parseInt(view.htpSumoylTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpSumoylTextField().getText())) {
                                                StringBuilder sb =
new StringBuilder(sumoLine);
```

```
                                                                sumoLine =
sb.replace(i, i+1, "s").toString();
                                                        }
                                                }
                                                else
if(ptm.getType().equals(PhosphoSiteService.UBIQUITINATED)) {
                                                        ptmSummary = ptmSummary
+ "-u";
                                                        if
(ptm.getHtpHits()>=Integer.parseInt(view.htpUbiquitinTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpUbiquitinTextField().getText())) {
                                                                StringBuilder sb =
new StringBuilder(ubiquitLine);
                                                                ubiquitLine =
sb.replace(i, i+1, "u").toString();
                                                        }
                                                }
                                                else
if(ptm.getType().equals(PhosphoSiteService.OGALNAC)) {
                                                        ptmSummary = ptmSummary
+ "-O gal";

        if(ptm.getHtpHits()>=Integer.parseInt(view.htpOgalnacTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpOglcnacTextField().getText())) {
                                                            StringBuilder sb =
new StringBuilder(ogalnacLine);
                                                            ogalnacLine =
sb.replace(i, i+1, "o").toString();
                                                     }
                                              }
                                              else
if(ptm.getType().equals(PhosphoSiteService.OGLCNAC)) {
                                                     ptmSummary = ptmSummary
+ "-O glc";

       if(ptm.getHtpHits()>=Integer.parseInt(view.htpOglcnacTextField().getText()) ||
ptm.getLtpHits()>=Integer.parseInt(view.ltpOglcnacTextField().getText())) {
                                                            StringBuilder sb =
new StringBuilder(oglcnacLine);
                                                            oglcnacLine =
sb.replace(i, i+1, String.valueOf('\u00F6')).toString();
                                                     }
                                              }

                                              //ptmSummary = ptmSummary + ":
HTP:" + ptm.getHtpHits() + " LTP:" + ptm.getLtpHits()+"\n";
```

```
                                                text = String.format("%1$-20s
%2$10s %3$10s", ptmSummary+":", ptm.getHtpHits(),ptm.getLtpHits());

     summaryView.appendString(text+"\n");
                                           }

                                      }
                                 }



                                 if (phosphoLine.contains("p")) {

     presenter.insertLineAboveTarget("Phosphorylations",
getInsertLine(phosphoLine));
                                 }
                                 if (acetylLine.contains("a")) {
                                      presenter.insertLineAboveTarget("Acetylations",
getInsertLine(acetylLine));
                                 }
                                 if (methylLine.contains("m")) {
                                      presenter.insertLineAboveTarget("Methylations",
getInsertLine(methylLine));
                                 }
```

```
				if (sumoLine.contains("s")) {
					presenter.insertLineAboveTarget("Sumoylations",
getInsertLine(sumoLine));
				}
				if (ubiquitLine.contains("u")) {
					presenter.insertLineAboveTarget("Ubiquitinations",
getInsertLine(ubiquitLine));
				}
				if (ogalnacLine.contains("o")) {
					presenter.insertLineAboveTarget("O-GalNAc",
getInsertLine(ogalnacLine));
				}
				if (oglcnacLine.contains("\u00F6")) {
					presenter.insertLineAboveTarget("O-GlcNAc",
getInsertLine(oglcnacLine));
				}

				//uniprot ptms
				if(view.uniprotCheckBox().isSelected()) {
					summaryView.appendString("\nUniProt PTMs:\n");
					ArrayList<UniProtPTM> ptmList =
presenter.protein.getPTMs();
					ArrayList<UniProtAminoAcid> uniAAlist =
getAminoAcids();
                                        System.out.println("&&&"+ptmList.size());
                                        LinkedHashMap<Integer,Integer> map = new
LinkedHashMap<Integer,Integer>();
                                        LinkedHashMap<Integer,UniProtAminoAcid>
ptmMap = new LinkedHashMap<Integer,UniProtAminoAcid>();
                                        int disulfideNumber = 0;
                                        for(UniProtPTM ptm: ptmList) {
                                             if (ptm.getPosition() != 0) {

     uniAAlist.get(ptm.getPosition()).addPTM(ptm);
                                             }
                                             else {
                                                  disulfideNumber++;

     ptm.setDisulfideNumber(disulfideNumber);

     uniAAlist.get(ptm.getBeginPosition()).addPTM(ptm);

     uniAAlist.get(ptm.getEndPosition()).addPTM(ptm);
                                             }
                                        }
                                        //make enough lines for ptms
                                        int numberOfLines = 0;
                                        for(int i = 0; i < uniAAlist.size(); i++) {

        if(uniAAlist.get(i).getUniProtPTMs().size()>numberOfLines) {
                                                        numberOfLines =
uniAAlist.get(i).getUniProtPTMs().size();
                                                }
                                        }



        System.out.println("numberOfLines"+numberOfLines);


                                        String line = getUniProtLine();
                                        ArrayList<String> lines = new ArrayList<String>();
                                        for(int i = 0; i < numberOfLines; i++) {
                                                String s = new String(line);
                                                lines.add(s);
                                        }



                                        for(int i = 0; i < uniAAlist.size(); i++) {
                                                UniProtAminoAcid aa = uniAAlist.get(i);
                                        ArrayList<UniProtPTM> aaPTMs =
aa.getUniProtPTMs();

    System.out.println("@@@"+aaPTMs.size());
                                        for(int j = 0;  j < aaPTMs.size(); j++) {
                                            UniProtPTM ptm = aaPTMs.get(j);
                                            String ptmSummary = "";

                                            String insertLine = lines.get(j);
                                            StringBuilder sb = new
StringBuilder(insertLine);

                                            //replace amino acid
                                            if
(ptm.getUniprotType().toLowerCase().equals("modified residue")) {
                                                ptmSummary =
presenter.protein.getSequence().charAt(ptm.getPosition()-1)+""+ptm.getPosition();

    if(ptm.getPtmType().equals(UniProtPTM.ACETYLATED)) {
                                                    insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('a')).toString();
                                                    ptmSummary =
ptmSummary + "-a";
```

```
                                                        }
                                                        else
if(ptm.getPtmType().equals(UniProtPTM.AMIDATION)) {
                                                             insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('d')).toString();
                                                             ptmSummary =
ptmSummary + "-d";
                                                        }
                                                        else
if(ptm.getPtmType().equals(UniProtPTM.FLAVIN)) {
                                                             insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('f')).toString();
                                                             ptmSummary =
ptmSummary + "-f";
                                                        }
                                                        else
if(ptm.getPtmType().equals(UniProtPTM.HYDROXYLATION)) {
                                                             insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('h')).toString();
                                                             ptmSummary =
ptmSummary + "-h";
                                                        }
                                                        else
```

```
if(ptm.getPtmType().equals(UniProtPTM.ISOMERIZATION)) {
                                                            insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('i')).toString();
                                                            ptmSummary =
ptmSummary + "-i";
                                                       }
                                                       else
if(ptm.getPtmType().equals(UniProtPTM.METHYLATED)) {
                                                            insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('m')).toString();
                                                            ptmSummary =
ptmSummary + "-m";
                                                       }
                                                       else
if(ptm.getPtmType().equals(UniProtPTM.PHOSPHOYLATED)) {
                                                            insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('p')).toString();
                                                            ptmSummary =
ptmSummary + "-p";
                                                       }
                                                       else
if(ptm.getPtmType().equals(UniProtPTM.PYRROLIDONE)) {
                                                            insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('z')).toString();
                                                                        ptmSummary =
ptmSummary + "-z";
                                                               }
                                                               else
if(ptm.getPtmType().equals(UniProtPTM.SULFATION)) {
                                                                        insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('$')).toString();
                                                                        ptmSummary =
ptmSummary + "-$";
                                                               }
                                                               else
if(ptm.getPtmType().equals(UniProtPTM.SUMOYLATED)) {
                                                                        insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('s')).toString();
                                                                        ptmSummary =
ptmSummary + "-s";
                                                               }
                                                               else
if(ptm.getPtmType().equals(UniProtPTM.UBIQUITINATED)) {
                                                                        insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('u')).toString();
                                                                        ptmSummary =
ptmSummary + "-u";
                                                                    }
                                                                    ptmSummary = ptmSummary
+": "+ ptm.getDescription();

        summaryView.appendString(ptmSummary+"\n");
                                                            }
                                                            else
if(ptm.getUniprotType().toLowerCase().equals("glycosylation site")) {
                                                                    ptmSummary =
presenter.protein.getSequence().charAt(ptm.getPosition()-1)+""+ptm.getPosition();
                                                                    System.out.println("!@#" +
ptm.getPtmType());

        if(ptm.getPtmType().equals(UniProtPTM.GLYCOSYLATED)) {
                                                                            insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('g')).toString();
                                                                            ptmSummary =
ptmSummary + "-g";
                                                                    }
                                                                    else
if(ptm.getLinkage().equals(UniProtPTM.OLINKED) &&
ptm.getPtmType().equals(UniProtPTM.GALNAC)) {
```

```
                                                                    insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('o')).toString();
                                                                    ptmSummary =
ptmSummary + "-O gal";
                                                              }
                                                              else
if(ptm.getLinkage().equals(UniProtPTM.OLINKED) &&
ptm.getPtmType().equals(UniProtPTM.GLCNAC)) {
                                                                    insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('\u00F6')).toString();
                                                                    ptmSummary =
ptmSummary + "-O glc";
                                                              }
                                                              else
if(ptm.getLinkage().equals(UniProtPTM.NLINKED) &&
ptm.getPtmType().equals(UniProtPTM.GALNAC)) {
                                                                    insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('n')).toString();
                                                                    ptmSummary =
ptmSummary + "-N gal";
                                                              }
                                                              else
if(ptm.getLinkage().equals(UniProtPTM.NLINKED) &&
ptm.getPtmType().equals(UniProtPTM.GLCNAC)) {
                                                                        insertLine =
sb.replace(ptm.getPosition()-1, ptm.getPosition(), String.valueOf('\u00F1')).toString();
                                                                        ptmSummary =
ptmSummary + "-N glc";
                                                                }
                                                                ptmSummary = ptmSummary
+": "+ ptm.getDescription();

        summaryView.appendString(ptmSummary+"\n");
                                                        }
                                                        else
if(ptm.getUniprotType().toLowerCase().equals("disulfide bond") &&
insertLine.charAt(ptm.getBeginPosition()-1)=='-') {
                                                                insertLine =
sb.replace(ptm.getBeginPosition()-1, ptm.getBeginPosition(),
String.valueOf(ptm.getDisulfideNumber())).toString();
                                                                insertLine =
sb.replace(ptm.getEndPosition()-1, ptm.getEndPosition(),
String.valueOf(ptm.getDisulfideNumber())).toString();
                                                                ptmSummary =
presenter.protein.getSequence().charAt(ptm.getBeginPosition()-
1)+""+ptm.getBeginPosition() + "---" +
presenter.protein.getSequence().charAt(ptm.getEndPosition()-1) +
ptm.getEndPosition()+": "+ "disulfide bond";

          summaryView.appendString(ptmSummary+"\n");
                                                            }
                                                            lines.set(j, insertLine);
                                                  }
                                        }
                                        for(int i = lines.size()-1; i >= 0 ; i--) {
                                                  int lineNumber = i +1;
                                                  presenter.insertLineAboveTarget("UniProt
PTM Line "+lineNumber, getInsertLine(lines.get(i)));
                                        }
                              }
                    }
          });

          view.htpPhosphoTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpPhosphoTextField()));
          view.ltpPhosphoTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpPhosphoTextField()));

          view.htpAcetylTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpAcetylTextField()));
                view.ltpAcetylTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpAcetylTextField()));


                view.htpMethylTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpMethylTextField()));
                view.ltpMethylTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpMethylTextField()));


                view.htpOgalnacTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpOgalnacTextField()));
                view.ltpOgalnacTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpOgalnacTextField()));


                view.htpOglcnacTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpOglcnacTextField()));
                view.ltpOglcnacTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpOglcnacTextField()));


                view.htpSumoylTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpSumoylTextField()));
                view.ltpSumoylTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpSumoylTextField()));
```

```
		view.htpUbiquitinTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.htpUbiquitinTextField()));
		view.ltpUbiquitinTextField().getDocument().addDocumentListener(new
SubmitButtonListener(view.ltpUbiquitinTextField()));


		view.acetylCheckBox().addActionListener(new
CheckBoxListener(view));
		view.phosphoCheckBox().addActionListener(new
CheckBoxListener(view));
		view.methylCheckBox().addActionListener(new
CheckBoxListener(view));
		view.ogalnacCheckBox().addActionListener(new
CheckBoxListener(view));
		view.oglcnacCheckBox().addActionListener(new
CheckBoxListener(view));
		view.sumoylCheckBox().addActionListener(new
CheckBoxListener(view));
		view.ubiquitinCheckBox().addActionListener(new
CheckBoxListener(view));
		view.uniprotCheckBox().addActionListener(new
CheckBoxListener(view));
```

```
view.chckbxSelectAll().addActionListener(new ActionListener() {

        @Override
        public void actionPerformed(ActionEvent e) {
                // TODO Auto-generated method stub
                if (view.chckbxSelectAll().isSelected()) {
                        view.phosphoCheckBox().setSelected(true);
                        view.acetylCheckBox().setSelected(true);
                        view.methylCheckBox().setSelected(true);
                        view.ogalnacCheckBox().setSelected(true);
                        view.oglcnacCheckBox().setSelected(true);
                        view.sumoylCheckBox().setSelected(true);
                        view.ubiquitinCheckBox().setSelected(true);
                        view.uniprotCheckBox().setSelected(true);
                        view.btnSubmit().setEnabled(true);
                }
                else {
                        view.phosphoCheckBox().setSelected(false);
                        view.acetylCheckBox().setSelected(false);
                        view.methylCheckBox().setSelected(false);
                        view.ogalnacCheckBox().setSelected(false);
                        view.oglcnacCheckBox().setSelected(false);
                        view.sumoylCheckBox().setSelected(false);
                        view.ubiquitinCheckBox().setSelected(false);
                        view.uniprotCheckBox().setSelected(false);
                        view.btnSubmit().setEnabled(false);
                    }
                }

