

# Assignment report, Java

MSc Applied Bioinformatics,

Marie Schmit

# Relations between classes

The GUI architecture is made of a main frame that calls three instances of the following panels: actionPanel, displayResultsPanel and fileChooserPanel. The first one is the only visible when the program is run. It 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 are activated, that show this 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, which are display in instance displayResultsPane, thanks to the display method of its class.

When exons display is chosen, an instance of a child class of fileChooserPanel is created, and 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.

Some methods of the parent class fileChooserPanel where overridden. For instance, 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*. When launched, it creates panels with which the user interacts.

## fileChooserPanel

A text area allows the user to write a file path, which is defined as search term of the file browser thanks to regular expressions (*java.util.regex.Pattern*), 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” reset 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 appened 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 content (the length of the file and of its line is unknown when they are read). If the file is not found or empty, exception are raised (*java.io.FileNotFoundException, java.io.IOException*). The type of the file is checked with regular expressions: it should end with .gtf or .fa. 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 actionPanel created in the mainframe.

## ActionPanel

This class is extended from *java.swing.JPanel*. When an instance of this class is created, one of it’s internal frame functionality, which is useless, is hidden thanks to *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 create instances of the classes fastaStatistics and gtfStatistics, which methods (getMinMaxLength(), statisticSeqLength()) calculate statistics that are displayed with displayResultsPane instance (displayText() method). They results are often stored in HashMap (*java.util.HashMap*). For exons display, methods of objects from exonsPanel and secondFileChooser class, created in displayResultsPanel, are called.

For every button, the corresponding panels in displayResultsPanel are set to visible (setPanelVisbile()), other are kept hidden, and necessary information are passed to every instance of other classes thanks to their “set” functions.

## DisplayResultsPane

This class also 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, with thousands of lines allowed in a page. If more than one page is needed, the panel buttonsPanel is visible. When its Next button is pressed, the current page number increases of one and the function displayText is called to refresh the page. Same for previous button with a decrease of current page number. The “Go to page” one set the current page number to the one written by the user, if this number is within the range of displayable pages for the document.

The method displayText takes in argument the text content (*java.util.ArrayList*), page number and type of file. It calculates the indices of the start and end lines for the current page. For fasta files, it calls displayPage, that happens the line of the file to the text area, when they are within those start and end indexes. For gtf files, it calls displayTablePage. This method create an instance of DefaultTableModel (*javax.swing.table.DefaultTableModel*) for the jTable1 table that is in tabPanel (*javax.swing.JTable*). It adds each line of the gtf file to the table (addToTable()), after conversion from StringBuffer to string with contentToString(). Then, the size of the columns are adapted to the text (setColSize()).

DisplayText is also overridden when its argument is a string, it displays it in the text area, allowing the instance of actionPane to call is to display statistic values.

## 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() is called when the sequence length (or average) are required in actionPanel. For single sequence, the listLength() is called: if the file is not empty (not 0 sequence), it go 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 returned with listLength().

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

getGC is called in actionPanel to calculate GC content of the file. It calls numberGC() to get a list of the number of G and C content in each sequence: for lines that are not annotations, numbers of G or C characters in the line 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*) or a line of gtf file. Its keys are: Sequence name, Source, Feature, Start, End, Score, Strand, Frame, Attributes, Gene ID. It allows to easily retrieve information in other methods of this class.

averageExon is called in actionPanel to calculate 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, a hashMap exonsPerGene is created to contain gene ID and their corresponding number of exons. If the gene ID of the gtf line is not in this map, it is added with 0 number of exons. The method actualiseNumberofExons is called: for each Gene ID, (key of the hashMap), the corresponding value, number of exons, increases by one. averageExon gets this hasmap of genes and corresponding number of exons, goes through it (*java.util.Map*) and sums the numbers of exons to calculate their average. An exception is thrown is the number of genes is null (division per 0, 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 of the gtf file. A list stores those length. The minimum, maximum and average (AverageLength() method) of this list of length are calculated.

## exonsPanel

This class extends from *javax.swing.JPanel.* It uses a CardLayout to switch between textExons and graphicExons instances (java.awt.CardLayout) with method changeCardPanel. This panel opens on an instance of secondFileChooserPanel. It is a child class of fileChooserPanel.

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. getSingleColoration or getMultipleColoration() are called from exons instance to calculate those values.

## secondFileChooserPanel

It gets the values of the first chosen file (setVariable) in actionPanel (*java.util.ArrayList* to handle files contents). The function 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() set the card panel to textual and calls exonText method from exonsPanel.

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

## Exons

parseAnnotation() retrives the start and length of a fasta sequence buy splitting its annotation in a list of chromosome name, start value, end value.

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 to get them to the right scale. 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(), to store is results in a matrix. 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: the end and start of the overlapping of fasta sequence and gtf exons provides the sequence length.

Bar chart

Description automatically generated

Figure End and start of exons calculation

getMultipleColoration() gets the coordinates of exons of every sequence of a multiple sequence fasta file. Each couple of line annotation and corresponding sequence are considered as a single sequence and their list of start index and length is calculated via getSingleColoration(). Those lists are stored in an ArrayList: each value of the ArrayList corresponds to a sequence indexes.

exonsGraphical() returns the coordinates of each exon rectangle. It uses the start and length values from isSingleColoratin() and rescale them 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().

The functionpaintComponents() is overriddento set a Graphics (*java.awt.Graphics*). Its color of exons is set (*java.awt.Color*) and the coordinates of the rectangles are read (java.java.util.ArrayList) from an instance of exons (methods exonsGraphical). Rectangles are drawn with those coordinates (*java.awt.Dimension*).