Assignment report, Java

MSc Applied Bioinformatics,

Marie Schmit

Relations between classes

The GUI architecture is made of a main frame. When launched, it creates panels with which the user interacts. The first allows the user to choose a file, before redirecting to both actionPanel and displayResultsPanel. Action panel is a menu, which options depend on the type of the file chosen (either fasta or gtf).

When display of file content (either in table for gtf, or text for fasta) is chosen, functions of the class displayResultsPane shows its content on the display panel. For statistics calculation, instances of either fastaStatistics or gtfStatistics are created in action panel. Their methods calculate the required statistics, displayed in instance displayResultsPane.

When exons display is chosen, an instance of the panel exonsPanel is displayed. It contains an instance of a child class of fileChooserPanel: secondFileChooserPanel, as well as a text panel and an instance of class exonsPanel. The first panel to be visible is secondFileChooserPanel, in which the user can choose a second file, used to cross-reference the information of the previous one.

When second file is selected for textual representation, a function to display text exons is called from an instance of exonsPanel. This function uses functionalities of another class: exons, which contains methods to calculate the start indexes of the nucleotides in the fasta sequence that are contained in the exons indicated in the gtf file. For graphical display, an instance of the panel graphicExons is called, itself declared in exonsPanel class. The methods of this class graphicExons allows to display rectangles at the indexes and length also indicated by methods of the class exons.

Classes

mainframe

MainFrame is extended from JFram class, imported from *javax.swing*. It calls three instances of the following panels: actionPanel, displayResultsPanel and fileChooserPanel.

fileChooserPanel

A text area allows the user to write a file path, defined as search terms of the file when the button “Browse file” is pressed. This action opens a new frame (*java.awt.Frame*) in mode FileDialog (*java.awt.FileDialog*). When a file is chosen, its name is saved and buttons “clear” and “confirm” are set to visible. “Clear” resets some text labels, while “confirm” launches the file reading. A BufferedReader and a FileReader are created for the chosen file name (*java.io.BufferedReader, java.io.FileReader*). Each line is added to the file reader, then to an ArrayList of StringBuffer (*java.lang.StringBuffer, java.util.ArrayList*). Those formats where chosen to store file content because they are flexible with size of the. If the file is not found or empty, exceptions are raised (*java.io.FileNotFoundException, java.io.IOException*). The type of file is checked with regular expressions (*java.util.regex.Pattern*). Then, the user access actionPanel and displayResultsPanel instances, also created in the main frame. A method from actionPanel is called, to display the menu corresponding to the type of the chosen file. The content of the file is passed to the instance actionaPanel.

ActionPanel

This class extends *java.swing.JPanel*. At instance creation, the design of its internal frame is set with *javax.swing.plaf.basic.BasicInternalFrameUI* and *java.awt.Container*. The buttons “display text” or “table” call methods from the class displayResultsPane. The buttons to display fasta or gtf statistics creates instances of fastaStatistics and gtfStatistics, which methods (getMinMaxLength(), statisticSeqLength()) calculates statistics that are displayed with displayResultsPane instance

(displayText() method). Their results are stored in HashMap (*java.util.HashMap*). For exons display, methods of objects from exonsPanel and secondFileChooser class are called, and values of file chosen are passed to instance of secondFileChooser.

For every button, the corresponding panels in displayResultsPanel are set to visible (setPanelVisbile()), and necessary information are passed to every instance of used classes thanks to their "set" functions.

DisplayResultsPane

This class extends *java.swing.JPanel*. Three panels are set in a JLayeredPane for either text, table or exons display, with a call of exonsPanel instance. The method maxNumberPages calculates how many pages are needed to display the fasta or gtf documents. One page contains thousand lines. If numerous pages are needed, buttonsPanel is visible. When its Next or previous buttons are pressed, the current page number increases or decreases and the displayText() is called to refresh the page. The "Go to page" button sets the current page number to the one written by the user, if it stays within the range of displayable pages for the document.

displayText() calculates the indices of the start and end lines (*java.util.ArrayList*) for the current page. For fasta files, it calls displayPage(), that happens the lines of the file to the text area, when they are within those start and end indexes. For gtf files, displayTablePage is called. This method creates an instance of DefaultTableModel (*javax.swing.table.DefaultTableModel*) for the table that is in tabPanel (*javax.swing.JTable*). Each line of the gtf file is added to the table (addToTable). Then, the size of the columns are adapted to the text (setColSize()).

DisplayText is also overridden to display strings in the text area.

FastaStatistics

Java.util.ArrayList is imported to read the content of the analysed fasta file.

numberSequence() calculates the number of sequences in the file to determine if it contains a single or multiple sequences.

statisticSeqLength() calculate the sequence length (or average). For single sequence, listLength() is called: if the file is not empty, it goes through the lines of the file that are not annotations and save their length in a returned list. Those values are then summed up in statisticSeqLength(). For multiple sequences file, averageLengthSequence() calculates the average of the list generated by listLength().