            });
        }

        private String getUniProtLine() {
            String line = "";
            for(int i = 0; i < presenter.protein.getSequence().length(); i++) {
                line = line + "-";
            }
            return line;
        }

        private void addDashToAllLines() {
            oglcnacLine = oglcnacLine + "-";
            phosphoLine = phosphoLine + "-";
            acetylLine = acetylLine + "-";
            methylLine = methylLine + "-";
            sumoLine = sumoLine + "-";
            ubiquitLine = ubiquitLine + "-";
            ogalnacLine = ogalnacLine + "-";
}

private boolean isInteger(String s) {
            try {
        Integer.parseInt(s);
    } catch(NumberFormatException e) {
        return false;
    } catch(NullPointerException e) {
        return false;
    }
    // only got here if we didn't return false
    return true;
}

public void checkTextField(String text) {
            if(isInteger(text) && Integer.parseInt(text) >= 0) {
                        view.btnSubmit().setEnabled(true);
            }
            else {
                        view.btnSubmit().setEnabled(false);
            }
```

```
	}

	public String getInsertLine(String inputLine) {
		String mainLine = presenter.getMainLine();
		for(int i = 0; i < mainLine.length(); i++) {
			if(mainLine.charAt(i) == '-') {
				inputLine = new StringBuilder(inputLine).insert(i, "
").toString();
			}
		}
		return inputLine;
	}

	public ArrayList<UniProtAminoAcid> getAminoAcids() {
		ArrayList<UniProtAminoAcid> aaList = new
ArrayList<UniProtAminoAcid>();
		for (int i = 0; i< presenter.protein.getSequence().length(); i++) {
			aaList.add(new UniProtAminoAcid());
		}
		return aaList;
	}

	public class SubmitButtonListener implements DocumentListener {
```

```
JTextField textField;

public SubmitButtonListener(JTextField textField){
	this.textField = textField;
}

@Override
public void insertUpdate(DocumentEvent e) {
	// TODO Auto-generated method stub
	checkTextField(textField.getText());
}

@Override
public void removeUpdate(DocumentEvent e) {
	// TODO Auto-generated method stub
	checkTextField(textField.getText());
}

@Override
public void changedUpdate(DocumentEvent e) {
	// TODO Auto-generated method stub
	checkTextField(textField.getText());
```

```
		}

	}

	public class CheckBoxListener implements ActionListener {

		View phosphoSiteView;

		public CheckBoxListener(View phosphoSiteView) {
			this.phosphoSiteView = phosphoSiteView;
		}

		@Override
		public void actionPerformed(ActionEvent e) {
			// TODO Auto-generated method stub
			if (phosphoSiteView.acetylCheckBox().isSelected() ||
phosphoSiteView.methylCheckBox().isSelected() ||
phosphoSiteView.phosphoCheckBox().isSelected() ||
					phosphoSiteView.ogalnacCheckBox().isSelected() ||
phosphoSiteView.oglcnacCheckBox().isSelected() ||
					phosphoSiteView.sumoylCheckBox().isSelected() ||
phosphoSiteView.ubiquitinCheckBox().isSelected() ||
phosphoSiteView.uniprotCheckBox().isSelected()) {
```

```
                    view.btnSubmit().setEnabled(true);
                }
                else {
                    view.btnSubmit().setEnabled(false);
                }
            }

        }

}
package epiprot.services.phosphosite;

public class PhosphoSitePTM {

        private String type;
        private String residueModification;
        private int ltpHits;
        private int htpHits;
        private int cstHtpHits;
        private int litHtpHits;

        public PhosphoSitePTM() {
                // TODO Auto-generated constructor stub
```

```
}

public String getType() {
	return type;
}

public void setType(String type) {
	this.type = type;
}

public int getLtpHits() {
	return ltpHits;
}

public void setLtpHits(int ltpHits) {
	this.ltpHits = ltpHits;
}

public int getHtpHits() {
	return htpHits;
}

public void setHtpHits(int htpHits) {
		this.htpHits = htpHits;
	}

	public int getCstHtpHits() {
		return cstHtpHits;
	}

	public void setCstHtpHits(int cstHtpHits) {
		this.cstHtpHits = cstHtpHits;
	}

	public int getLitHtpHits() {
		return litHtpHits;
	}

	public void setLitHtpHits(int litHtpHits) {
		this.litHtpHits = litHtpHits;
	}

	public String getResidueModification() {
		return residueModification;
	}
	public void setResidueModification(String residueModification) {
		this.residueModification = residueModification;
	}

	@Override
	public String toString() {
		return this.type+":h"+this.htpHits+":l"+this.ltpHits;
	}

}
package epiprot.services.phosphosite;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.FileReader;
import java.io.IOException;
import java.net.URL;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.zip.GZIPInputStream;
import epiprot.Protein;
import epiprot.services.Service;


public class PhosphoSiteService extends Service {


	//Where the phosphosite data is
	private static final String INPUT_GZIP_FILE =
"http://www.phosphosite.org/downloads/";


	//PTM constants
	public static final String PHOSPHOYLATED = "phosphoylated";
	public static final String ACETYLATED = "acetylated";
	public static final String METHYLATED = "methylated";
	public static final String OGALNAC = "o-galnac";
	public static final String OGLCNAC = "o-glcnac";
	public static final String SUMOYLATED = "sumoylated";
	public static final String UBIQUITINATED = "ubiquitinated";


	public String sequence;
	public String uniprotId;
	public boolean getPhosphorylation;
	public boolean getAcetylation;
	public boolean getMethylation;
	public boolean getOGalNAc;
	public boolean getOGlcNAc;
	public boolean getSumoylation;
	public boolean getUbiquitination;
	public ArrayList<PhosphoSiteAminoAcid> aaList = new
ArrayList<PhosphoSiteAminoAcid>();

	public File dataFile;

	//constructor that sets all the PTMs to true
	public PhosphoSiteService(String sequence, String uniprotId) {
		// TODO Auto-generated constructor stub
		this.sequence = sequence;
		this.uniprotId = uniprotId;
		for (int i = 0; i < sequence.length(); i++) {
			PhosphoSiteAminoAcid aa = new PhosphoSiteAminoAcid();
			aa.setPTM(false);
			aaList.add(aa);
	}
		this.getAcetylation = true;
		this.getMethylation = true;
		this.getOGalNAc = true;
		this.getOGlcNAc = true;
		this.getPhosphorylation = true;
		this.getSumoylation = true;
		this.getUbiquitination = true;
	}

	public PhosphoSiteService(String sequence, String uniprotId,boolean
getPhosphorylation, boolean getAcetylation, boolean getMethylation, boolean
getOGalNAc, boolean getOGlcNAc, boolean getSumoylation, boolean
getUbiquitination) {
		this.sequence = sequence;
		this.uniprotId = uniprotId;
		for (int i = 0; i < sequence.length(); i++) {
			aaList.add(new PhosphoSiteAminoAcid());
	}
		this.getAcetylation = getAcetylation;
		this.getMethylation = getMethylation;
		this.getOGalNAc = getOGalNAc;
		this.getOGlcNAc = getOGlcNAc;
		this.getPhosphorylation = getPhosphorylation;
		this.getSumoylation = getSumoylation;
		this.getUbiquitination = getUbiquitination;
	}
```

```
        public File getDatabaseTextFile(String databaseString) {
                byte[] buffer = new byte[1024];
                String databaseStringTxt = databaseString.replace("gz", "txt");
          try{
                GZIPInputStream gzis = new GZIPInputStream(new
URL(INPUT_GZIP_FILE+databaseString).openStream());
                FileOutputStream out = new
FileOutputStream(SERVICEFILEPATH+databaseStringTxt);

              int len;
              while ((len = gzis.read(buffer)) > 0) {
                out.write(buffer, 0, len);
              }
              gzis.close();
                out.close();
          } catch(IOException ex) {
             ex.printStackTrace();
          }

                return new File(SERVICEFILEPATH+databaseStringTxt);
        }

        @Override
```

```
	public void run() {
		if (getPhosphorylation) {
			dataFile =
getDatabaseTextFile("Phosphorylation_site_dataset.gz");
			setAminoAcids(dataFile,1);
			//System.out.println(PHOSPHOYLATED);
		}
		if (getAcetylation) {
			dataFile = getDatabaseTextFile("Acetylation_site_dataset.gz");
			setAminoAcids(dataFile,2);
			//System.out.println(ACETYLATED);
		}
		if (getMethylation) {
			dataFile = getDatabaseTextFile("Methylation_site_dataset.gz");
			setAminoAcids(dataFile,3);
			//System.out.println(METHYLATED);
		}
		if (getOGalNAc) {
			dataFile = getDatabaseTextFile("O-GalNAc_site_dataset.gz");
			setAminoAcids(dataFile,4);
			//System.out.println(OGALNAC);
		}
		if(getOGlcNAc) {
```

```
                dataFile = getDatabaseTextFile("O-GlcNAc_site_dataset.gz");
                setAminoAcids(dataFile,5);
                //System.out.println(OGLCNAC);
            }
            if(getSumoylation) {
                dataFile = getDatabaseTextFile("Sumoylation_site_dataset.gz");
                setAminoAcids(dataFile,6);
                //System.out.println(SUMOYLATED);
            }
            if(getUbiquitination) {
                dataFile = getDatabaseTextFile("Ubiquitination_site_dataset.gz");
                setAminoAcids(dataFile,7);
                //System.out.println(UBIQUITINATED);
            }
        }

        /**
    * setAminoAcids()
    *
    * Each line is split into an array which is placed in an arraylist
    * array information:
    * 0: Protein name
    * 1: Accession id
```

```
     * 2: Gene
     * 3: Human chromosome location
     * 4: Modified residue
     * 5: Site group id <-- very important in order to locate equivalent location in mouse
and rat
     * 6: organism
     * 7: molecular weight
     * 8: Domain
     * 9: Site peptide <-- very important, matches site to protein sequence
     * 10: Low Throughput Literature (ltp) hits
     * 11: High Throughput Literature (htp1) hits
     * 12: High Throughput CST (htp2) hits
     * 13: CST catalog #
     */
    public void setAminoAcids(File file, int ptmType) {

                //get each line as an String[] separated by tab
                ArrayList <String[]> lineList = getLines(file,uniprotId,uniprotId+"-");

                /*
                 * Once we get all the amino acids with phosphosites, we
                 * can now make an AminoAcid, fill in the fields and transfer the
                 * information to the ArrayList<AminoAcid> list at the proper index.
```

```
 * If the site peptide is not found in protein sequence, add AminoAcid
 * to end of ArrayList<AminoAcid> list. MuscleAlignment later checks
 * if the Phosphosite ArrayList is longer in length then the protein seq.
 * If so it will note the non-matched AminoAcid(s) at the end of the
 * alignment file.
 */
for (String[] array : lineList) {
        String peptideSeq = array[9].toUpperCase();
        int dashCount = 0;
        loop:
        for (int i = 0; i < peptideSeq.length(); i++) {
                if (Character.isLetter(peptideSeq.charAt(i))) {
                        break loop;
                }
                dashCount++;
        }
        peptideSeq = peptideSeq.replaceAll("_", "");

        //get the position of the phosphosite to match site to arraylist
        int sitePosition = sequence.indexOf(peptideSeq)+7-dashCount;
        if (sitePosition != -1) {
                PhosphoSiteAminoAcid aa = aaList.get(sitePosition);
                aa.setResidue(sequence.substring(sitePosition,
sitePosition+1));
                                        aa.setPosition(sitePosition+1);
                                        PhosphoSitePTM ptm = new PhosphoSitePTM();

                                        switch(ptmType) {
                                                case 1: aa.setPhosphorylated(true);
ptm.setType(PHOSPHOYLATED);break;
                                                case 2: aa.setAcetylated(true);
ptm.setType(ACETYLATED);break;
                                                case 3: aa.setMethylated(true);
ptm.setType(METHYLATED);break;
                                                case 4: aa.setOGalNAc(true);
aa.setOGlycosylated(true); ptm.setType(OGALNAC); break;
                                                case 5: aa.setOGlcNAc(true);
aa.setOGlycosylated(true); ptm.setType(OGLCNAC);break;
                                                case 6: aa.setSumoylated(true);
ptm.setType(SUMOYLATED);break;
                                                case 7: aa.setUbiquitinated(true);
ptm.setType(UBIQUITINATED);break;
                                        }
                                        aa.setPTM(true);
                                        //set phospho peptide
                                        aa.setPeptide(array[9]);
```

```
                    /*
                     * An amino acid can potentially have more than one PTM.
                     * In order to account for this PhosphoSiteService creates
                     * a PhosphoSitePTM, places the info in the ptm object,
and
                     * that is added to the PhosphoSiteAminoAcid.
                     */

                    //set ltp hits
                    int ltpHits = getTPNumber(array,10);
                    //set htp hits, have to check lit and CST hits separately and
add them together
                    int litHtpHits = getTPNumber(array,11);
                    int cstHtphits = getTPNumber(array,12);

                    /*
                     * The number of ltp and htp hits for a site on PhosphoSite
includes hits
                     * in other similar proteins. For example, human and mouse
Akt1 are very
                     * similar. A phospho-site throughput numbers on
PhosphoSite combines the
```

```
                         * throughput numbers for human, mouse, rat (and other
species if available).
                         * An unique id, "Site group id", is associated with every
site. Therefore
                         * to get all the hit numbers we have to look for the id
number and gather
                         * the results.
                         */

                        //get the id number for the site
                        String siteId = array[5];

                        //get all the lines with that site id, but not the current target
protein sites
                        ArrayList <String[]> siteList =
getLines(file,siteId,uniprotId);

                        //add in hits for that site id
                        for (String[] siteArray : siteList) {
                                ltpHits = ltpHits + getTPNumber(siteArray,10);
                                litHtpHits = litHtpHits +
getTPNumber(siteArray,11);
                                cstHtphits = cstHtphits +
getTPNumber(siteArray,12);
                                        }

                                        //set Through put numbers
                                        ptm.setHtpHits(litHtpHits+cstHtphits);
                                        ptm.setLtpHits(ltpHits);

                                        //set residue string info
                                        ptm.setResidueModification(array[4]);

                                        aa.addPTM(ptm);
                              }
                    }
   }

   /*
    * helper method for getPhosphosites()
    * reads the file and returns the lines containing the given string
    * as an ArrayList <String[]> where each String [] is a line
    * split
    */
   private ArrayList<String []> getLines(File file, String match, String notMatch) {
          ArrayList <String[]> lineList = new ArrayList<String[]>();
```

```
		try {
				BufferedReader br = new BufferedReader(new FileReader (file));
				String line = null;
				while ((line = br.readLine()) != null) {
					String [] array = line.split("\t");
					for (int i = 0; i < array.length; i++) {
						if (array[i].equals("")) {
							array[i] = 0+"";
						}
					}
					if (Arrays.asList(array).contains(match)) {
						if (!Arrays.asList(array).contains(notMatch)) {
							lineList.add(array);
						}
						else if (Arrays.asList(array).contains(notMatch+"-
")) {
							lineList.add(array);
						}
					}
				}
				br.close();
			} catch (FileNotFoundException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			} catch (IOException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
		return lineList;
	}

