School of Computing, Creative Technologies and Engineering

Assessment Brief

Component 1 COURSEWORK

Module name and CRN	Object Oriented Programming 21540				
Module Leader	Duncan Mullier				
Semester	2	**Level**	4	**Approx No of Students**	100

COMPONENT TITLE: **Lab Work**

COMPONENT WEIGHTING: **40% of Module Marks**

HAND-OUT DATE: **At time of beginning of the module**

SUGGESTED STUDENT EFFORT: **2 hours per session**

SUBMISSION DATE: **Assessed biweekly with final hand in on 5th May 12pm.**

SUBMISSION INSTRUCTIONS:

- All work is to be submitted to the OOP GitHub Classroom.
- To gain full marks each week's work must be committed within two weeks (this includes the flexible 5 day window)
- Weekly exercises that are late will be capped at 50%
- A link must be submitted to MyBeckett before the final deadline

FEEDBACK MECHANISM:
Feedback on the VLE.

LEARNING OUTCOMES ADDRESSED BY THIS COMPONENT:
Develop an understanding of the key object oriented concepts, such as classes, inheritance and polymorphism and have the ability to implement these in a modern object oriented language

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 your result 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 in July (see Reassessment information below). If you are granted deferral through the mitigation process, you may take the reassessment test with a full range of marks available.

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

Reassessment

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 phase test during reassessment in July (exact date to be announced nearer the time on OOP myBeckett page). If you are granted deferral through the mitigation process, you may complete the reassessment phase test with a full range of marks available.

DETAILS OF THE ASSESSMENT Task:

Each week's exercise should be completed as best as possible and committed to your GitHub repo within two weeks (i.e. week 1's exercises should be completed before Monday of week 3 etc.) (This extra week is to include the 5 day window, see flexible submission guidelines.) You are provided with weekly exercises as part of your OOP GitHub Portfolio. You should register with the GitHub Classroom by the end of week 1 and clone the repository. You should then keep up to date with the exercises. If you fail to keep up to date you will be provided with an opportunity to catch up with any late exercises but *Weekly exercises submitted after the window will be capped at 50%.*

The exercises are based on the course text book. **Java Foundations: Introduction to Programming and Data Structures, by John Lewis, Peter DePasquale and Joseph Chase. Pearson Addison-Wesley. ISBN: 0-321-48678-1**
This is available in physical and pdf format in the library, you are strongly encouraged to get it.

Click this link to accept the assignment.
https://classroom.github.com/a/giO1RSJf
Your last commit to your repository will be taken as your hand in, so don't commit again after the deadline.

MARKING SCHEME / CRITERIA

Credit will be given for trying to complete the exercises. They are fairly straightforward and self-explanatory.

All weeks carry equal weighting - there are 8 weeks, each contributing 5% to the overall module mark

School of Computing, Creative Technologies and Engineering

Assessment Brief

MAIN Component 2 COURSEWORK

Module name and CRN			*Object Oriented Programming 21540*		
Module Leader			*Duncan Mullier*		
Semester	*2*	***Level***	*4*	***Approx No of Students***	*100*

COMPONENT TITLE: ***Turtle Graphics Program***

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

HAND-OUT DATE: ***Week 1***

SUGGESTED STUDENT EFFORT: ***30 hours***

SUBMISSION DATE: **On or before 12pm 5 th May 2024 (week 13)**

You MUST demonstrate your work, via a recorded demo made according to the instructions below and uploaded to YouTube, to receive a mark. All code MUST be stored on GitHub as outlined below.

SUBMISSION INSTRUCTIONS: Submit via the VLE and GitHub Classroom, full instructions will be on the VLE in the assessment section. Read them carefully.

Do not commit any further commits after the deadline or it will be marked as late.

FEEDBACK MECHANISM:

Via the VLE.

LEARNING OUTCOMES ADDRESSED BY THIS COMPONENT:

LO1	*Develop an understanding of the key object oriented concepts, such as classes, inheritance and polymorphism and have the ability to implement these in a modern object oriented language.*

LO2	*Gain the ability to develop non-trivial computer programs made up of multiple files and classes in order to reflect modern object oriented based program architectures.*
LO3	*Apply design concepts using a suitable Object Oriented modelling notation.*

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

submit a demonstration and

demonstration Google form

and a GitHub portfolio repo

by the scheduled date and time without agreed mitigation, you will be given one further opportunity to demonstrate your work (incurring a 5% late penalty).

If you miss this second opportunity, your result will be recorded as Non-Submission. If your result 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 (see Reassessment information below). If you are granted deferral through the mitigation process, you may attend the reassessment demonstration with a full range of marks available.

