CE203 ASSIGNMENT 1 2018

Credit: 10% of total module mark

Submission Deadline: 11.59.59, Wednesday 14th November 2018

Submission will be via the Faser online submission system.

You should refer to the Undergraduate Students' Handbook for details of the departmental policy regarding late submission and university regulations regarding plagiarism; **the work handed in must be entirely your own**.

It is expected that marking of the assignment will be completed within 4 weeks of the submission deadline.

Introduction

This assignment involves writing two separate Java applications. **Both applications must display your registration number**. They must be hand-coded by you; if you use IntelliJ Idea or any other IDE you must not use the GUI Designer or any similar tool. The files should contain ***brief*** comments.

WARNING AND ADVICE ABOUT POSSIBLE ACADEMIC OFFENCES

Your solutions should be your own unaided work. You can make use of any of the programs from the CE203 lecture notes and the lab templates on Moodle. You may use any features from the Java Standard Edition Application Programming Interface (Jave SE API) including those not covered in CE203.

You must NOT use any third-party classes (e.g. classes that are not provided as part of the Java SE download). If you use any other sources, you must clearly indicate this as comments in the program, and the extent of the reference must be clearly indicated. For more information, please see the University pages on plagiarism and the Academic Offences Procedures.

DO NOT COPY PROGRAM CODE FOR THIS ASSIGNMENT FROM ANOTHER STUDENT OR FROM THE INTERNET OR FROM ANY OTHER SOURCES. DO NOT LET OTHER STUDENTS COPY YOUR WORK

Submission

Since you will be required to submit all of the files in a single folder you should not reuse any class names from exercise 1 in exercise 2.

Exercise 1 (50%)

A colour may be specified by its *RGB value* – the amount of red, green and blue it contains. These amounts are integers between 0 and 255. For example, red is represented by (255, 0, 0), blue by (0, 0, 255) and yellow (which is made up from red and green) by (255, 255, 0). White is (255, 255, 255) and black is (0, 0, 0). The `Color` class has a constructor with 3 arguments that allows a user to create a `Color` object by providing the RGB values.

Write, in a file called `CE203_2018_Ex1.java`, a frame-based application that allows the user to specify RGB values in three text fields, and, when a button is pressed, displays your registration number in the chosen colour. The button and the input fields should be placed at the *bottom* of the frame, and the text should be written on a panel positioned at the centre (on both the horizontal and vertical axis). The program should contain the following features:

a) The application should initially display the following welcome message (in blue) on the centre panel: “CE203 Assignment 1, submitted by: “(this should be followed by your registration number).

b) If, when the button is pressed, the content of any of the text fields is not an integer, the invalid field(s) should be cleared and an appropriate message should appear on the centre panel; text fields containing integers should, however, never be cleared.

c) If any of the text fields contain a value less than 0, the value 200 should be used in its place when generating the colour – the displayed value in the text field should also be changed to 200. Similarly, if any of the fields contains a number greater than 255, the value 255 should be used and displayed.

d) Add a “Reset” button at the *top* of the application which, if pressed, will result in all text fields being emptied and the initial welcome message being displayed.

e) Any possible exceptions should be captured by the program.

Exercise 2 (50%)

Write, in a file called `CE203_2018_Ex2.java`, a frame-based application that allows the user to store a list of words. The layout should be such that any buttons will be displayed at the top of the panel, user input is provided in a text field at the bottom and any system output will be displayed in the centre. The application should provide features and corresponding buttons that allow the user to

a) add a word to the list (the word to be added is provided in a text field at the bottom of the application).

b) display all of the words from the list that end with a specified letter (this method should be case-insensitive, e.g. no matter whether the user input is 'A' or ‘a’, all words ending with 'a' should be displayed)

c) search the list for a word (that is provided in the text field) and display how many times it is found and the positions in the list in which it is found; if however the text field is empty when this button is pressed, then the system should display the total number of all words in the list (together with an appropriate message)

d) remove from the list the last occurrence of a word

e) remove from the list all occurrences of a word

f) clear the list

The list may contain duplicate entries. Words may contain only letters, numbers, and hyphens (`-`) and must begin with a letter.

For each action specified above an appropriate message should be displayed on the centre panel confirming the action, e.g.

“Word ‘toast’ has been added to the list”.

An appropriate error message should be displayed if any of the actions failed, e.g.

“The string ‘12’ was not added to the list as it is not a valid word.”

The list should be implemented using a `LinkedList`.

You should aim to make your user interface easy to use.

Commenting the Programs

The programs should contain brief comments indicating precisely what each method does and what each instance variable is used for. You should not write any comments stating what individual lines of code do; in places it may be appropriate to state what a block of code does (e.g. “check that the input is a valid word”).

The programs should be laid out neatly with consistent indentation.

Program Testing Output

The programs produced for Exercise 1 and Exercise 2 should be fully tested to demonstrate that each of the features you are asked to implement in the programs GUI’s is working correctly.

You are therefore asked to include either a Word or PDF document where you clearly show for each exercise the correct output of your code. This can be shown as a series of screenshots that demonstrate the correct response of the GUI to each button press or user input / button press event.

For example, in Exercise 1 where the user must enter valid numbers for the R, G, B values in each text field, a screenshot of the output of the program showing correctly, the text displayed in the colour specified by the corresponding RGB values should be shown. Equally, for example, in Exercise 2 where you are asked to display all of the words from the list that end with a specified letter, you should display a screenshot of the GUI where this is correctly displayed following the button press event.

Please briefly explain or label the screenhot(s) for each feature being tested for exercises 1 and 2, for example ‘Ex 1 b)…..’

The Word of PDF document Should include your registration number and should be named according to the following naming convention:

Assignment 1_Testing_RegNo_<*registration number*>.

Submission

You are required to submit the source code files (`CE203_2018_Ex1.java` and `CE203_2018_Ex2.java`) and all of the `.class` files generated by the compiler. You need to also include the Word/PDF file containing your program testing output. All of the files should be placed in a single folder, which should then be *zipped* for submission to the online system, i.e. the submission should be a single file in *zip format*. **Other formats (e.g. *rar* format) and submissions without source code *will not be accepted* and will result in a mark of zero.**

Marks will be awarded for following *all* the requirements (including naming of files, layout of the interface etc.) and applying object-oriented programming principles appropriately.