COM6516: Final assignment (60%)

3 December 2019; due Monday 13th January 2020 at 17:00

1 Task

The task for this assignment is to design and build a viewer for different types of works of fiction. In particular, a poem, a novel and a play. The viewer should load the documents from text files and the user should be able to ask for different interesting facts and stats about the document such as the author, the total number of words and the year it was published.

The due date is Monday 13 January 2020 at 17:00 (UK time, as indicated on MOLE/ Blackboard).

1.1 Specification

1.1.1 Obtaining Example Documents

Your program should be able to load documents from text files. I have provided three example documents: `daffodils.txt`, `PrideAndPrejudice.txt` and `Earnest.txt`.

1.1.2 Handling The Document Files

Each `.txt` file has at the top of the file some header information. Some of this information is the same for all of the different types of documents (such as 'Author') and some is different (such as the 'Number of acts' for the play).

1.1.3 Program GUI (basic version)

When the program starts, an initial window should be displayed with three areas.

The **first area** should have a textfield where the user can type in the name of the document file they want to load and a 'Load' button to cause the file to be loaded. You can assume that the document will be in the current folder, i.e., you do not need to ask for full path information.

Once the user has typed the filename, the program should load that file. This means firstly parsing the header information (including reading the type of document in the first line) and secondly reading the main body of the text.

You should store the document's information in an appropriate structure.

The **second area** should contain a 'Display Text' button. When you click this button you should check whether a document has already been loaded. If not, you want to display a warning message letting the user know that they need to load a document first (this could be a pop-up window). If you have loaded a document, you should open a new window and display the text (without the header information) in a way that enables the user to scroll through the lines (Hint: consider using `JScrollPane`) - make sure the user can't edit the text by accident!

The **third area** should contain a 'Show Stats' button. When that is clicked you should again check that you have already loaded a document (see above). If you have loaded a document, you should open a new window that displays some stats about the document. These facts will vary depending on what type of document it is. In addtion, you should print the total number of words in the main body of the text (that is all of the words from the word 'Text:' till the end of the .txt file. This includes counting words like 'SCENE' and 'ACT' in the play).

1.1.4 Extending the GUI (extra credit)

You may choose to extend the GUI by adding an extra statistics to the stats panel, namely to display the top 10 most frequent words in the text.

1.1.5 Implementation

The main class in your solution should be called DocumentViewer.java. Your solution should run using Java version 1.8. Your code should compile on the command line by executing the command "javac DocumentViewer.java" and run by executing the command "java DocumentViewer". (You may assume that the Sheffield package is accessible when your code is compiled and run.)

Your solution should only make use of the core Java Class Libraries (you may use JavaFX classes in addition to Swing and AWT classes). It should not make use of any external Java libraries (e.g. packages handling text) other than, optionally, the Sheffield package.

Your GUI should display the required data clearly. There should be one window that allows the user to select the input document and additional windows as specified above in the requirements. Credit will be assigned for GUIs that are easy to use, display the information in a way that is easy to understand and are aesthetically pleasing.

It is important that all of the windows are designed well. Think about things like choice of colour, font size and window size to make a useful and usable interface.

Sensible actions should take place, when the user closes the various windows and when the user specifies a *new* document to load.

Readability is of great importance for coding style. You should therefore take great care with line breaks, document your code appropriately and use indentation consistently. Before you hand in your code, ask yourself whether the code is understandable by someone marking it.

1.2 Hints and suggestions

Start by taking a look at the supplied .txt files using a text editor such as jEdit, and make sure that you understand how it is structured. Your program should be able to deal with this data in a sensible way.

When you design your solution you should consider separating the reading and storing of the document data from displaying it to the user. One approach would be to first read the header from the .txt file and then the body of text. You will want to store your body of text in a sensible collection class like an ArrayList. You can use the Sheffield package to read data from the file.

2 Rules

2.1 Unfair means

This is an individual assignment so you are expected to work on your own. You may discuss the general principles with other students but you must not work together to develop your solution. You must write your own code. All code submitted may be examined using specialized software to identify evidence of code written by another student or downloaded from online resources.

2.2 Submission

2.2.1 Deliverables: what to hand in

Upload your `*.java` files to MOLE/Blackboard using the assignments button on the menu bar. Do not submit any other files, such as `*.class` or `*.jar` files. The general expectation is that you will have a separate `*.java` for each class. If you choose to use the `sheffield` package you do not need to upload it, I will compile with it. This assignment does not require any other libraries other than the Java standard library, and no other libraries will be used when your code is compiled and executed for marking.

Your code should compile and run under Java 1.8 on the command line (make sure to check your code for inclusions of packages that I will not have access to!):

```
javac *.java
java NameOfMainClass
```

You can submit your work more than once; the last version submitted will be marked, so check it carefully. You will not be able to submit any code after the final deadline.

If you are unable to upload your code to MOLE for technical reasons, e-mail `heidi.christensen@sheffield.ac.uk` **in advance** to make arrangements for submitting it by e-mail.

3 Marking

Because of the much greater time available to work on this assignment (compared with the assessed labs), the *compiling* category constitutes a lower proportion of the mark, and the relative value of *comments, style, and readability* is greater.

Points	Categories and example scores	
10		**Compiling**
	10	It compiles with `javac *.java` with no modifications.
	6–9	All files compile with a few simple changes to the package structure (e.g., deleting `package` directives).
	1–5	Some files do not compile; or compiling requires more significant changes to the directory, file or package structures.
	0	It can't be compiled without modifying the code itself.
15		**Running**
	+2 points	Extra points if a significantly extended GUI has been implemented
	10–13	It runs and meets most of the specification, and retrieve and display data as required. There is a main window for selecting which document to display, and separate windows for displaying the document and for displaying various stats about it.
	1–9	It runs but deviates from the specification
	0	It doesn't run.
10		**GUI**
	10	A very good GUI with a main component that makes it easy to read the text and look at the different stats.
	7	A GUI of more basic quality
15		**OOP principles: encapsulation, class design, event handling**
10		**Comments, style, readability**
60		**Total**

Marks and feedback will be returned through MOLE/Blackboard within 3 weeks.