	/*
	 * returns the throughput number (high and low) for a given site
	 * index will equal 10 for low throughput
	 * 11 =  literature high throughput
	 * 12 = CST high throughput
	 */
	private int getTPNumber(String[]array, int index) {
		int tpNumber = 0;
		if (array.length > index && isNumeric(array[index])) {
			tpNumber = Integer.parseInt(array[index]);
			}
		return tpNumber;
	}

	//checks to see if string is a number.
```

```
    public boolean isNumeric(String s) {
            try {
                    double d = Double.parseDouble(s);
            } catch(NumberFormatException nfe) {
                    return false;
                    }
            return true;
    }

    public ArrayList<PhosphoSiteAminoAcid> getAminoAcids() {
            return aaList;
    }

    public static void main (String[]args) {
            Protein protein = new Protein("P31749", true);
            PhosphoSiteService service = new
PhosphoSiteService(protein.getSequence(),protein.acc);
            service.run();
            ArrayList<PhosphoSiteAminoAcid> aaList = service.getAminoAcids();
            for(PhosphoSiteAminoAcid aa: aaList) {
                    System.out.println(aa.toString());
            }
    }
```

```
}

package epiprot.services.views;


import javax.swing.JFrame;

import javax.swing.JScrollPane;

import java.awt.BorderLayout;

import java.awt.Dimension;


import javax.swing.JTextPane;

import javax.swing.text.BadLocationException;

import javax.swing.text.StyledDocument;


public class PhosphoSiteSummaryView extends JFrame implements
PhosphoSitePresenter.SummaryView {


        private JTextPane textPane;


        public PhosphoSiteSummaryView() {


                setTitle("PhosphoSite Summary");

                setSize(new Dimension(400, 600));


                JScrollPane scrollPane = new JScrollPane();
```

```
		getContentPane().add(scrollPane, BorderLayout.CENTER);


		textPane = new JTextPane();

		scrollPane.setViewportView(textPane);


		setVisible(true);

	}


	@Override

	public JTextPane textPane() {

		// TODO Auto-generated method stub

		return textPane;

	}


	@Override

	public void appendString(String str) {

		StyledDocument document = (StyledDocument) textPane.getDocument();

	   try {

			document.insertString(document.getLength(), str, null);

		} catch (BadLocationException e) {

			// TODO Auto-generated catch block

			e.printStackTrace();

		}
	}
}
package epiprot.services.views;


import javax.swing.JFrame;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.border.TitledBorder;


import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.GridBagConstraints;
import java.awt.GridBagLayout;
import java.awt.Insets;
import javax.swing.JLabel;
import javax.swing.JTextField;
import javax.swing.JCheckBox;
import javax.swing.JButton;


public class PhosphoSiteView extends JFrame implements PhosphoSitePresenter.View {
	private JTextField htpPhosphoTextField;
	private JTextField ltpPhosphoTextField;
	private JTextField htpAcetylTextField;
private JTextField ltpAcetylTextField;

private JTextField htpMethylTextField;

private JTextField ltpMethylTextField;

private JTextField htpOgalnacTextField;

private JTextField ltpOgalnacTextField;

private JTextField htpOglcnacTextField;

private JTextField ltpOglcnacTextField;

private JTextField htpSumoylTextField;

private JTextField ltpSumoylTextField;

private JTextField htpUbiquitinTextField;

private JTextField ltpUbiquitinTextField;

private JCheckBox phosphoCheckBox;

private JCheckBox acetylCheckBox;

private JCheckBox methylCheckBox;

private JCheckBox ogalnacCheckBox;

private JCheckBox oglcnacCheckBox;

private JCheckBox sumoylCheckBox;

private JCheckBox ubiquitinCheckBox;

private JCheckBox uniprotCheckBox;

private JCheckBox chckbxSelectAll;

private JButton btnSubmit;


public PhosphoSiteView() {
		setTitle("Post-Translational Modifications");
		// TODO Auto-generated constructor stub
		setSize(new Dimension(500, 600));

		JScrollPane scrollPane = new JScrollPane();
		getContentPane().add(scrollPane, BorderLayout.CENTER);

		JPanel panel = new JPanel();
		scrollPane.setViewportView(panel);
		GridBagLayout gbl_panel = new GridBagLayout();
		gbl_panel.rowWeights = new double[]{1.0, 0.0, 0.0, 0.0, 0.0, 0.0, 0.0, 1.0,
0.0, 0.0};
		gbl_panel.columnWeights = new double[]{1.0};
		panel.setLayout(gbl_panel);

		JPanel panel_1 = new JPanel();
		GridBagConstraints gbc_panel_1 = new GridBagConstraints();
		gbc_panel_1.insets = new Insets(5, 5, 5, 0);
		gbc_panel_1.fill = GridBagConstraints.BOTH;
		gbc_panel_1.weightx = 1.0;
		gbc_panel_1.weighty = 1.0;
		gbc_panel_1.gridx = 0;
		gbc_panel_1.gridy = 0;
```

```
        panel.add(panel_1, gbc_panel_1);
        TitledBorder phosphoTitle = new TitledBorder("Phosphorylation");
        panel_1.setBorder(phosphoTitle);
        GridBagLayout gbl_panel_1 = new GridBagLayout();
        gbl_panel_1.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
        gbl_panel_1.rowHeights = new int[]{0, 0};
        gbl_panel_1.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
        gbl_panel_1.rowWeights = new double[]{0.0, Double.MIN_VALUE};
        panel_1.setLayout(gbl_panel_1);

        JLabel lblHtp = new JLabel("HTP");
        GridBagConstraints gbc_lblHtp = new GridBagConstraints();
        gbc_lblHtp.fill = GridBagConstraints.BOTH;
        gbc_lblHtp.insets = new Insets(0, 0, 5, 5);
        gbc_lblHtp.gridx = 0;
        gbc_lblHtp.gridy = 0;
        panel_1.add(lblHtp, gbc_lblHtp);

        htpPhosphoTextField = new JTextField();
        GridBagConstraints gbc_textField = new GridBagConstraints();
        gbc_textField.insets = new Insets(0, 0, 5, 5);
        gbc_textField.fill = GridBagConstraints.BOTH;
        gbc_textField.gridx = 1;
        gbc_textField.gridy = 0;
        panel_1.add(htpPhosphoTextField, gbc_textField);
        htpPhosphoTextField.setColumns(10);

        JLabel lblLtp = new JLabel("LTP");
        GridBagConstraints gbc_lblLtp = new GridBagConstraints();
        gbc_lblLtp.fill = GridBagConstraints.BOTH;
        gbc_lblLtp.insets = new Insets(0, 0, 5, 5);
        gbc_lblLtp.anchor = GridBagConstraints.EAST;
        gbc_lblLtp.gridx = 2;
        gbc_lblLtp.gridy = 0;
        panel_1.add(lblLtp, gbc_lblLtp);

        ltpPhosphoTextField = new JTextField();
        GridBagConstraints gbc_textField_1 = new GridBagConstraints();
        gbc_textField_1.insets = new Insets(0, 0, 5, 5);
        gbc_textField_1.fill = GridBagConstraints.BOTH;
        gbc_textField_1.gridx = 3;
        gbc_textField_1.gridy = 0;
        panel_1.add(ltpPhosphoTextField, gbc_textField_1);
        ltpPhosphoTextField.setColumns(10);
```

```
phosphoCheckBox = new JCheckBox("");

GridBagConstraints gbc_checkBox = new GridBagConstraints();

gbc_checkBox.insets = new Insets(0, 0, 5, 0);

gbc_checkBox.fill = GridBagConstraints.BOTH;

gbc_checkBox.gridx = 4;

gbc_checkBox.gridy = 0;

panel_1.add(phosphoCheckBox, gbc_checkBox);


JPanel panel_2 = new JPanel();

GridBagConstraints gbc_panel_2 = new GridBagConstraints();

gbc_panel_2.insets = new Insets(5, 5, 5, 0);

gbc_panel_2.fill = GridBagConstraints.BOTH;

gbc_panel_2.weightx = 1.0;

gbc_panel_2.weighty = 1.0;

gbc_panel_2.gridx = 0;

gbc_panel_2.gridy = 1;

panel.add(panel_2, gbc_panel_2);

TitledBorder acetylationTitle = new TitledBorder("Acetylation");

panel_2.setBorder(acetylationTitle);

GridBagLayout gbl_panel_2 = new GridBagLayout();

gbl_panel_2.columnWidths = new int[]{0, 0, 0, 0, 0, 0};

gbl_panel_2.rowHeights = new int[]{0, 0};

gbl_panel_2.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
		gbl_panel_2.rowWeights = new double[]{0.0, Double.MIN_VALUE};
		panel_2.setLayout(gbl_panel_2);

		JLabel lblHtp_1 = new JLabel("HTP");
		GridBagConstraints gbc_lblHtp_1 = new GridBagConstraints();
		gbc_lblHtp_1.insets = new Insets(0, 0, 0, 5);
		gbc_lblHtp_1.anchor = GridBagConstraints.EAST;
		gbc_lblHtp_1.gridx = 0;
		gbc_lblHtp_1.gridy = 0;
		panel_2.add(lblHtp_1, gbc_lblHtp_1);

		htpAcetylTextField = new JTextField();
		GridBagConstraints gbc_textField_2 = new GridBagConstraints();
		gbc_textField_2.insets = new Insets(0, 0, 0, 5);
		gbc_textField_2.fill = GridBagConstraints.HORIZONTAL;
		gbc_textField_2.gridx = 1;
		gbc_textField_2.gridy = 0;
		panel_2.add(htpAcetylTextField, gbc_textField_2);
		htpAcetylTextField.setColumns(10);

		JLabel lblLtp_1 = new JLabel("LTP");
		GridBagConstraints gbc_lblLtp_1 = new GridBagConstraints();
		gbc_lblLtp_1.insets = new Insets(0, 0, 0, 5);
		gbc_lblLtp_1.anchor = GridBagConstraints.EAST;
		gbc_lblLtp_1.gridx = 2;
		gbc_lblLtp_1.gridy = 0;
		panel_2.add(lblLtp_1, gbc_lblLtp_1);

		ltpAcetylTextField = new JTextField();
		GridBagConstraints gbc_textField_3 = new GridBagConstraints();
		gbc_textField_3.insets = new Insets(0, 0, 0, 5);
		gbc_textField_3.fill = GridBagConstraints.HORIZONTAL;
		gbc_textField_3.gridx = 3;
		gbc_textField_3.gridy = 0;
		panel_2.add(ltpAcetylTextField, gbc_textField_3);
		ltpAcetylTextField.setColumns(10);

		acetylCheckBox = new JCheckBox();
		GridBagConstraints gbc_chckbxNewCheckBox = new
GridBagConstraints();
		gbc_chckbxNewCheckBox.gridx = 4;
		gbc_chckbxNewCheckBox.gridy = 0;
		panel_2.add(acetylCheckBox, gbc_chckbxNewCheckBox);

		JPanel panel_3 = new JPanel();
		GridBagConstraints gbc_panel_3 = new GridBagConstraints();

		gbc_panel_3.insets = new Insets(5, 5, 5, 0);

		gbc_panel_3.fill = GridBagConstraints.BOTH;

		gbc_panel_3.weightx = 1.0;

		gbc_panel_3.weighty = 1.0;

		gbc_panel_3.gridx = 0;

		gbc_panel_3.gridy = 2;

		panel.add(panel_3, gbc_panel_3);

		TitledBorder methylationTitle = new TitledBorder("Methylation");

		panel_3.setBorder(methylationTitle);

		GridBagLayout gbl_panel_3 = new GridBagLayout();

		gbl_panel_3.columnWidths = new int[]{0, 0, 0, 0, 0, 0};

		gbl_panel_3.rowHeights = new int[]{0, 0};

		gbl_panel_3.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};

		gbl_panel_3.rowWeights = new double[]{0.0, Double.MIN_VALUE};

		panel_3.setLayout(gbl_panel_3);


		JLabel lblHtp_2 = new JLabel("HTP");

		GridBagConstraints gbc_lblHtp_2 = new GridBagConstraints();

		gbc_lblHtp_2.insets = new Insets(0, 0, 0, 5);

		gbc_lblHtp_2.anchor = GridBagConstraints.EAST;

		gbc_lblHtp_2.gridx = 0;
```

```
gbc_lblHtp_2.gridy = 0;

panel_3.add(lblHtp_2, gbc_lblHtp_2);


htpMethylTextField = new JTextField();

GridBagConstraints gbc_textField_4 = new GridBagConstraints();

gbc_textField_4.insets = new Insets(0, 0, 0, 5);

gbc_textField_4.fill = GridBagConstraints.HORIZONTAL;

gbc_textField_4.gridx = 1;

gbc_textField_4.gridy = 0;

panel_3.add(htpMethylTextField, gbc_textField_4);

htpMethylTextField.setColumns(10);


JLabel lblLtp_2 = new JLabel("LTP");

GridBagConstraints gbc_lblLtp_2 = new GridBagConstraints();

gbc_lblLtp_2.insets = new Insets(0, 0, 0, 5);

gbc_lblLtp_2.anchor = GridBagConstraints.EAST;

gbc_lblLtp_2.gridx = 2;

gbc_lblLtp_2.gridy = 0;

panel_3.add(lblLtp_2, gbc_lblLtp_2);


ltpMethylTextField = new JTextField();

GridBagConstraints gbc_textField_5 = new GridBagConstraints();

gbc_textField_5.insets = new Insets(0, 0, 0, 5);
```

```
gbc_textField_5.fill = GridBagConstraints.HORIZONTAL;
gbc_textField_5.gridx = 3;
gbc_textField_5.gridy = 0;
panel_3.add(ltpMethylTextField, gbc_textField_5);
ltpMethylTextField.setColumns(10);


methylCheckBox = new JCheckBox("");
GridBagConstraints gbc_checkBox_1 = new GridBagConstraints();
gbc_checkBox_1.gridx = 4;
gbc_checkBox_1.gridy = 0;
panel_3.add(methylCheckBox, gbc_checkBox_1);


JPanel panel_4 = new JPanel();
GridBagConstraints gbc_panel_4 = new GridBagConstraints();
gbc_panel_4.insets = new Insets(5, 5, 5, 0);
gbc_panel_4.fill = GridBagConstraints.BOTH;
gbc_panel_4.weightx = 1.0;
gbc_panel_4.weighty = 1.0;
gbc_panel_4.gridx = 0;
gbc_panel_4.gridy = 3;
panel.add(panel_4, gbc_panel_4);
TitledBorder oGalnacTitle = new TitledBorder("O-Galnac");
panel_4.setBorder(oGalnacTitle);
```