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

Reassessment

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 (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. You must demonstrate your work during the week of TBA July (exact date to be announced on OOP myBeckett page nearer the time).

DEMONSTRATION

You are required to produce a demonstration video, using a script provided as a Google form. There are instructions as to how to do this (in terms of what software to use to produce your screen recording and the script you should use) on myBeckett. You MUST follow the instructions carefully and ensure you comply with them all as you may lose marks if you do not. They will be provided on myBeckett in detail but involve:

1..following the script provided on a google form whilst screen recording your software. You MUST submit the form and provide your GitHub link.

2..Submitting your software on the GitHub Classroom (link provided on myBeckett).

3..Uploading your screen recording demo to your student YouTube Account.

4..Pasting the link to your demo in the submission box when you submit your assignment on myBeckett and in the Readme.md file in your portfolio repo.

5..Provide you Git username and a valid link to your GitHub repository in your myBeckett submission.

6..Submit a portfolio of weekly exercises to your GitHub repository

Your GitHub repository should be in the GitHub classroom, the link is in the Assessment Information directory on myBeckett..

DETAILS OF THE ASSESSMENT MARKING SCHEME / CRITERIA

Your Task:

For this assignment you are required to develop a programmatic solution, using the Java programming language, to the problem below. ***You will be required to produce a YouTube demonstration video according to a provided Google Form Script, which MUST be shown being filled in in he video. The video must be uploaded to your student YouTube account and the link pasted into the submission box***. Note that some aspects of this assignment, notably the more advanced marks in a section, may require that you find out for yourself how to achieve a solution.

Overview

The overall aim of this assignment is to implement a simple graphics tool. This must be built as a graphical application using the Java Swing and/or AWT classes. The software will allow users to type in simple commands which cause a virtual pen (sometimes it is also called a turtle after the Logo programming language which was popular in schools in the 1980s) to move around a virtual canvas area drawing lines as it moves.

Requirement 1 – basic application 20 marks

The first requirement is to produce a Java project which usesLBUGraphics, which is provided in a jar file available on the OOP myBeckett page, under Assessments. This class will do all the drawing and displaying when your class calls its methods (ensure you always have the latest version of the jar file, as with any software it may be updated). There is also documentation for this class in the same place on myBeckett, which has simple instructions on how to use it. Note this documentation shows facilities that will be of use and some that may not be needed and seem confusing (just like the standard Java documentation).

Mark Breakdown

Project complete using the provided Jar file LBUGraphics.jar. Your project should also have a main class (MainClass.java) which sets up the LBUGraphics class correctly. You should call the “about()” method in LBUGraphics which will output a simple graphic.
(10 marks)

You should now make another class called TurtleGraphics.java that uses inheritance to extend the LBUGraphics class and it should function as above (note you can go straight to this and still get the marks for the above).

(5 marks)

Command Processing

You should take input from the console. When the user types “about” into the console it should call the LBUGraphic’s about() method, instead of automatically calling it.

(5 marks)

The LBUGraphics panel has a text field and a button. You should be able to respond to text typed into this text field. The LBUGraphics system will call your program’s “processCommand()” method when the user presses return on the text field or clicks the ok button. You should make it so that your program only calls the “about()” (which displays a simple graphic) method (to draw the dancing turtle” when the user types “about” into the text field and either presses return or clicks the “ok” button.

(5 marks)

Requirement 2 – command support 20 marks

The second requirement is to implement some basic commands to allow drawing. The user should be able to type in these commands within the text field provided by LBUGraphics. The application should be able to spot invalid commands and report this to the user. The commands to be supported are very explicit and MUST match those shown in the following table. The command MUST be typed in by the user and not selected from a menu as some of them will have parameters which MUST be typed together with the command, for example “forward 90”. The parameters MUST not be entered separately, either after the command or in a separate text field. Note that these commands involve calling the methods inside the LBUGraphics object. You should look at the documentation for LBUGraphics.

When the program first runs or the reset command is issued, the turtle/pen should be set to the middle of the canvas and point down the screen and the

pen should be set to "down". Hence if the first command was "forward 100" a line from the middle of the screen to nearer the bottom would be drawn.

Command	Description
`penup`	Lifts the pen from the canvas, so that movement does not get shown.
`pendown`	Places the pen down on the canvas so movement gets shown as a drawn line.
`left <degrees>`	Turn <degrees> to the left
`right <degrees>`	Turn <degrees> to the right.
`move <distance>`	Move forward the specified `distance`.
`reverse <distance>`	Move backwards the specified `distance`.
`black`	Sets the output pen colour to black.
`green`	Sets the output pen colour to green.
`red`	Sets the output pen colour to red.
`white`	Sets the output pen colour to white.
`reset`	Resets the canvas to its initial state with turtle pointing down but does not clear the display.
`clear`	Clears the display.