This method statisticSeqLength() returns an instance of a nested class lengthResult. This class stores the length values of the sequences, and their type (single or multiple).

getGC() calculates GC content of the file. It calls numberGC() to get a list of the number of G and C in each sequence: for lines that are not annotations, numbers of G or C characters are count. Those numbers are summed up. The division G/C is calculated if C is not null. If it is, an exception is thrown (IllegalStateException).

gtfStatistics

hashLine returns a hashMap (*java.util.HashMap*) of a line of gtf file. Each element of the line is a value, which are easily retrieved thanks to their keys.

averageExon calculates the average number of exons in each gene. First, setExonsPerGene is called to return a hashMap of the number of exons per gene: every line of the gtf file is stored in a hashMap with hashLine(), another hashMap exonsPerGene is created to contain Gene IDs and their

corresponding number of exons. The method actualiseNumberofExons() is called: for each key Gene ID, the corresponding number of exons, increases. averageExon() goes through this HashMap (*java.util.Map*) and calculate their average. An exception is thrown is the number of genes is null (IllegalStateException).

getMinMaxLength() calculates the longest and shortest gene models within a gtf file. Each line of the file is converted to a hashMap, its length calculated with getLength(): for a given feature (here "gene"), the length is calculated with start and end values given by the line HashMap. Those length are stored in a list. Its minimum, maximum and average (AverageLength() method) are calculated.

exonsPanel

This class extends *javax.swing.JPanel.* This panel opens on an instance of secondFileChooserPanel.

changeCardPanel() uses CardLayout to switch between textExons and graphicExons instances (*java.awt.CardLayout*).

exonsText() coordinates textExonsSingle() and textExonsMultiple() methods. They call for each line the methods textLineExons() that sets the style of the text (*javax.swing.text.StyledDocument*, *javax.swing.text.SimpleAttributeSet, javax.swing.text.StyleConstants*) to cyan color (*java.awt.Color*) for a given set of start indexes and length values. Those values are calculated with getSingleColoration() or getMultipleColoration(), from exons instance.

secondFileChooserPanel

This class is a child of fileChooserPanel.

checkFileType(), called in override method confirm(), checks that the type of the chosen file is either gtf or fasta, and is different then the type of the first chosen file. It calls methods textualDisplay() or graphicalDisplay() according to the type of display chosen by the user.

textualDisplay() sets the card panel to textual and calls exonText() method from exonsPanel.

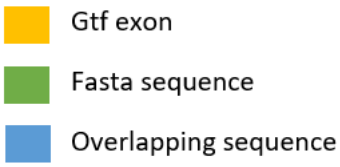
graphicalDisplay() sets the card to graphical and creates an instance of exons to calculate the dimensions of exons rectangles (getSingleColoration()), and parse the fasta annotation(parseAnnotation). It passes those values to the instance graphicExons of exonsPanel (setAnnotation() and setCoordinates()) and repaints this panel.

Exons

parseAnnotation() splits a fasta annotation in a list of chromosome, start and end values, to retrieve the start and length of a fasta sequence.

lineCoordinate() calculates the coordinates of the sequence line drawn between exons, using information from the parsed fasta annotation (*java.util.ArrayList*). A coefficient is applied to the start and length values, for scaling. If the width of the line is too large to be contained within a panel, the coordinates are recalculated to make a line break.

getSingleColoration() scans each line of gtf content, parse them into a hashMap (*java.util.*HashMap) and calls isGtfExons(). isGtfExons() checks that the gtf line is an exons, corresponds to the chromosome of fasta file and overlaps this sequence. If yes, colorationIndex() is called, to return a list of start and length of the exon sequence. Each list is stored in a matrix.



- Gtf starts after fasta sequence: start of overlapping sequence is gtf start.
- Fasta sequence ends before gtf sequence: end of overlapping sequence is fasta end.

- Gtf starts before fasta sequence: start of overlapping sequence is fasta start.
- Fasta sequence ends after gtf sequence: end of overlapping sequence is gtf end.

Figure 1 End and start of overlapped exons calculation (exons contained in both gtf file and fasta sequence)

getMultipleColoration() gets the coordinates of exons for every sequence of a multiple sequence fasta file. Each line annotation and its corresponding sequence are considered as a single sequence, which start and index are calculated via getSingleColoration(). The resulting lists are stored in an ArrayList.

exonsGraphical() returns the coordinates of each exon rectangle. It rescales the start and length values from isSingleColoratin() with the same coefficients as for parseAnnotation(). If the width of the exons sequence is too large to be contained in the panel, their coordinates are changed to display them on a next line.

graphicExons

This class is extended from *javax.swing.JPanel*.

Coordinates of rectangles to draw are set with setCoordinates() and setAnnotation().

paintComponents() is overridden to set a Graphic (*java.awt.Graphics*) and its color (*java.awt.Color*). Rectangles are drawn with coordinates and dimensions (*java.awt.Dimension*) read (java.util.ArrayList) from an instance of exons (method exonsGraphical()).