```
                GridBagLayout gbl_panel_4 = new GridBagLayout();

                gbl_panel_4.columnWidths = new int[]{0, 0, 0, 0, 0, 0};

                gbl_panel_4.rowHeights = new int[]{0, 0};

                gbl_panel_4.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};

                gbl_panel_4.rowWeights = new double[]{0.0, Double.MIN_VALUE};

                panel_4.setLayout(gbl_panel_4);


                JLabel lblHtp_3 = new JLabel("HTP");

                GridBagConstraints gbc_lblHtp_3 = new GridBagConstraints();

                gbc_lblHtp_3.insets = new Insets(0, 0, 0, 5);

                gbc_lblHtp_3.anchor = GridBagConstraints.EAST;

                gbc_lblHtp_3.gridx = 0;

                gbc_lblHtp_3.gridy = 0;

                panel_4.add(lblHtp_3, gbc_lblHtp_3);


                htpOgalnacTextField = new JTextField();

                GridBagConstraints gbc_textField_6 = new GridBagConstraints();

                gbc_textField_6.insets = new Insets(0, 0, 0, 5);

                gbc_textField_6.fill = GridBagConstraints.HORIZONTAL;

                gbc_textField_6.gridx = 1;

                gbc_textField_6.gridy = 0;

                panel_4.add(htpOgalnacTextField, gbc_textField_6);
```

```
		htpOgalnacTextField.setColumns(10);


		JLabel lblLtp_3 = new JLabel("LTP");

		GridBagConstraints gbc_lblLtp_3 = new GridBagConstraints();

		gbc_lblLtp_3.insets = new Insets(0, 0, 0, 5);

		gbc_lblLtp_3.anchor = GridBagConstraints.EAST;

		gbc_lblLtp_3.gridx = 2;

		gbc_lblLtp_3.gridy = 0;

		panel_4.add(lblLtp_3, gbc_lblLtp_3);


		ltpOgalnacTextField = new JTextField();

		GridBagConstraints gbc_textField_7 = new GridBagConstraints();

		gbc_textField_7.insets = new Insets(0, 0, 0, 5);

		gbc_textField_7.fill = GridBagConstraints.HORIZONTAL;

		gbc_textField_7.gridx = 3;

		gbc_textField_7.gridy = 0;

		panel_4.add(ltpOgalnacTextField, gbc_textField_7);

		ltpOgalnacTextField.setColumns(10);


		ogalnacCheckBox = new JCheckBox();

		GridBagConstraints gbc_chckbxNewCheckBox_1 = new
GridBagConstraints();

		gbc_chckbxNewCheckBox_1.gridx = 4;
```

```
        gbc_chckbxNewCheckBox_1.gridy = 0;
        panel_4.add(ogalnacCheckBox, gbc_chckbxNewCheckBox_1);


        JPanel panel_5 = new JPanel();
        GridBagConstraints gbc_panel_5 = new GridBagConstraints();
        gbc_panel_5.insets = new Insets(5, 5, 5, 0);
        gbc_panel_5.fill = GridBagConstraints.BOTH;
        gbc_panel_5.weightx = 1.0;
        gbc_panel_5.weighty = 1.0;
        gbc_panel_5.gridx = 0;
        gbc_panel_5.gridy = 4;
        panel.add(panel_5, gbc_panel_5);
        TitledBorder oGlcnacTitle = new TitledBorder("O-Glcnac");
        panel_5.setBorder(oGlcnacTitle);
        GridBagLayout gbl_panel_5 = new GridBagLayout();
        gbl_panel_5.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
        gbl_panel_5.rowHeights = new int[]{0, 0};
        gbl_panel_5.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
        gbl_panel_5.rowWeights = new double[]{0.0, Double.MIN_VALUE};
        panel_5.setLayout(gbl_panel_5);


        JLabel lblHtp_4 = new JLabel("HTP");
```

```
GridBagConstraints gbc_lblHtp_4 = new GridBagConstraints();
gbc_lblHtp_4.insets = new Insets(0, 0, 0, 5);
gbc_lblHtp_4.anchor = GridBagConstraints.EAST;
gbc_lblHtp_4.gridx = 0;
gbc_lblHtp_4.gridy = 0;
panel_5.add(lblHtp_4, gbc_lblHtp_4);

htpOglcnacTextField = new JTextField();
GridBagConstraints gbc_textField_8 = new GridBagConstraints();
gbc_textField_8.insets = new Insets(0, 0, 0, 5);
gbc_textField_8.fill = GridBagConstraints.HORIZONTAL;
gbc_textField_8.gridx = 1;
gbc_textField_8.gridy = 0;
panel_5.add(htpOglcnacTextField, gbc_textField_8);
htpOglcnacTextField.setColumns(10);

JLabel lblLtp_4 = new JLabel("LTP");
GridBagConstraints gbc_lblLtp_4 = new GridBagConstraints();
gbc_lblLtp_4.insets = new Insets(0, 0, 0, 5);
gbc_lblLtp_4.anchor = GridBagConstraints.EAST;
gbc_lblLtp_4.gridx = 2;
gbc_lblLtp_4.gridy = 0;
panel_5.add(lblLtp_4, gbc_lblLtp_4);
```

```
ltpOglcnacTextField = new JTextField();

GridBagConstraints gbc_textField_9 = new GridBagConstraints();

gbc_textField_9.insets = new Insets(0, 0, 0, 5);

gbc_textField_9.fill = GridBagConstraints.HORIZONTAL;

gbc_textField_9.gridx = 3;

gbc_textField_9.gridy = 0;

panel_5.add(ltpOglcnacTextField, gbc_textField_9);

ltpOglcnacTextField.setColumns(10);


oglcnacCheckBox = new JCheckBox("");

GridBagConstraints gbc_checkBox_2 = new GridBagConstraints();

gbc_checkBox_2.gridx = 4;

gbc_checkBox_2.gridy = 0;

panel_5.add(oglcnacCheckBox, gbc_checkBox_2);


JPanel panel_6 = new JPanel();

GridBagConstraints gbc_panel_6 = new GridBagConstraints();

gbc_panel_6.insets = new Insets(5, 5, 5, 0);

gbc_panel_6.fill = GridBagConstraints.BOTH;

gbc_panel_6.weightx = 1.0;

gbc_panel_6.weighty = 1.0;

gbc_panel_6.gridx = 0;
```

```
                    gbc_panel_6.gridy = 5;
                    panel.add(panel_6, gbc_panel_6);
                    TitledBorder sumoylationTitle = new TitledBorder("Sumoylation");
                    panel_6.setBorder(sumoylationTitle);
                    GridBagLayout gbl_panel_6 = new GridBagLayout();
                    gbl_panel_6.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
                    gbl_panel_6.rowHeights = new int[]{0, 0};
                    gbl_panel_6.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
                    gbl_panel_6.rowWeights = new double[]{0.0, Double.MIN_VALUE};
                    panel_6.setLayout(gbl_panel_6);

                    JLabel lblHtp_5 = new JLabel("HTP");
                    GridBagConstraints gbc_lblHtp_5 = new GridBagConstraints();
                    gbc_lblHtp_5.insets = new Insets(0, 0, 0, 5);
                    gbc_lblHtp_5.anchor = GridBagConstraints.EAST;
                    gbc_lblHtp_5.gridx = 0;
                    gbc_lblHtp_5.gridy = 0;
                    panel_6.add(lblHtp_5, gbc_lblHtp_5);

                    htpSumoylTextField = new JTextField();
                    GridBagConstraints gbc_textField_10 = new GridBagConstraints();
                    gbc_textField_10.insets = new Insets(0, 0, 0, 5);
```

```
gbc_textField_10.fill = GridBagConstraints.HORIZONTAL;

gbc_textField_10.gridx = 1;

gbc_textField_10.gridy = 0;

panel_6.add(htpSumoylTextField, gbc_textField_10);

htpSumoylTextField.setColumns(10);


JLabel lblLtp_5 = new JLabel("LTP");

GridBagConstraints gbc_lblLtp_5 = new GridBagConstraints();

gbc_lblLtp_5.insets = new Insets(0, 0, 0, 5);

gbc_lblLtp_5.anchor = GridBagConstraints.EAST;

gbc_lblLtp_5.gridx = 2;

gbc_lblLtp_5.gridy = 0;

panel_6.add(lblLtp_5, gbc_lblLtp_5);


ltpSumoylTextField = new JTextField();

GridBagConstraints gbc_textField_11 = new GridBagConstraints();

gbc_textField_11.insets = new Insets(0, 0, 0, 5);

gbc_textField_11.fill = GridBagConstraints.HORIZONTAL;

gbc_textField_11.gridx = 3;

gbc_textField_11.gridy = 0;

panel_6.add(ltpSumoylTextField, gbc_textField_11);

ltpSumoylTextField.setColumns(10);
```

```
		sumoylCheckBox = new JCheckBox("");
		GridBagConstraints gbc_checkBox_3 = new GridBagConstraints();
		gbc_checkBox_3.gridx = 4;
		gbc_checkBox_3.gridy = 0;
		panel_6.add(sumoylCheckBox, gbc_checkBox_3);


		JPanel panel_7 = new JPanel();
		GridBagConstraints gbc_panel_7 = new GridBagConstraints();
		gbc_panel_7.insets = new Insets(5, 5, 5, 0);
		gbc_panel_7.fill = GridBagConstraints.BOTH;
		gbc_panel_7.weightx = 1.0;
		gbc_panel_7.weighty = 1.0;
		gbc_panel_7.gridx = 0;
		gbc_panel_7.gridy = 6;
		panel.add(panel_7, gbc_panel_7);
		TitledBorder ubiquitinationTitle = new TitledBorder("Ubiquitination");
		panel_7.setBorder(ubiquitinationTitle);
		GridBagLayout gbl_panel_7 = new GridBagLayout();
		gbl_panel_7.columnWidths = new int[]{0, 0, 0, 0, 0, 0};
		gbl_panel_7.rowHeights = new int[]{0, 0};
		gbl_panel_7.columnWeights = new double[]{0.0, 1.0, 0.0, 1.0, 0.0,
Double.MIN_VALUE};
		gbl_panel_7.rowWeights = new double[]{0.0, Double.MIN_VALUE};
```

```
panel_7.setLayout(gbl_panel_7);

JLabel lblHtp_6 = new JLabel("HTP");
GridBagConstraints gbc_lblHtp_6 = new GridBagConstraints();
gbc_lblHtp_6.insets = new Insets(0, 0, 0, 5);
gbc_lblHtp_6.anchor = GridBagConstraints.EAST;
gbc_lblHtp_6.gridx = 0;
gbc_lblHtp_6.gridy = 0;
panel_7.add(lblHtp_6, gbc_lblHtp_6);

htpUbiquitinTextField = new JTextField();
GridBagConstraints gbc_textField_12 = new GridBagConstraints();
gbc_textField_12.insets = new Insets(0, 0, 0, 5);
gbc_textField_12.fill = GridBagConstraints.HORIZONTAL;
gbc_textField_12.gridx = 1;
gbc_textField_12.gridy = 0;
panel_7.add(htpUbiquitinTextField, gbc_textField_12);
htpUbiquitinTextField.setColumns(10);

JLabel lblLtp_6 = new JLabel("LTP");
GridBagConstraints gbc_lblLtp_6 = new GridBagConstraints();
gbc_lblLtp_6.insets = new Insets(0, 0, 0, 5);
gbc_lblLtp_6.anchor = GridBagConstraints.EAST;
```

```
gbc_lblLtp_6.gridx = 2;

gbc_lblLtp_6.gridy = 0;

panel_7.add(lblLtp_6, gbc_lblLtp_6);


ltpUbiquitinTextField = new JTextField();

GridBagConstraints gbc_textField_13 = new GridBagConstraints();

gbc_textField_13.insets = new Insets(0, 0, 0, 5);

gbc_textField_13.fill = GridBagConstraints.HORIZONTAL;

gbc_textField_13.gridx = 3;

gbc_textField_13.gridy = 0;

panel_7.add(ltpUbiquitinTextField, gbc_textField_13);

ltpUbiquitinTextField.setColumns(10);


ubiquitinCheckBox = new JCheckBox("");

GridBagConstraints gbc_checkBox_4 = new GridBagConstraints();

gbc_checkBox_4.gridx = 4;

gbc_checkBox_4.gridy = 0;

panel_7.add(ubiquitinCheckBox, gbc_checkBox_4);


JPanel panel_8 = new JPanel();

GridBagConstraints gbc_panel_8 = new GridBagConstraints();

gbc_panel_8.insets = new Insets(0, 0, 5, 0);

gbc_panel_8.fill = GridBagConstraints.BOTH;
```

```
		gbc_panel_8.gridx = 0;

		gbc_panel_8.gridy = 7;

		panel.add(panel_8, gbc_panel_8);


		uniprotCheckBox = new JCheckBox("UniProt PTMs");

		panel_8.add(uniprotCheckBox);


		chckbxSelectAll = new JCheckBox("Select All");

		panel_8.add(chckbxSelectAll);


		btnSubmit = new JButton("Submit");

		GridBagConstraints gbc_btnSubmit = new GridBagConstraints();

		gbc_btnSubmit.insets = new Insets(0, 0, 5, 0);

		gbc_btnSubmit.gridx = 0;

		gbc_btnSubmit.gridy = 8;

		panel.add(btnSubmit, gbc_btnSubmit);

		btnSubmit.setEnabled(false);


		JLabel lblHtpHigh = new JLabel("HTP = High Throughput : LTP = Low
Throughput");

		GridBagConstraints gbc_lblHtpHigh = new GridBagConstraints();

		gbc_lblHtpHigh.gridx = 0;

		gbc_lblHtpHigh.gridy = 9;
		panel.add(lblHtpHigh, gbc_lblHtpHigh);


		htpPhosphoTextField.setText("5");
		ltpPhosphoTextField.setText("1");
		htpAcetylTextField.setText("1");
		ltpAcetylTextField.setText("1");
		htpMethylTextField.setText("1");
		ltpMethylTextField.setText("1");
		htpOgalnacTextField.setText("1");
		ltpOgalnacTextField.setText("1");
		htpOglcnacTextField.setText("1");
		ltpOglcnacTextField.setText("1");
		htpSumoylTextField.setText("1");
		ltpSumoylTextField.setText("1");
		htpUbiquitinTextField.setText("1");
		ltpUbiquitinTextField.setText("1");


		setVisible(true);
	}


	@Override
	public JTextField htpPhosphoTextField() {
		// TODO Auto-generated method stub
		return htpPhosphoTextField;
	}

	@Override
	public JTextField ltpPhosphoTextField() {
		// TODO Auto-generated method stub
		return ltpPhosphoTextField;
	}

