School of Computing, Creative Technologies and Engineering

Assessment Brief

**Component 2**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Module name and CRN** | | | Software Development CRN: 18654 | | | |
| **Module Leader** | | | Mark Dixon | | | |
| **Semester** | A | **Level** | | 5 | **Approx No of Students** | 120 |

**COMPONENT TITLE: A Graphical Drawing Tool**

**COMPONENT WEIGHTING: 60% of Module Marks**

**HAND-OUT DATE:** Week 1 (w/c 26th September 2016).

**SUGGESTED STUDENT EFFORT:** 30 hours

**SUBMISSION DATE:** Tuesday 10th January 2017 at 23:00.

**SUBMISSION INSTRUCTIONS:**

Submission of program code should be via the VLE, followed by demonstration to the tutor within the subsequent lab session. Your solutions should be zipped up into a single file in the form <studentID>.zip. You MUST submit and demonstrate your work in order to receive a mark.

**FEEDBACK MECHANISM:**

You will be provided with verbal feedback regarding your work during the lab based demonstrations.

**LEARNING OUTCOMES ADDRESSED BY THIS COMPONENT:**

1. Develop non-trivial computer programs from a problem specification.
2. Identify and apply appropriate algorithms and data structures within a strongly typed programming language.
3. Use appropriate tools & techniques to develop and debug software based solutions.

**NOTES:**

**The usual University penalties apply for late submission.**

**This is an individual assessment. Submission of an assessment indicates that you, as a student, have completed the assessment yourself and the work of others has been fully acknowledged and referenced.**

**By submitting this assessed work, you are declaring that you are fit to submit, and you will therefore not normally be eligible to submit a request for mitigation for this work.**

**If you fail to attend the demonstration at the scheduled date and time without agreed mitigation, you will be given one further opportunity to demonstrate your work (incurring a 5% late penalty) at a time scheduled by the module team. If you miss this second opportunity, your result will be recorded as Non-Submission.**

**If your result for this assessment is recorded as Non-Submission or your mark for this assessment and for the whole module is below 40%, you will have opportunity to take reassessment with a submission date of 7th July 2017 (see Reassessment information below). If you are granted deferral through the mitigation process, you may complete the reassessment with a full range of marks available.**

**For further information, please refer to your Course Handbook or University Assessment Regulations.**

**DETAILS OF THE ASSESSMENT**

**Assignment Overview**

This assignment is based on the development of a graphical drawing tool. The basic concept is to produce a piece of software which reads in instructions from a text file, analyses the instructions, then presents a graphical drawing to the user. The solution should be developed using the Java programming language and the AWT/Swing libraries for the Graphical User Interface. No other 3rd party libraries outside those available within the standard Java Development Kit (JDK) should be used.

The drawing tool should take the form of a typical desktop type application. It will contain a menu bar, dialogue boxes and a main window area to be used for showing various representations of the data. The exact layout of the whole application is not prescribed, however details of the outputs to be shown are given and should be adhered to as close as possible.

This work has been split into four specific requirements as described below.

**Requirement 1 – Basic GUI and File loading**

The first requirement for this assignment is to implement the basic Graphical User Interface (GUI) that is capable of loading and displaying the instructions present within a text file. It should consist of an application frame, a menu bar and a main application panel split into two halves. One half should be a text panel that is capable of displaying a textual representation of the file being processed, and the other will (eventually) show the graphical drawing derived from interpreting the instructions.

The menu bar should consist of a 'File' and 'Help' menu. The File menu should include options for loading, saving and exiting the application. The 'Help' menu should contain an option for showing a dialogue box which identifies information about the application. At this point however the 'Save' option does not need to work.

The application should have the ability to actually load and display the instructional data. The 'File | Load' option should show a file open dialogue allowing selection of a data file. Once this is done the file should be opened, read and displayed within the text panel area of the GUI.

The application should be designed so that it uses layout managers where appropriate, and can be sensibly resized by the user. The menu options should also include short-cuts and icons where appropriate.

**Requirement 2 – Instruction Validation**

This requirement involves improving the application so that it examines the instructions within the text file for validity. Only very specific instructions should be accepted, exactly as detailed within Appendix A. This shows the instructions that are to be supported by the application.

The validation process should do the following for each line within the input file –

* Check that the instruction itself is known.
* Check that the correct number of parameters is provided for that instruction.
* Check that the values within the parameters are of an appropriate type.

If the instruction is valid then it should be shown in the text panel as before, however if something within a line is either missing or invalid then a suitable error message should be shown within the text panel at the point where that instruction would have appeared. This message should explain the nature of the error.

**Requirement 3 – Instruction Execution and Image Saving**

For the third requirement the ability to actually execute the instructions should be added. This will involve examining each valid instruction, then carrying out an appropriate action on the graphics panel.

Use the information provided in Appendix A to decide exactly what actions should be performed for each instruction.

An additional part of the requirement will be to provide an implementation for the ‘Save’ menu option. Add code to your application that allows a user to select a filename, via a dialog box, then saves the contents of the current graphics panel as an image file.