Note: A `<distance>` is an integer value, e.g. 100. The user does not surround the number with <>.

Turtle/Pen is in the middle of the screen pointing down

"penup" (2 marks)

"pendown" (2 marks)

"left" (3 marks) (1 mark only if no parameter)

"right" (3 marks) (1 mark only if no parameter)

"move" (3 marks) (1 mark only if no parameter)

"reverse" (2 marks) (1 mark only if no parameter)

at least four colour command (such as "red") (3 marks)

"reset" move the turtle to the initial position and pointing down (1 mark)

''clear" clears display, turtle stays in the same position (1 mark)

45

Requirement 3 Validating Commands. 15 Marks

Reports invalid commands (i.e. something not on the commands list above). (4 marks)
Detects valid command with missing parameter. (3 marks)
Detects non numeric data for a parameter. (4 marks)
Correctly bounds parameters (i.e. negative or non sensible values are reported as errors). (4 marks)

60

Requirement 4 – loading, saving and exiting. 15 marks

"Load" and "Save" should allow the user save and load the image and to save and load a set of commands that the user has typed in.

The current image should be saved to a file and also be able to be loaded back into the system and drawing recommenced. If the user attempts to load a new image without the current one been saved first then a warning should be shown to the user, which should provide the opportunity for the current image to be saved first.

Save Image (to suitable image graphics format such as png or jpeg). (2 Marks)
Load Image and redisplayed on the display. (2 marks)
Saves all commands previously typed as a text file. (2 marks)

- You may want to use a Java TextArea/JTextArea so the user can see all the commands but you can just save those that have “passed through” the text field.

Loads a text file of commands and executes them. (5 marks)

- If you haven’t used a JTextAea then they should be shown on the console.

Warning if the current image/commands is not saved. (2 marks)

Use of GUI dialogues. (2 marks)

75

Requirement 5 – extending LBUGraphics using inheritance. 20 marks

You have so far used LBUGraphics.jar, which is a precompiled file of a class I have written called LBUGraphics.java (technically a jar file is like a zip file and if you open it with a zip program you will see it has LBUGraphics.class inside it, or open it in the project explorer of Eclipse).

You have already used inheritance because you have extended the LBUGraphics class. This is because you have been forced to add a processCommands(String) method. Without putting it you would get a syntax error because LBUGraphics needs to call it when it detects an event on the button or text field. You can do much more with inheritance however. You can add new methods and you can replace existing methods.

All of the following should be implemented as methods in your code but also as commands that the user can type at run time.

1 Override the about() method so that it still does the same “dance” but appends a message that contains your name**.**

about

(4 marks)

2 Make a square command that takes a parameter for the length and draws a square. The turtle should remain in the original position after this command has been issued. It should “walk” around the perimeter of the square.

square <length>

(2 marks)

3 A pencolour Command that takes three parameters, one for red, one for green and one for blue to make an RGB colour.

pencolour <red>,<green>,<blue>

(5 marks)

4 A penwidth command that takes a width for the pen so that all further drawing operations are of the new width.

penwidth <width>

(2 marks)

5 A triangle command that draws equilateral triangles with one parameter.

triangle <size>

(3 marks)

6 A further triangle command which specified three sides that draws any triangle (hint – look at polygons).

triangle <side1>,<side2>,<side3>

(4 marks)

NOTE the reset command should reset all pen colours and widths back to what they were at the start.

95

Requirement 7 Do Something I Haven't Asked for. 10 marks.

It has to be something new and not more of the same (like another colour). Look at the documentation and see what it offers.

100

Deferral

If you have been given a deferral then you should complete the original assignment in full and provide a video recorded demo as described in the original assignment.

Reassessment

If you have been given a reassessment then you are to complete the original assignment to the following reduced specification.

You must use LBUGraphics.jar provided in the Assessment folder and install it correctly and use it to implement the following. Note it may be helpful to read the original assignment specification.

Note: A `<distance>` is an integer value, e.g. 100. The user does not surround the number with <>.

"penup"
"pendown"
"right"
"left"
"move <distance>"
"reverse <distance>"
at least four colour command (such as "red")
"reset" move the turtle to the initial position and pointing down
‘‘new" clears display, turtle stays in the same position

Your application needs to additionally:

Reports invalid commands.
Detects and reports valid command with invalid parameter.
Detects and reports non numeric data for a parameter.

You should record your YouTube video as per the original instructions using the original form and upload your video to YouTube and provide the link in the submission box of the assignment.