	@Override
	public JTextField htpAcetylTextField() {
		// TODO Auto-generated method stub
		return htpAcetylTextField;
	}

	@Override
	public JTextField ltpAcetylTextField() {
		// TODO Auto-generated method stub
		return ltpAcetylTextField;
	}

	@Override
	public JTextField htpMethylTextField() {
		// TODO Auto-generated method stub
		return htpMethylTextField;
	}

	@Override
	public JTextField ltpMethylTextField() {
		// TODO Auto-generated method stub
		return ltpMethylTextField;
	}

	@Override
	public JTextField htpOgalnacTextField() {
		// TODO Auto-generated method stub
		return htpOgalnacTextField;
	}

	@Override
	public JTextField ltpOgalnacTextField() {
		// TODO Auto-generated method stub
		return ltpOgalnacTextField;
	}

	@Override
```

```
public JTextField htpOglcnacTextField() {
	// TODO Auto-generated method stub
	return htpOglcnacTextField;
}

@Override
public JTextField ltpOglcnacTextField() {
	// TODO Auto-generated method stub
	return ltpOglcnacTextField;
}

@Override
public JTextField htpSumoylTextField() {
	// TODO Auto-generated method stub
	return htpSumoylTextField;
}

@Override
public JTextField ltpSumoylTextField() {
	// TODO Auto-generated method stub
	return ltpSumoylTextField;
}
```

```
@Override
public JTextField htpUbiquitinTextField() {
        // TODO Auto-generated method stub
        return htpUbiquitinTextField;
}

@Override
public JTextField ltpUbiquitinTextField() {
        // TODO Auto-generated method stub
        return ltpUbiquitinTextField;
}

@Override
public JCheckBox phosphoCheckBox() {
        // TODO Auto-generated method stub
        return phosphoCheckBox;
}

@Override
public JCheckBox acetylCheckBox() {
        // TODO Auto-generated method stub
        return acetylCheckBox;
}
```

```
@Override
public JCheckBox methylCheckBox() {
        // TODO Auto-generated method stub
        return methylCheckBox;
}

@Override
public JCheckBox ogalnacCheckBox() {
        // TODO Auto-generated method stub
        return ogalnacCheckBox;
}

@Override
public JCheckBox oglcnacCheckBox() {
        // TODO Auto-generated method stub
        return oglcnacCheckBox;
}

@Override
public JCheckBox sumoylCheckBox() {
        // TODO Auto-generated method stub
        return sumoylCheckBox;
```

```
}

@Override
public JCheckBox ubiquitinCheckBox() {
	// TODO Auto-generated method stub
	return ubiquitinCheckBox;
}

@Override
public JCheckBox uniprotCheckBox() {
	// TODO Auto-generated method stub
	return uniprotCheckBox;
}

@Override
public JButton btnSubmit() {
	// TODO Auto-generated method stub
	return btnSubmit;
}

@Override
public JCheckBox chckbxSelectAll() {
	// TODO Auto-generated method stub
```

```
			return chckbxSelectAll;
		}
	}
	package epiprot.services;


	import java.awt.BorderLayout;


	import javax.swing.JFrame;
	import javax.swing.JLabel;
	import javax.swing.JProgressBar;
	import javax.swing.SwingConstants;


	public class ProgressWindow extends JFrame {


		JLabel jLabel = new JLabel();


		public ProgressWindow() {
			// TODO Auto-generated constructor stub
		}


		public void createUI() {
		   //setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		   JProgressBar aJProgressBar = new
```

```
JProgressBar(SwingConstants.HORIZONTAL);
            aJProgressBar.setStringPainted(true);
            aJProgressBar.setIndeterminate(true);

            add(jLabel,BorderLayout.NORTH);
            add(aJProgressBar, BorderLayout.CENTER);
            setSize(300, 80);
            setVisible(true);
        }

        public void setProgressText(String text) {
                jLabel.setText(text);
        }

        public void setTitleName(String serviceName) {
                setTitle(serviceName+ " Progress");
        }

        public static void main (String[]args) {
                ProgressWindow pw = new ProgressWindow();
                pw.setTitleName("Hello");
                pw.setProgressText("Hello");
                pw.createUI();
```

```
	}

}
package epiprot.services.ssp;

import epiprot.AminoAcid;

public class PsiPredAminoAcid extends AminoAcid {

	private int confidenceScore;
	private String prediction;

	public PsiPredAminoAcid() {
		// TODO Auto-generated constructor stub
	}

	public int getConfidenceScore() {
		return confidenceScore;
	}

	public void setConfidenceScore(int confidenceScore) {
		this.confidenceScore = confidenceScore;
	}

	public String getPrediction() {
		return prediction;
	}

	public void setPrediction(String prediction) {
		this.prediction = prediction;
	}

	@Override
	public String toString() {
		return "Conf: " + confidenceScore + " Pred: " + prediction + " AA: " +
super.getResidue();
	}

}
package epiprot.services.ssp;

import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.IOException;

import java.util.ArrayList;

import java.util.Iterator;

import java.util.stream.Stream;


import epiprot.Protein;

import epiprot.services.Service;


public class PsiPredService extends Service {


	private File fastaFile;


	final private String runpsipredplusDir = "/usr/local/psipred/blast+/";


	public PsiPredService(File file) {

		// TODO Auto-generated constructor stub

		this.fastaFile = file;

	}


	@Override

	public void run() {

		// TODO Auto-generated method stub

		try {
```

```
                    String cmd = runpsipredplusDir+"runpsipredplus
"+fastaFile.getAbsolutePath();
    Process p = Runtime.getRuntime().exec(cmd);
    p.waitFor();
        } catch (IOException e) {
    System.out.println("exception happened - here's what I know: ");
    e.printStackTrace();
   } catch (InterruptedException e) {
                    // TODO Auto-generated catch block
                    e.printStackTrace();
   }
    }

    public ArrayList<PsiPredAminoAcid> getAminoAcids() {
        ArrayList<PsiPredAminoAcid> aaList =  new
ArrayList<PsiPredAminoAcid>();
        File file = getPredictionFile();
        System.out.println(file.getAbsolutePath());
        try {
            BufferedReader br = new BufferedReader(new FileReader(file));
            Stream<String> lineStream = br.lines();
            Iterator<String> it = lineStream.iterator();
            StringBuilder confLineSb = new StringBuilder();
        StringBuilder predLineSb = new StringBuilder();
        StringBuilder residueLineSb = new StringBuilder();

while(it.hasNext()) {
        String line = it.next();
    if(line.contains("Conf:")) {
        confLineSb.append(line.substring(6));
    }
    else if(line.contains("Pred:")) {
        predLineSb.append(line.substring(6));
    }
    else if(line.contains("AA:")) {
        residueLineSb.append(line.substring(6));
    }
}
br.close();

String conLine = confLineSb.toString();
String predLine = predLineSb.toString();
String residueLine = residueLineSb.toString();
for(int i = 0; i < conLine.length(); i++) {
        PsiPredAminoAcid aa = new PsiPredAminoAcid();
        aa.setConfidenceScore(Integer.parseInt(conLine.substring(i,
i+1)));
                        aa.setPosition(i+1);
                        aa.setPrediction(predLine.substring(i, i+1));
                        aa.setResidue(residueLine.substring(i, i+1));
                        aaList.add(aa);
                    }

                } catch (FileNotFoundException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                } catch (IOException e) {
                        // TODO Auto-generated catch block
                        e.printStackTrace();
                }
                //delete the files
                deleteFiles();
                return aaList;
        }

        public File getPredictionFile() {
                String fileName = fastaFile.getName().replace(".fasta", ".horiz");
                return new File(SERVICEFILEPATH+fileName);
        }
```

```
	private void deleteFiles() {
		File directory = new File(SERVICEFILEPATH);
		File[] files = directory.listFiles();
		for(File file: files) {
			if(file.getName().contains("psitmp") ||
file.getName().contains(fastaFile.getName().replace(".fasta", ""))) {
				file.delete();
			}
		}
	}

	public static void main(String[]args) {
		Protein protein = new Protein("Q99523");
		System.out.println("test1");
		PsiPredService pps = new PsiPredService(protein.getFastaFile());
		pps.run();
		System.out.println("test2");
		ArrayList<PsiPredAminoAcid> aaList = pps.getAminoAcids();
		System.out.println(aaList.size());
		for (PsiPredAminoAcid aa : aaList) {
			System.out.println(aa.toString());
		}
	}
}
package epiprot.services.views;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;


import javax.swing.JCheckBox;


import epiprot.Presenter;
import epiprot.services.sifts.PdbEntry;


public class SelectPDBPresenter {


	interface View {
		JCheckBox checkBox();
		String getCheckBoxText();
		void createUI();
	}


	View view;
```

```
String proteinAcc;


public SelectPDBPresenter(View view, String proteinAcc) {
	// TODO Auto-generated constructor stub
	this.view = view;
	this.proteinAcc = proteinAcc;
	bindHandler();
}


void bindHandler() {
	view.checkBox().addActionListener(new ActionListener() {
		@Override
		public void actionPerformed(ActionEvent e) {
			// TODO Auto-generated method stub
			String text = view.getCheckBoxText();


			String[] textArray = text.split("\\s+");
			String pdbId = textArray[0];
			String resolution = textArray[2];
			if (view.checkBox().isSelected()) {
				PdbEntry pdbEntry = new PdbEntry();
				pdbEntry.setProteinAcc(proteinAcc);
				pdbEntry.setPdbId(pdbId);
```

```
          pdbEntry.setResolution(Double.parseDouble(resolution));
                                        Presenter.pdbEntryList.add(pdbEntry);
                              }
                              else {
                                        ArrayList<PdbEntry> entryList =
Presenter.pdbEntryList;
                                        for(PdbEntry pdbEntry: entryList) {
                                                  if(pdbEntry.getPdbId().equals(pdbId)) {
                                                            entryList.remove(pdbEntry);
                                                  }
                                        }
                              }
                    }
          });
     }
}
package epiprot.services.views;

import java.awt.Dimension;
import java.awt.FlowLayout;

import javax.swing.JCheckBox;
import javax.swing.JPanel;

public class SelectPDBView extends JPanel implements SelectPDBPresenter.View {

        private String proteinAcc;
        private String pdbId;
        private double resolution;
        private String method;
        private String chain;
        private JCheckBox checkBox;
        private String positions;


        public SelectPDBView(String proteinAcc, String pdbId, String method, double
resolution, String chain, String positions) {
                setSize(new Dimension(500, 30));
                this.proteinAcc = proteinAcc;
                this.pdbId = pdbId;
                this.method = method;
                this.resolution = resolution;
                this.chain = chain;
                this.positions = positions;
                createUI();
```

```
	}

	public SelectPDBView(String proteinAcc, String pdbId, String method, double
resolution, String positions) {
		setSize(new Dimension(500, 30));
		this.proteinAcc = proteinAcc;
		this.pdbId = pdbId;
		this.method = method;
		this.resolution = resolution;
		this.positions = positions;
		createUI();
	}

	@Override
	public void createUI() {
		setPreferredSize(new Dimension(500, 30));
		setMaximumSize(new Dimension(32767, 30));
		setMinimumSize(new Dimension(10, 30));
		FlowLayout flowLayout = new FlowLayout(FlowLayout.LEFT, 5, 5);
		setLayout(flowLayout);

		checkBox = new JCheckBox(pdbId+" "+method+" "+resolution+"
"+positions);
```

```
            checkBox.setPreferredSize(new Dimension(500, 23));
            add(checkBox);

    }

    @Override
    public JCheckBox checkBox() {
            // TODO Auto-generated method stub
            return checkBox;
    }

    @Override
    public String getCheckBoxText() {
            return checkBox.getText();
    }

    public static void main (String[]args) {
            SelectPDBView view = new SelectPDBView("P31749","3OCB","X-
ray",2.70,"A/B","144-480");
            System.out.println(view.getCheckBoxText());
    }

}
```

```
package epiprot.services;

import java.util.TimerTask;

public abstract class Service extends TimerTask {

	// Directory of the project: Users/Patrick/Documents/workspace/epiprot/
	public final static String SERVICEFILEPATH =
System.getProperty("user.dir").toString() + "/";

	public Service () {}
}
package epiprot.services.sifts;

import epiprot.AminoAcid;

public class SiftAminoAcid extends AminoAcid{

	private String pdbResidue;
	private double resolution;
	private String chain;
	private String pdbId;
	private int pdbPosition;
private String uniprotId;

private String secondaryStructure;


public SiftAminoAcid() {

	// TODO Auto-generated constructor stub

}


public String getPdbResidue() {

	return pdbResidue;

}


public void setPdbResidue(String pdbResidue) {

	this.pdbResidue = pdbResidue;

}


public double getResolution() {

	return resolution;

}


public void setResolution(double resolution) {

	this.resolution = resolution;

}
```

```
public String getChain() {

        return chain;

}


public void setChain(String chain) {

        this.chain = chain;

}


public String getPdbId() {

        return pdbId;

}


public void setPdbId(String pdbId) {

        this.pdbId = pdbId;

}


public int getPdbPosition() {

        return pdbPosition;

}


public void setPdbPosition(int pdbPosition) {

        this.pdbPosition = pdbPosition;

}
```

```
public String getUniprotId() {
        return uniprotId;
}

public void setUniprotId(String uniprotId) {
        this.uniprotId = uniprotId;
}

public String getSecondaryStructure() {
        return secondaryStructure;
}

public void setSecondaryStructure(String secondaryStructure) {
        this.secondaryStructure = secondaryStructure;
}

@Override
public String toString() {
        return getResidue()+"|"+getPosition()+"|"+secondaryStructure;
}
```

```
}

package epiprot.services.sifts;


import java.io.IOException;

import java.io.InputStream;

import java.net.MalformedURLException;

import java.net.URL;

import java.net.URLConnection;

import java.util.ArrayList;

import javax.xml.parsers.DocumentBuilder;

import javax.xml.parsers.DocumentBuilderFactory;

import javax.xml.parsers.ParserConfigurationException;


import org.w3c.dom.Document;

import org.w3c.dom.Element;

import org.w3c.dom.Node;

import org.w3c.dom.NodeList;

import org.xml.sax.SAXException;


import epiprot.services.Service;


public class SiftService extends Service {
```

```
private String pdbId;

private String acc;

private double structureResolution;

private Document doc = null;


public SiftService (String pdbId, String acc, double structureResolution) {

        this.acc = acc;

        this.pdbId = pdbId;

        this.structureResolution = structureResolution;

}


@Override

public void run() {

        // TODO Auto-generated method stub

        try {

    URL url = new URL("http://www.rcsb.org/pdb/files/"+pdbId+".sifts.xml");