**Requirement 4 – Instruction Extension**

In order to satisfy the final requirement your job is to create additional Instructions within the system. What these Instructions do is your choice, but it should make sense in the context of the application. Adding new instructions may sound easy, but for each one keep the following in mind –

* Each instruction may have its own set of specific parameters that have to be processed.
* The validation process (Requirement 2) will have to be done for all new instructions.
* The execution process (Requirement 3) will also have to be done for all new instruction.

For example you may decide to add an instruction which can draw a rectangle. Hence you would not only have to add the ability to recognize ‘rectangle’ within your code, but you would also need to add code to validate the associated parameters, and also add the code to actually draw the rectangle.

You should aim to add at least five new Instructions in order to fulfill this requirement. Use your imagination a little, as more impressive Instructions are likely to attract more marks.

**MARKING SCHEME / CRITERIA**

The marking criteria are shown below. An initial mark will be established during the demonstration, but this may be subject to later moderation by the module team.

Note: The mark awarded is primarily for the explanation of the implemented solution as well as the quality of the submitted code. Hence submission of a working program that satisfies the requirements does not in itself guarantee marks. Marks will only be awarded following a satisfactory explanation of the code and response to questions during the demonstration.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Grade** | **80%+** | **1st**  **70%+** | **2.1**  **60% +** | **2.2**  **50+** | **3rd**  **40%** | **Fail**  **<40%** |
|  | All requirements completed.  Code is well formatted and all aspects are fully commented to a professional level  All output is presented to the highest standard possible.  Explanation during demonstration is impeccable showing in-depth knowledge. | Requirement 1-4 fully completed  Code is well formatted and commented.  Output is very well presented.  Explanations to all questions are clear and concise and additional knowledge related to the topic in general is apparent. | Requirement 1-3 fully completed  Code is generally well formatted and commented.  Output is well presented.  Explanations to all questions are clear and concise | Requirement 1 and 2 fully completed, with some attempt at requirement 3.  Code is generally understandable, but formatting could be better.  Output is adequate  Explanations to most questions are clear and concise | Requirement 1 is completed along with a reasonable attempt at least one other requirement  Code a little untidy but compiles.  Explanations to most questions are acceptable and generally correct. | Zero or only one requirement completed to a satisfactory level.  Code does may not compile.  Explanations to some questions are incorrect or reflect lack of knowledge regarding the solution. |

**REASSESSMENT and DEFERRAL OPPORTUNITIES**

**Reassessment** will take the form of resubmission, based on the original assignment specification.

In order to reach a minimum mark of 40% you must complete Requirement 1 and most of Requirement 2.

Upon submission of reassessment you must contact the module leader in order to arrange a demonstration. Details will be made available on the VLE.

**Appendix A – Instruction Summary**

The graphical drawing tool should use a coordinate system in which the 0,0 point is located at the top-left part of the graphics panel. Hence increasing X values move right and increasing Y values move down. The instructions are as follows-

|  |  |
| --- | --- |
| Instruction Syntax | MOVE <x> <y> |
| Description | Moves the current pen position to the given <x> and <y> position. |
| Parameters | Two, both Integer values. |
| Example: Move 100 pixels to the right and 200 pixels down. | MOVE 100 200 |

|  |  |
| --- | --- |
| Command Syntax | LINE <x> <y> |
| Description | Draws a line from the current pen position to the given <x> and <y> position. Following the command the pen position is updated so that it is positioned at the end of the drawn line. |
| Parameters | Two, both Integer values. |
| Example: Draw a square in the middle of the screen | LINE 200 150  LINE 300 150  LINE 300 250  LINE 200 250  LINE 200 150 |

|  |  |
| --- | --- |
| Command Syntax | CIRCLE <r> |
| Description | Draws an outline of a circle at the current pen position with a radius of <r>. The pen position is unchanged following this command. |
| Parameters | One, a positive Integer value. |
| Example: Draw a circle with a radius of 90 at the top left area of the canvas. | MOVE 100 100  CIRCLE 90 |

|  |  |
| --- | --- |
| Command Syntax | SOLID\_CIRCLE <r> |
| Description | Draws a filled circle at the current pen position with a radius of <r>. The pen position is unchanged following this command. |
| Parameters | One, a positive Integer value. |
| Example: Draw a circle with a radius of 90 at the top left area of the canvas. | MOVE 100 100  SOLID\_CIRCLE 90 |

|  |  |
| --- | --- |
| Command Syntax | CLEAR |
| Description | Clears the drawing area and resets the pen position to 0, 0. |
| Parameters | Zero |
| Example: | CLEAR |

|  |  |
| --- | --- |
| Command Syntax | COLOUR <red> <green> <blue> |
| Description | Sets the current pen colour used for drawing. |
| Parameters | Three, all integer values within the range of 0 to 255. |
| Example: Sets the current colour to magenta. | COLOUR 255 0 255 |

|  |  |
| --- | --- |
| Command Syntax | TEXT <text> |
| Description | Draws the text at the current pen position. The ‘y’ position of the current pen position is updated so future output is shown directly below. |
| Parameters | One, a textual value enclosed in double quotes. |
| Example: Outputs “hello world” and “back again” directly below. | TEXT “hello world”  TEXT “back again” |