**TANIYA PAL [S373094]**

**MODULE SIX ASSIGNMENT**

**DOCUMENTATION**

**Classes:**

* FastaParser
* OMSSAParser
* PeptidePanel

**FastaPasrser**

Packages:

“FastaParser” mainly reads the FASTA file given by the user as input. It uses Java packages such as “File”, “FileNotFoundException”, “Scanner”, “IOException”, “HashMap” for reading the fasta file.

Constructor and Methods:

It consists of a constructor “FastaParser” and a method “FastaLineReader”. The constructor “FastaParser” checks if the FASTA file provided by the user exists, if not prints out accordingly “Fasta file not found!”.

“FastaParser” also consists of a method called “FastaLineReader” which uses a Java package Scanner, which helps to read the file one line at a time. The method loops through the fasta file until it reaches the end and stores the header of the file in a variable named header and the sequence in a variable named sequence.

**OMSSAParser**

Packages:

OMSSAParser reads the OMSSA file given by the user as input. Java packages namely “CSVReader” , “File”, “FileNotFOundException”, “IOException”, “Arrays” and “List”.

Constructor and Methods:

It consists of constructors such as “OMSSAParser”, and method such as “OMSSAFileReader”. “OMSSAFileReader” uses the “readAll” method of “CSVReader” package in Java, to parse through the whole csv file and store it in a variable named body. While the constructor “OMSSAParser” checks if the “OMSSAfile” provided by the user exists, if not, prints out accordingly “OMSSAfile not found!”.

**PeptidePanel**

Constructor and Methods:

It consists of two methods, “addPositions” and “paintComponent”.

The addPosition method is responsible for taking the start and stop postions from the omssa file and adding the corresponding values to the arrays named “start” and “stop”. The method also checks if the number of values in the mentioned arrays are same, or else prints out error.

The paintComponent method is responsible for drawing the peptide in the jpanel taking the values of the start and stop positions from the addPosition method, and accordingly adds the black boxes (representing peptides) along the rectangle ( representing the protein).

**Interface:**

The **Jframe MyFrame** consists of a

* jtextpane called proteintextpane,
* jtable called jTable1
* jpanel called PeptidePanel
* menubar which contains JMenuItem1

The “JMenuItem1” menu item named “Open” opens a File Dialog box, from where the user can choose the FASTA and OMSSA files which they want to display in the peptide visualization tool. The file directory of the file chosen by the user is store in the variable “fileDirectory” in “JMenuItemActionPerformed” class from where filename of the FASTA or OMSSA file chosen by the user is extracted and stored in variable called “filename”.

If the file is a FASTA file, the file is parsed through FastaPaser class and the sequence of the fasta file without the header stored in the variable “sequence” is displayed in the JTextPane.

If the file is a OMSSA file, the file is parsed through OMSSAParser class and the contents of the file are displayed in the jtable.

The contents of the column of the OMSSA file which contains the peptide sequences of the same protein whose fasta sequence the user chooses to display in the textpane are extracted and highlighted in the textpane in yellow colour.

Also, the start and stop positions of the peptide are extracted from the OMSSA file and that is utilized to draw the protein diagram representing the position of the peptide in the protein in the jpanel.

Diagram

Description automatically generated

Figure 1: UML Diagram of the Peptide Visualization tool source code

**TANIYA PAL [S373094]**

**MODULE SIX ASSIGNMENT**

**USER MANUAL**

**About the tool:**

The peptide visualization tool helps the user to parse OMSSA (.csv) files and FASTA (.fasta) files. After parsing, the user can visualize the omssa file in the tool and the peptides from the omssa files that get are present in the fasta sequence are highlighted for the user. Also a diagram of the corresponding peptide positions in the protein can also be viewed in the tool.

**Choosing a FASTA file:**

On opening the visualization tool, to choose the FASTA file, the user needs to click on File then Open then select a “.fasta” file only from the user’s computer. After choosing the FASTA file, it gets displayed in the right text pane.

**Choosing an OMSSA file:**

To choose the OMSSA file, the user needs to click on File then Open and select a “.csv” file only from the user’s computer. After choosing the OMSSA file, it gets displayed on downward side of the tool.

**Output:**

The matching sequences of the peptides in the OMSSA file and fasta file gets highlighted in yellow color. A corresponding diagram of the protein representing the positions of the peptides on the protein gets displayed on the left side of the tool.

**Reference to the imported classes:**

import java.awt.Color

import java.awt.Graphics

import java.io.File

import java.io.FileNotFoundException

import java.io.IOException

import java.util.HashMap

import java.util.Scanner

import java.awt.Color

import java.awt.Component

import java.awt.FileDialog

import java.io.FileNotFoundException

import java.io.IOException

import java.util.ArrayList

import java.util.Arrays

import java.util.Vector

import java.util.logging.Level

import java.util.logging.Logger

import java.util.regex.Matcher

import java.util.regex.Pattern

import javax.swing.JScrollPane

import javax.swing.JTable

import javax.swing.table.DefaultTableCellRenderer

import javax.swing.table.DefaultTableModel

import javax.swing.table.JTableHeader

import javax.swing.table.TableColumn

import javax.swing.table.TableColumnModel

import javax.swing.table.TableModel

import javax.swing.text.BadLocationException

import javax.swing.text.DefaultHighlighter

import javax.swing.text.Document

import au.com.bytecode.opencsv.CSVReader

import java.io.File

import java.io.FileNotFoundException

import java.io.FileReader

import java.io.IOException