    URLConnection connection = url.openConnection();

    connection.setConnectTimeout(10000);

    DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();

    dbf.setNamespaceAware(false);

    dbf.setValidating(false);

    dbf.setFeature("http://xml.org/sax/features/namespaces", false);

    dbf.setFeature("http://xml.org/sax/features/validation", false);
```

```
                dbf.setFeature("http://apache.org/xml/features/nonvalidating/load-dtd-
grammar", false);
                dbf.setFeature("http://apache.org/xml/features/nonvalidating/load-external-
dtd", false);
                DocumentBuilder db = dbf.newDocumentBuilder();
                InputStream inputStream = connection.getInputStream();
                   doc = db.parse(inputStream);
        } catch (MalformedURLException e) {
                   e.printStackTrace();
        } catch (ParserConfigurationException e) {
                   e.printStackTrace();
        } catch (IOException e) {
                   e.printStackTrace();
        } catch (SAXException e) {
                   e.printStackTrace();
        }
        }

        public ArrayList<SiftAminoAcid> getAminoAcids() {
                   //this.run();
                   ArrayList<SiftAminoAcid> aminoAcidList = new
ArrayList<SiftAminoAcid>();
                   NodeList list = doc.getElementsByTagName("residue");
                for (int i = 0; i < list.getLength(); i++) {
                        Element resElement = (Element) list.item(i);
                        if (resElement.getAttribute("dbSource").equals("PDBe")) {
                                SiftAminoAcid aminoAcid = new SiftAminoAcid();
                                String pdbResidue =
resElement.getAttribute("dbResName");
                                NodeList childNodes1 = resElement.getChildNodes();
                                aminoAcid.setPdbResidue(pdbResidue);
                                aminoAcid.setResolution(structureResolution);
                                for (int j = 0; j < childNodes1.getLength(); j++) {
                                        Node childNode = childNodes1.item(j);
                                        if
(childNode.getNodeType()==Node.ELEMENT_NODE) {
                                                Element childElement = (Element)
childNode;
                                                if
(childElement.getAttribute("dbSource").equals("PDB")) {
                                                        String pdbPosString =
getNumber(childElement.getAttribute("dbResNum"));
                                                        int pdbPosition =
Integer.parseInt(pdbPosString);
                                                        String chain =
childElement.getAttribute("dbChainId");
```

```
                                                String pdbId =
childElement.getAttribute("dbAccessionId");


        aminoAcid.setPdbPosition(pdbPosition);
                                                aminoAcid.setChain(chain);
                                                aminoAcid.setPdbId(pdbId);
                                        }
                                        else if
(childElement.getAttribute("dbSource").equals("UniProt")) {
                                                int uniprotPosition =
Integer.parseInt(childElement.getAttribute("dbResNum"));
                                                String uniprotId =
childElement.getAttribute("dbAccessionId");
                                                String uniprotResidue =
childElement.getAttribute("dbResName");


        aminoAcid.setPosition(uniprotPosition);
                                                aminoAcid.setUniprotId(uniprotId);


        aminoAcid.setResidue(uniprotResidue);
                                        }
                                        else if
(childElement.getAttribute("dbSource").equals("PDBe") &&
childElement.getAttribute("property").equals("codeSecondaryStructure")) {
                                                            String secondaryStructure =
childElement.getTextContent();

        aminoAcid.setSecondaryStructure(secondaryStructure);
                                                }
                                        }
                                }
                                if (aminoAcid.getUniprotId() != null &&
aminoAcid.getUniprotId().equals(acc)) {
                                        aminoAcidList.add(aminoAcid);
                                }
                        }
                }
                return aminoAcidList;
        }

        private String getNumber(String s) {
                String number = "";
                for (int i = 0; i < s.length(); i++) {
                        if (Character.isDigit(s.charAt(i))) {
                                number += s.charAt(i);
                        }
```

```
				}
				return number;
		}


		public static void main (String [] args) {
				SiftService siftService = new SiftService("3G2U","Q99523",2.70);
				siftService.run();
				ArrayList<SiftAminoAcid> aminoAcidList =
siftService.getAminoAcids();
				for (int i = 0; i < aminoAcidList.size(); i++) {
						SiftAminoAcid aminoAcid = aminoAcidList.get(i);
						System.out.println(aminoAcid.toString());
				}
		}


}
package epiprot.services.views;


import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.util.ArrayList;


import javax.swing.JButton;
import javax.swing.JPanel;


import epiprot.Presenter;
import epiprot.services.sifts.PdbEntry;
import epiprot.services.sifts.SiftAminoAcid;
import epiprot.services.sifts.SiftService;


public class SiftsPresenter {


	interface View {
		JPanel panel();
		JButton btnSubmit();


		void insertEntry(SelectPDBView spv);
	}


	View view;
	Presenter presenter;


	public SiftsPresenter(Presenter presenter, SiftsView siftsView) {
		// TODO Auto-generated constructor stub
		this.presenter = presenter;
		this.view = siftsView;
		bindHandlers();
	}


	public void bindHandlers() {
		view.btnSubmit().addActionListener(new ActionListener() {
			@Override
			public void actionPerformed(ActionEvent e) {
				// TODO Auto-generated method stub
				System.out.println("test submit siftspresenter");
				ArrayList<PdbEntry> pdbEntries = Presenter.pdbEntryList;
				for(PdbEntry pdbEntry: pdbEntries) {
					SiftService siftService = new
SiftService(pdbEntry.getPdbId(),pdbEntry.getProteinAcc(),pdbEntry.getResolution());
					siftService.run();
					ArrayList<SiftAminoAcid> aaList =
siftService.getAminoAcids();
					for (int i = 0; i < aaList.size(); i++) {
						SiftAminoAcid aminoAcid = aaList.get(i);
						System.out.println(aminoAcid.toString());
					}
					String line = "";
					for(int i = 0; i <
presenter.protein.getSequence().length(); i++) {
```

```
                                        line = line + "-";
                                }

                                for(SiftAminoAcid aa: aaList) {
                                        if(aa.getSecondaryStructure() != null) {
                                                line =
line.substring(0,aa.getPosition()-
1)+aa.getSecondaryStructure()+line.substring(aa.getPosition());
                                        }
                                }

                                String mainLine = presenter.getMainLine();
                                for(int i = 0; i < mainLine.length(); i++) {
                                        if(mainLine.charAt(i) == '-') {
                                                line = new
StringBuilder(line).insert(i, ' ').toString();
                                        }
                                }

                                System.out.println("PDB: "+pdbEntry.getPdbId()
+"|"+ line);
                                presenter.insertLineAboveTarget("PDB:
"+pdbEntry.getPdbId(), line);
```

```
                                }
                        }
                });
        }
}
package epiprot.services.views;


import javax.swing.JFrame;
import java.awt.Dimension;
import javax.swing.JPanel;


import java.awt.BorderLayout;
import javax.swing.BoxLayout;
import javax.swing.JButton;


public class SiftsView extends JFrame implements SiftsPresenter.View {


        JPanel panel = new JPanel();
        JButton btnSubmit = new JButton("Submit");


        public SiftsView() {
                // TODO Auto-generated constructor stub
                setTitle("SIFTS Secondary Structure");
                setSize(new Dimension(500, 700));

                getContentPane().add(panel, BorderLayout.CENTER);
                panel.setLayout(new BoxLayout(panel, BoxLayout.Y_AXIS));

                JPanel submitPanel = new JPanel();
                submitPanel.setPreferredSize(new Dimension(10, 50));
                submitPanel.setMinimumSize(new Dimension(30, 30));
                submitPanel.setSize(new Dimension(0, 30));
                getContentPane().add(submitPanel, BorderLayout.SOUTH);

                submitPanel.add(btnSubmit);
                setVisible(true);
        }

        @Override
        public JPanel panel() {
                // TODO Auto-generated method stub
                return panel;
        }

        @Override
        public JButton btnSubmit() {
```

```
		// TODO Auto-generated method stub
		return btnSubmit;
	}

	@Override
	public void insertEntry(SelectPDBView spv) {
		// TODO Auto-generated method stub
		panel.add(spv);
	}

	public static void main(String[]args) {
		SiftsView siftsView = new SiftsView();
		SelectPDBView spv = new SelectPDBView("P31749","3OCB","X-
ray",2.70,"A/B","144-480");

	}

}
package epiprot.services.uniprot;

import java.util.ArrayList;

import epiprot.AminoAcid;
```

```
public class UniProtAminoAcid extends AminoAcid{

	private boolean isPhosphorylated;
	private boolean isAcetylated;
	private boolean isMethylated;
	private boolean isGlycosylated;
	private boolean isOGalNAc;
	private boolean isOGlcNAc;
	private boolean isSumoylated;
	private boolean isUbiquitinated;
	private boolean isPTM;

	private String peptide;

	private ArrayList<UniProtPTM> ptmList = new ArrayList<UniProtPTM>();

	public UniProtAminoAcid() {
		// TODO Auto-generated constructor stub
	}

	public boolean isAcetylated() {
		return isAcetylated;
```

```
}

public void setAcetylated(boolean isAcetylated) {
        this.isAcetylated = isAcetylated;
}

public boolean isPhosphorylated() {
        return isPhosphorylated;
}

public void setPhosphorylated(boolean isPhosphorylated) {
        this.isPhosphorylated = isPhosphorylated;
}

public boolean isMethylated() {
        return isMethylated;
}

public void setMethylated(boolean isMethylated) {
        this.isMethylated = isMethylated;
}

public boolean isGlycosylated() {
        return isGlycosylated;
}

public void isGlycosylated(boolean isGlycosylated) {
        this.isGlycosylated = isGlycosylated;
}

public boolean isOGalNAc() {
        return isOGalNAc;
}

public void setOGalNAc(boolean isOGalNAc) {
        this.isOGalNAc = isOGalNAc;
}

public boolean isOGlcNAc() {
        return isOGlcNAc;
}

public void setOGlcNAc(boolean isOGlcNAc) {
        this.isOGlcNAc = isOGlcNAc;
}
```

```
public boolean isSumoylated() {
        return isSumoylated;
}

public void setSumoylated(boolean isSumoylated) {
        this.isSumoylated = isSumoylated;
}

public boolean isUbiquitinated() {
        return isUbiquitinated;
}

public void setUbiquitinated(boolean isUbiquitinated) {
        this.isUbiquitinated = isUbiquitinated;
}

public String getPeptide() {
        return peptide;
}

public void setPeptide(String peptide) {
        this.peptide = peptide;
}
```

```
public UniProtAminoAcid(UniProtAminoAcid aminoAcidToCopy) {
        this.isAcetylated = aminoAcidToCopy.isAcetylated;
        this.isMethylated = aminoAcidToCopy.isMethylated;
        this.isOGalNAc = aminoAcidToCopy.isOGalNAc;
        this.isOGlcNAc = aminoAcidToCopy.isOGlcNAc;
        this.isGlycosylated = aminoAcidToCopy.isGlycosylated;
        this.isPhosphorylated = aminoAcidToCopy.isPhosphorylated;
        this.isSumoylated = aminoAcidToCopy.isSumoylated;
        this.isUbiquitinated = aminoAcidToCopy.isUbiquitinated;
        this.isPTM = aminoAcidToCopy.isPTM;
        this.peptide = aminoAcidToCopy.peptide;
        aminoAcidToCopy.setPosition(super.getPosition());
        aminoAcidToCopy.setResidue(super.getResidue());
}

public boolean isPTM() {
        return isPTM;
}

public void setPTM(boolean isPTM) {
        this.isPTM = isPTM;
}
```

```
	public void addPTM(UniProtPTM phosphoSitePLM) {
		ptmList.add(phosphoSitePLM);
	}

	public ArrayList<UniProtPTM> getUniProtPTMs() {
		return ptmList;
	}

	@Override
	public String toString() {
		String aa = this.getResidue()+"|"+this.getPosition();
		for(UniProtPTM ptm : this.ptmList) {
			aa = aa + "|" + ptm.toString();
		}
		return aa;
	}
}
package epiprot.services.blast;

import java.util.ArrayList;
import java.util.Iterator;
import java.util.concurrent.CompletableFuture;
import java.util.concurrent.ExecutionException;


import epiprot.Protein;

import epiprot.services.ProgressWindow;

import epiprot.services.Service;

import uk.ac.ebi.uniprot.dataservice.client.Client;

import uk.ac.ebi.uniprot.dataservice.client.ServiceFactory;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.BlastInput;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.BlastResult;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.UniProtBlastService;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.UniProtHit;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.AlignmentCutoffOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.DatabaseOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.DropOffOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.ExpectationOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.FilterOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.MatrixOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.ScoreCutoffOption;

import uk.ac.ebi.uniprot.dataservice.client.alignment.blast.input.SequenceTypeOption;


public class UniprotBlastService extends Service {


        private String sequence;
```

```
	private String species;
	private String database;
	private DatabaseOption databaseOption;
	private ExpectationOption expectationOption;
	private FilterOption filterOption;
	private DropOffOption dropOffOption;
private boolean isGapAlign;
private int gapExt;
private int gapOpen;
private MatrixOption matrixOption;
private AlignmentCutoffOption alignmentCutoffOption;
private ScoreCutoffOption scoreCutoffOption;
	private boolean limitToTargetSpecies;
	private boolean limitToSwissProtDB;

	ArrayList<UniProtHit> uniProtHitList = new ArrayList<UniProtHit>();

	public UniprotBlastService(String sequence, DatabaseOption databaseOption) {
		// TODO Auto-generated constructor stub
		this.sequence = sequence;
		this.databaseOption = databaseOption;
	}
	public UniprotBlastService(Protein protein, DatabaseOption databaseOption) {
		// TODO Auto-generated constructor stub
		this.sequence = protein.getSequence();
		this.species = protein.getOrganism();
		this.database = protein.getDatabase();
		this.databaseOption = databaseOption;
	}

	public UniprotBlastService(Protein protein, DatabaseOption databaseOption,
ExpectationOption expectationOption, FilterOption filterOption, DropOffOption
dropOffOption, MatrixOption matrixOption, ScoreCutoffOption scoreCutoffOption,
boolean isGapAlign, int gapExt, int gapOpen, boolean limitToTargetSpecies, boolean
limitToSwissProtDB) {
		// TODO Auto-generated constructor stub
		this.sequence = protein.getSequence();
		this.species = protein.getOrganism();
		this.database = protein.getDatabase();
		this.databaseOption = databaseOption;
		this.expectationOption = expectationOption;
		this.filterOption = filterOption;
		this.dropOffOption = dropOffOption;
	    this.isGapAlign = isGapAlign;
	    this.gapExt = gapExt;
```

```
        this.gapOpen = gapOpen;
        this.matrixOption = matrixOption;
        this.scoreCutoffOption = scoreCutoffOption;
            this.limitToTargetSpecies = limitToTargetSpecies;
            this.limitToSwissProtDB = limitToSwissProtDB;
    }

    @Override
    public void run() {
            ProgressWindow progressWindow = new ProgressWindow();
            progressWindow.setTitleName("UniProt BLAST");
            //Get the UniProt Service. This is how to access the blast service
            // Create UniProt blast service
            ServiceFactory serviceFactoryInstance =
Client.getServiceFactoryInstance();
            UniProtBlastService uniProtBlastService =
serviceFactoryInstance.getUniProtBlastService();
            uniProtBlastService.start();
        //Create a blast input with a Database and sequence
        BlastInput.Builder builder = new BlastInput.Builder(databaseOption,sequence);
        if(expectationOption!=null) {builder.withExpectation(expectationOption);}
        if(filterOption!=null) {builder.withFilter(filterOption);}
        if(dropOffOption!=null) {builder.withDropOff(dropOffOption);}
```

```
            builder.withGapAlign(isGapAlign);
            if(gapExt!=0) {builder.withGapExt(gapExt);}
            if(gapOpen!=0) {builder.withGapOpen(gapOpen);}
            if(matrixOption!=null) {builder.withMatrix(matrixOption);}
            if(alignmentCutoffOption!=null)
{builder.withMaximumNumberOfAlignments(alignmentCutoffOption);}
            if(scoreCutoffOption!=null)
{builder.withMaximumNumberOfScores(scoreCutoffOption);}
            builder.withSequenceType(SequenceTypeOption.PROTEIN);

            BlastInput input = builder.build();

            CompletableFuture <BlastResult<UniProtHit>> resultFuture =
uniProtBlastService.runBlast(input);

            try {
                  BlastResult blastResult = resultFuture.get();

                  Iterator it = blastResult.hits().iterator();

                  while(it.hasNext()) {
                        UniProtHit hit = (UniProtHit) it.next();
```

```
        //System.out.println(hit.getEntry().getPrimaryUniProtAccession().getValue());
                    uniProtHitList.add(hit);
                }

        } catch (ExecutionException e) {
        } catch (InterruptedException e) {
        } finally {
            uniProtBlastService.stop();
        }
            }

            public ArrayList<UniProtHit> getBlastResults() {
                ArrayList<UniProtHit> returnedBlastResult = new
ArrayList<UniProtHit>();

                for(UniProtHit uniProtHit: uniProtHitList) {
                    Protein hitProtein = new
Protein(uniProtHit.getSummary().getEntryAc(),true);
                    System.out.println(hitProtein.getOrganism()+" "+species);
                    System.out.println(hitProtein.getDatabase()+" "+"Swiss-Prot");
                    if(limitToTargetSpecies && limitToSwissProtDB) {
                        if(hitProtein.getOrganism().equals(species) &&
hitProtein.getDatabase().equals("Swiss-Prot")) {
```

```
                        returnedBlastResult.add(uniProtHit);
                    }
                }
                else if(limitToTargetSpecies) {
                    if(hitProtein.getOrganism().equals(species)) {
                        returnedBlastResult.add(uniProtHit);
                    }
                }
                else if(limitToSwissProtDB) {
                    if(hitProtein.getDatabase().equals("Swiss-Prot")) {
                        returnedBlastResult.add(uniProtHit);
                    }
                }
                else {
                    returnedBlastResult.add(uniProtHit);
                }
            }
            return returnedBlastResult;
        }

        public void setMatrixOption(MatrixOption matrixOption) {
            this.matrixOption = matrixOption;
        }
```

```
	/*
	public BlastData<UniProtEntry> getBlastData() {
		return blastResult;
	}
	*/
	public static void main (String[]args) {
		UniprotBlastService ubs = new UniprotBlastService(new
Protein("Q99523",true),DatabaseOption.UNIPROT_HUMAN);
		//UniprotBlastService ubs = new
UniprotBlastService("TSDDRGIVYSKSLD",DatabaseOption.UNIPROT_HUMAN);
		ubs.setMatrixOption(MatrixOption.BLOSUM_62);
		ubs.run();
		ArrayList<UniProtHit> returnedBlastResult = ubs.getBlastResults();
		for(UniProtHit uniProtHit: returnedBlastResult) {

	System.out.println(uniProtHit.getSummary().getEntryId()+"|"+uniProtHit.getSum
mary().getDatabase()+"|"+uniProtHit.getSummary().getHitNumber());
		}
	}

}
package epiprot.services.uniprot;
```

```
public class UniProtFeature {

        private String uniprotType;
        private String description;
        private int []evidence;
        private int position;
        private int beginPosition;
        private int endPosition;

        //subcellular location and protein processing
        //unprot types
        public static final String TOPOLOGICAL_DOMAIN = "topological domain";
        public static final String TRANSMEMBRANE_REGION = "transmembrane
region";
        public static final String INTRAMEMBRANE_REGION = "intramembrane
region";
        public static final String SIGNAL_PEPTIDE = "signal peptide";
        public static final String PROPEPTIDE = "propeptide";
        public static final String CHAIN = "chain";
        //types of topological domains, in description
        public static final String EXTRACELLULAR = "extracellular";
        public static final String CYTOPLASMIC = "cytoplasmic";
```

```
	public UniProtFeature() {
		// TODO Auto-generated constructor stub
	}

	public String getUniprotType() {
		return uniprotType;
	}

	public void setUniprotType(String uniprotType) {
		this.uniprotType = uniprotType;
	}



	@Override
	public String toString() {
		return
this.uniprotType+":position:"+this.position+":beginPosition:"+this.beginPosition+":endP
osition:"+this.endPosition;
	}

	public String getDescription() {
		return description;
```

```
}

public void setDescription(String description) {
	this.description = description;
}

public int []getEvidence() {
	return evidence;
}

public void setEvidence(int []evidence) {
	this.evidence = evidence;
}

public int getPosition() {
	return position;
}

public void setPosition(int position) {
	this.position = position;
}

public int getBeginPosition() {
		return beginPosition;
	}

	public void setBeginPosition(int beingPosition) {
		this.beginPosition = beingPosition;
	}

	public int getEndPosition() {
		return endPosition;
	}

	public void setEndPosition(int endPosition) {
		this.endPosition = endPosition;
	}

	public boolean hasMultiplePositions() {
		return position == 0;
	}

}
package epiprot.services.uniprot;

public class UniProtPTM extends UniProtFeature{
	private String ptmType;
	private String linkage;
	private int disulfideNumber;

	//PTM constants
	public static final String PHOSPHOYLATED = "phosphoylated";  //p
	public static final String ACETYLATED = "acetylated";  //a
	public static final String AMIDATION = "amidation";  //d
	public static final String METHYLATED = "methylated";  //m
	public static final String SUMOYLATED = "sumoylated"; //s
	public static final String UBIQUITINATED = "ubiquitinated"; //u
	public static final String PYRROLIDONE = "pyrrolidone"; //z
	public static final String ISOMERIZATION = "isomerization"; //i
	public static final String HYDROXYLATION = "hydroxylation"; //h
	public static final String SULFATION = "sulfation"; //$
	public static final String FLAVIN = "flavin"; //f

	//glycosylation uniprot
	public static final String GLYCOSYLATED = "glycosylated"; //g when not 'o' or
'n' linked
	public static final String NLINKED = "n-linked";
	public static final String OLINKED = "o-linked";
	public static final String OTHERLINKED = "other linked";
	public static final String GALNAC = "galnac";  //o or n
	public static final String GLCNAC = "glcnac"; //\u00F6 or \u00F1

	//disulfide
	public static final String DISULFIDEBOND = "disulfide"; //number

	public UniProtPTM(){
		 super();
	}

	public String getPtmType() {
		return ptmType;
	}

	public void setPtmType(String ptmType) {
		this.ptmType = ptmType;
	}

	public void setType() {

	//System.out.println("++++"+this.getUniprotType().toLowerCase().equals("modif
ied residue"));
```

```
                if(this.getUniprotType().toLowerCase().equals("modified residue")) {

                        //p

                        if(this.getDescription().toLowerCase().contains("phospho")) {

                                this.setPtmType(UniProtPTM.PHOSPHOYLATED);

                        }

                        //m

                        else if(this.getDescription().toLowerCase().contains("methyl")) {

                                this.setPtmType(UniProtPTM.METHYLATED);

                        }

                        //a

                        else if(this.getDescription().toLowerCase().contains("acetyl")) {

                                this.setPtmType(UniProtPTM.ACETYLATED);

                        }

                        //d

                        else if(this.getDescription().toLowerCase().contains("amide")) {

                                this.setPtmType(UniProtPTM.AMIDATION);

                        }

                        //z

                        else

        if(this.getDescription().toLowerCase().contains("pyrrolidone")) {

                                this.setPtmType(UniProtPTM.PYRROLIDONE);

                        }

                        //i
                        else
if(this.getDescription().toLowerCase().contains("isomerization")) {
                                this.setPtmType(UniProtPTM.ISOMERIZATION);
                        }
                        //h
                        else if(this.getDescription().toLowerCase().contains("hydroxy")) {
                                this.setPtmType(UniProtPTM.HYDROXYLATION);
                        }
                        //$
                        else if(this.getDescription().toLowerCase().contains("sulfo")) {
                                this.setPtmType(UniProtPTM.SULFATION);
                        }
                        //f
                        else if(this.getDescription().toLowerCase().contains("-
fad")||super.getDescription().toLowerCase().contains("fmn")) {
                                this.setPtmType(UniProtPTM.FLAVIN);
                        }
                }
                else if(this.getUniprotType().toLowerCase().equals("glycosylation site")){
                        this.setPtmType(UniProtPTM.GLYCOSYLATED);
                        if(this.getDescription().toLowerCase().contains("n-linked")) {
                                this.setLinkage(UniProtPTM.NLINKED);
                        }
```

```
				else if (this.getDescription().toLowerCase().contains("o-linked")) {
					this.setLinkage(UniProtPTM.OLINKED);
				}
				else {
					this.setLinkage(UniProtPTM.OTHERLINKED);
				}
				if (this.getDescription().toLowerCase().contains("glcnac")) {
					this.setPtmType(UniProtPTM.GLCNAC);
				}
				else if (this.getDescription().toLowerCase().contains("galnac")) {
					this.setPtmType(UniProtPTM.GALNAC);
				}
			}
			else if(this.getUniprotType().toLowerCase().equals("disulfide bond")){
				this.setPtmType(UniProtPTM.DISULFIDEBOND);
			}
			//System.out.println(this.toString());
		}

		public String getLinkage() {
			return linkage;
		}
```

```
	public void setLinkage(String linkage) {
		this.linkage = linkage;
	}

	@Override
	public String toString() {
		return
this.getPosition()+"|"+this.getBeginPosition()+"|"+this.getEndPosition()+"|"+this.getPtm
Type()+"|"+this.getLinkage();
	}

	public int getDisulfideNumber() {
		return disulfideNumber;
	}

	public void setDisulfideNumber(int disulfideNumber) {
		this.disulfideNumber = disulfideNumber;
	}

}
package epiprot.services.uniprot;

import java.io.File;
import java.io.IOException;
import java.io.PrintWriter;
import java.net.MalformedURLException;
import java.net.URL;

import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;

import org.apache.commons.io.FileUtils;
import org.w3c.dom.Document;
import org.xml.sax.SAXException;

import epiprot.services.Service;

public class UniprotService extends Service {

	public String id;
	public Document doc;

	public UniprotService (String id) {
		this.id = id;
	}
	public File fetchFastaFile() {
		File file = null;
	try {
			URL url = new
URL("http://www.uniprot.org/uniprot/"+id+".fasta");
			file = new File(SERVICEFILEPATH + id + ".fasta");
			PrintWriter writer = new PrintWriter(file);
			writer.close();
			FileUtils.copyURLToFile(url, file);
		} catch (MalformedURLException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		} catch (IOException e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	return file;
	}

	@Override
	public void run() {
		// TODO Auto-generated method stub
```

```
                try {
            URL url = new URL("http://www.uniprot.org/uniprot/"+id+".xml");
            DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
            DocumentBuilder db = dbf.newDocumentBuilder();
            doc = db.parse(url.openStream());
        } catch (MalformedURLException e) {
                e.printStackTrace();
        } catch (ParserConfigurationException e) {
                e.printStackTrace();
        } catch (IOException e) {
                e.printStackTrace();
        } catch (SAXException e) {
                e.printStackTrace();
        }
        }

        public Document getDocument() {
                return doc;
        }

        public static void main (String[]args) {
                UniprotService uniprot = new UniprotService("Q99523");
                System.out.println(Service.SERVICEFILEPATH);
```

Bibliography

AACon Web Service. (2016.). Retrieved June 15, 2016, from http://www.jalview.org/help/html/webServices/AACon.html

Ali, S. A., Hassan, M. I., Islam, A., & Ahmad, F. (2014). A review of methods available to estimate solvent-accessible surface areas of soluble proteins in the folded and unfolded states. *Current Protein & Peptide Science*, *15*(5), 456–476.

Altschul, S. F., Gish, W., Miller, W., Myers, E. W., & Lipman, D. J. (1990). Basic local alignment search tool. *Journal of Molecular Biology*, *215*(3), 403–410.

Angeletti, R. H. (1999). Design of useful peptide antigens. *Journal of Biomolecular Techniques: JBT*, *10*(1), 2–10.

Antibody Basics | Novus Biologicals. (2016.). Retrieved June 15, 2016, from http://www.novusbio.com/support/general-support/antibody-basics.html

ArrayList (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/java/util/ArrayList.html

Beck, K., Beedle, M., Van Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., ... & Kern, J. (2001). Manifesto for agile software development.

Berman, H. M., Westbrook, J., Feng, Z., Gilliland, G., Bhat, T. N., Weissig, H., … Bourne, P. E. (2000). The Protein Data Bank. *Nucleic Acids Research*, *28*(1), 235–242.

Biocompare. (2015). Retrieved June 14, 2016, from http://www.biocompare.com/Editorial-Articles/177815-2015-Antibody-Market-Report.

Boeckmann, B., Bairoch, A., Apweiler, R., Blatter, M.-C., Estreicher, A., Gasteiger, E., … Schneider, M. (2003). The SWISS-PROT protein knowledgebase and its supplement TrEMBL in 2003. *Nucleic Acids Research*, *31*(1), 365–370.

Bordeaux, J., Jennifer, B., Allison, W., Seema, A., Elizabeth, K., Maria, B., … David, R. (2010). Antibody validation. *BioTechniques*, *48*(3), 197–209.

Bradbury, A., & Plückthun, A. (2015). Reproducibility: Standardize antibodies used in research. *Nature*, *518*(7537), 27–29.

Brekke, O. H., & Inger, S. (2003). Therapeutic antibodies for human diseases at the dawn of the twenty-first century. *Nature Reviews. Drug Discovery*, *2*(1), 52–62.

Burns, R., & Robert, B. (2005). Immunization Strategies for Antibody Production. In *Immunochemical Protocols* (pp. 1–11).

Buus, S., Rockberg, J., Forsström, B., Nilsson, P., Uhlen, M., & Schafer-Nielsen, C. (2012). High-resolution mapping of linear antibody epitopes using ultrahigh-density peptide microarrays. *Molecular & Cellular Proteomics: MCP*, *11*(12), 1790–1800.

Chames, P., Patrick, C., Van Regenmortel, M., Etienne, W., & Daniel, B. (2009). Therapeutic antibodies: successes, limitations and hopes for the future. *British Journal of Pharmacology*, *157*(2), 220–233.

Chaplin, D. D. (2016.). Overview of the immune response. - PubMed - NCBI. Retrieved June 15, 2016, from http://www.ncbi.nlm.nih.gov/pubmed/20176265

Cheung, W. C., Beausoleil, S. A., Zhang, X., Sato, S., Schieferl, S. M., Wieler, J. S., … Polakiewicz, R. D. (2012). A proteomics approach for the identification and cloning of monoclonal antibodies from serum. *Nature Biotechnology*, *30*(5), 447–452.

Chou, P. Y. (1978). *Prediction of the Secondary Structure of Proteins from Their Amino Acid Sequence*.

Core J2EE Patterns - Data Access Object. (2016.). Retrieved June 15, 2016, from http://www.oracle.com/technetwork/java/dataaccessobject-138824.html

Cuff, J. A., Clamp, M. E., Siddiqui, A. S., Finlay, M., & Barton, G. J. (1998). JPred: a consensus secondary structure prediction server. *Bioinformatics* , *14*(10), 892–893.

Divan, A., & Royds, J. (2013). *Tools and Techniques in Biomolecular Science*. Oxford University Press.

Document (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/org/w3c/dom/Document.html

Dosztányi, Z., Csizmok, V., Tompa, P., & Simon, I. (2005). IUPred: web server for the prediction of intrinsically unstructured regions of proteins based on estimated energy content. *Bioinformatics* , *21*(16), 3433–3434.

Ebersbach, H., & Geisse, S. (2012). Antigen generation and display in therapeutic antibody drug discovery -- a neglected but critical player. *Biotechnology Journal*, *7*(12), 1433–1443.

Edgar, R. C. (2004). MUSCLE: multiple sequence alignment with high accuracy and high throughput. *Nucleic Acids Research*, *32*(5), 1792–1797.

Edgar, R. C., & Serafim, B. (2006). Multiple sequence alignment. *Current Opinion in Structural Biology*, *16*(3), 368–373.

Emini, E. A. et al. (1985). Induction of hepatitis A virus-neutralizing antibody by a virus-specific synthetic peptide. *Journal of Virology*, *55*(3), 836–839.

Hancock, D. C., & OReilly, N. J. (2005). Synthetic Peptides as Antigens for Antibody Production. In *Immunochemical Protocols* (pp. 13–25).

Harlow, E., & Lane, D. (1999). *Using Antibodies: A Laboratory Manual*.

Hedley, J. (2016.). jsoup Java HTML Parser, with best of DOM, CSS, and jquery. Retrieved June 15, 2016, from https://jsoup.org/

Hoge, S. (2003). *Peptide Antigen Design for Antibody Production*. Sigma. Retrieved from http://www.sigmaaldrich.com/content/dam/sigma-aldrich/docs/Sigma/General_Information/peptide_design.pdf

Hornbeck, P. V., Kornhauser, J. M., Tkachev, S., Zhang, B., Skrzypek, E., Murray, B., … Sullivan, M. (2012). PhosphoSitePlus: a comprehensive resource for investigating the structure and function of experimentally determined post-translational modifications in man and mouse. *Nucleic Acids Research*, *40*(Database issue), D261–70.

HTML Tutorial. (2016.). Retrieved June 15, 2016, from http://www.w3schools.com/html/

Janeway, C. A. (2001). *Immunobiology: The Immune System in Health and Disease*. Taylor & Francis Group.

Janin, J., Joël, J., Shoshanna, W., Michael, L., & Bernard, M. (1978). Conformation of amino acid side-chains in proteins. *Journal of Molecular Biology*, *125*(3), 357–386.

JComponent (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/javax/swing/JComponent.html

JDK - javaws.jar - JDK 1.6.0_06 Java Web Start. (2016.). Retrieved June 15, 2016, from http://jar.fyicenter.com/199_jdk_javaws.jar_jdk_1.6.0_06_java_web_start.html

JEditorPane (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/javax/swing/JEditorPane.html

Jewell, T., Power, D., Power, D., Pierson, R. M., Razani, A., Pierson, R. M., … Lawrence, C. (2013, April 16). Why Cloud Development Environments Are Better

Than Desktop Development - ReadWrite. Retrieved June 15, 2016, from http://readwrite.com/2013/04/16/why-cloud-development-environments-are-better-than-desktop-development/

Jmol: an open-source Java viewer for chemical structures in 3D. (2016.). Retrieved June 15, 2016, from http://www.jmol.org/.

JPred: A Protein Secondary Structure Prediction Server. (2016.). Retrieved June 15, 2016, from http://www.compbio.dundee.ac.uk/jpred/api.shtml

JTextComponent (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/javax/swing/text/JTextComponent.html

Karplus, P. A., & Schulz, G. E. (1985). Prediction of chain flexibility in proteins. *Die Naturwissenschaften*, *72*(4), 212–213.

Katoh, K., Misawa, K., Kuma, K.-I., & Miyata, T. (2002). MAFFT: a novel method for rapid multiple sequence alignment based on fast Fourier transform. *Nucleic Acids Research*, *30*(14), 3059–3066.

Kolaskar, A. S., & Tongaonkar, P. C. (1990). A semi-empirical method for prediction of antigenic determinants on protein antigens. *FEBS Letters*, *276*(1-2), 172–174.

Larsen, J. E. P., Lund, O., & Nielsen, M. (2006). Improved method for predicting linear B-cell epitopes. *Immunome Research*, *2*, 2.

Leenaars, M., & Hendriksen, C. F. M. (2005). Critical Steps in the Production of Polyclonal and Monoclonal Antibodies: Evaluation and Recommendations. *ILAR Journal / National Research Council, Institute of Laboratory Animal Resources*, *46*(3), 269–279.

Linding, R., Jensen, L. J., Diella, F., Bork, P., Gibson, T. J., & Russell, R. B. (2003). Protein disorder prediction: implications for structural proteomics. *Structure* , *11*(11), 1453–1459.

Linding, R., Russell, R. B., Neduva, V., & Gibson, T. J. (2003). GlobPlot: Exploring protein sequences for globularity and disorder. *Nucleic Acids Research*, *31*(13), 3701–3708.

LinkedHashMap (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/java/util/LinkedHashMap.html

Lipman, N. S., Jackson, L. R., Trudel, L. J., & Weis-Garcia, F. (2005). Monoclonal versus polyclonal antibodies: distinguishing characteristics, applications, and information resources. *ILAR Journal / National Research Council, Institute of Laboratory Animal Resources*, *46*(3), 258–268.

McGuffin, L. J., Bryson, K., & Jones, D. T. (2000). The PSIPRED protein structure prediction server. *Bioinformatics* , *16*(4), 404–405.

Miller, F., & Frederick, M. (1995). Immunobiology: The Immune System in Health and Disease. Charles A. Janeway, Jr. Paul Travers. *The Quarterly Review of Biology*, *70*(2), 257–258.

Milstein, C. (1999). The hybridoma revolution: an offshoot of basic research. *BioEssays: News and Reviews in Molecular, Cellular and Developmental Biology*, *21*(11), 966–973.

Notredame, C., Higgins, D. G., & Heringa, J. (2000). T-Coffee: A novel method for fast and accurate multiple sequence alignment. *Journal of Molecular Biology*, *302*(1), 205–217.

Olson, S. A. (1994). MacVector: an integrated sequence analysis program for the Macintosh. *Methods in Molecular Biology* , *25*, 195–201.

Parker, J. M., Guo, D., & Hodges, R. S. (1986). New hydrophilicity scale derived from high-performance liquid chromatography peptide retention data: correlation of predicted surface residues with antigenicity and X-ray-derived accessible sites. *Biochemistry*, *25*(19), 5425–5432.

Parkin, J., Jacqueline, P., & Bryony, C. (2001). An overview of the immune system. *The Lancet*, *357*(9270), 1777–1789.

Pathak, S., & Palan, U. (2005). *Immunology: Essential and Fundamental*. Science Publishers.

Patient, S., Wieser, D., Kleen, M., Kretschmann, E., Jesus Martin, M., & Apweiler, R. (2008). UniProtJAPI: a remote API for accessing UniProt data. *Bioinformatics* , *24*(10), 1321–1322.

Pellequer, J. L., Westhof, E., & Van Regenmortel, M. H. (1993). Correlation between the location of antigenic sites and the prediction of turns in proteins. *Immunology Letters*, *36*(1), 83–99.

Pisitkun, T., Hoffert, J. D., Saeed, F., & Knepper, M. A. (2012). NHLBI-AbDesigner: an online tool for design of peptide-directed antibodies. *American Journal of Physiology. Cell Physiology*, *302*(1), C154–64.

Ponnuswamy, P. K., Prabhakaran, M., & Manavalan, P. (1980). Hydrophobic packing and spatial arrangement of amino acid residues in globular proteins. *Biochimica et Biophysica Acta*, *623*(2), 301–316.

Potel, M. (2011, May). MVP: Model-View-Presenter The Taligent Programming Model for C++ and Java. Retrieved from http://www.wildcrest.com/Potel/Portfolio/mvp.pdf

Rahman, K. S., Chowdhury, E. U., Poudel, A., Ruettger, A., Sachse, K., & Kaltenboeck, B. (2015). Defining species-specific immunodominant B cell epitopes for molecular serology of Chlamydia species. *Clinical and Vaccine Immunology: CVI*, *22*(5), 539–552.

Rajewsky, K. (1996). Clonal selection and learning in the antibody system. *Nature*, *381*(6585), 751–758.

Rosano, G. L., & Ceccarelli, E. A. (2014). Recombinant protein expression in Escherichia coli: advances and challenges. *Frontiers in Microbiology*, *5*, 172.

Saha, S., Sudipto, S., & Raghava, G. P. S. (2004). BcePred: Prediction of Continuous B-Cell Epitopes in Antigenic Sequences Using Physico-chemical Properties. In *Lecture Notes in Computer Science* (pp. 197–204).

Saha, S., Sudipto, S., & Raghava, G. P. S. (2006). Prediction of continuous B-cell epitopes in an antigen using recurrent neural network. *Proteins: Structure, Function, and Bioinformatics*, *65*(1), 40–48.

Saha, S., Sudipto, S., & Raghava, G. P. S. (2007). Prediction Methods for B-cell Epitopes. In *Methods in Molecular Biology* (pp. 387–394).

Shi, S. R., Cote, R. J., & Taylor, C. R. (1997). Antigen retrieval immunohistochemistry: past, present, and future. *The Journal of Histochemistry and Cytochemistry: Official Journal of the Histochemistry Society*, *45*(3), 327–343.

Siddiqui, M. Z. (2010). Monoclonal antibodies as diagnostics; an appraisal. *Indian Journal of Pharmaceutical Sciences*, *72*(1), 12.

Sievers, F., & Higgins, D. G. (2014). Clustal Omega, accurate alignment of very large numbers of sequences. *Methods in Molecular Biology* , *1079*, 105–116.

Stephen Stoker, H. (2015). *Organic and Biological Chemistry*. Cengage Learning.

Surhone, L. M., Tennoe, M. T., & Henssonow, S. F. (2010). *Htmlunit*. Betascript Publishing.

Tamura, T., & Chiba, J. (2009). Production of antibodies against multipass membrane proteins expressed in human tumor cells using dendritic cell immunization. *Journal of Biomedicine & Biotechnology*, *2009*, 673098.

T Cell Tools. (2016.). Retrieved June 15, 2016, from http://tools.iedb.org/main/tcell/

The UniProt Consortium. (2014). UniProt: a hub for protein information. *Nucleic Acids Research*, *43*(D1), D204–D212.

Thompson, J. D., Gibson, T. J., & Higgins, D. G. (2002). Multiple Sequence Alignment Using ClustalW and ClustalX. In *Current Protocols in Bioinformatics*.

TimerTask (Java Platform SE 7). (2016.). Retrieved June 15, 2016, from https://docs.oracle.com/javase/7/docs/api/java/util/TimerTask.html

Tools-API. (2016.). Retrieved June 15, 2016, from http://tools.iedb.org/main/tools-api/

Trier, N. H., Hansen, P. R., & Houen, G. (2012). Production and characterization of peptide antibodies. *Methods* , *56*(2), 136–144.

UniProtHit (UniProt JAPI 1.0.6 API). (2016.). Retrieved June 15, 2016, from https://www.ebi.ac.uk/uniprot/japi/javadoc/uk/ac/ebi/uniprot/dataservice/client/alignment/blast/UniProtHit.html

Uversky, V. N., Kuznetsova, I. M., Turoverov, K. K., & Zaslavsky, B. (2015). Intrinsically disordered proteins as crucial constituents of cellular aqueous two phase systems and coacervates. *FEBS Letters*, *589*(1), 15–22.

Velankar, S., Dana, J. M., Jacobsen, J., van Ginkel, G., Gane, P. J., Luo, J., … Kleywegt, G. J. (2013). SIFTS: Structure Integration with Function, Taxonomy and Sequences resource. *Nucleic Acids Research*, *41*(Database issue), D483–9.

von Heijne, G. (2006). Membrane-protein topology. *Nature Reviews. Molecular Cell Biology*, *7*(12), 909–918.

Waterhouse, A. M., Procter, J. B., Martin, D. M. A., Clamp, M., & Barton, G. J. (2009). Jalview Version 2--a multiple sequence alignment editor and analysis workbench. *Bioinformatics* , *25*(9), 1189–1191.

Weaver, W. (1970). Molecular Biology: Origin of the Term. *Science*, *170*(3958), 581–582.

Winter, G., Greg, W., Griffiths, A. D., Hawkins, R. E., & Hoogenboom, H. R. (1994). Making Antibodies by Phage Display Technology. *Annual Review of Immunology*, *12*(1), 433–455.

Wurm, F. M. (2004). Production of recombinant protein therapeutics in cultivated mammalian cells. *Nature Biotechnology*, *22*(11), 1393–1398.