V1.0

LEEDS BECKETT UNIVERSITY

SCHOOL OF BUILT ENVIRONMENT, ENGINEERING & COMPUTING

COVER SHEET FOR ALL ASSIGNMENT BRIEFS

Name of Module	Object Orientated Programming
Name of Module Leader	Duncan Mullier
Main Assessment or Resit?	Main Assessment and Resits
Semester 1 or 2 or Term 1, 2 or 3	Semester 2
CRN	21540
Type of Assessment (Coursework; Presentation; Phase Test etc)	Coursework + Phase Test
Date & deadline time of Submission	Component 1:Scheduled lab session15/4/2024 (week 10) Component 2: 4pm 7th May 2024 (week 13)
Date for Return of feedback	4 weeks from submission date
Type of Submission (online via My Beckett; Handed in during Seminar; Presentation). It is expected that all assignments will be submitted electronically.	Online via MyBeckett
Feedback (please specify how will feedback be given to students)	Online and face-to-face in the labs
Franchise delivery Is the assessment for campus delivery the same for the franchise partner, if not please provide assessment for franchise partner.	N/A

School of Built Environment, Engineering and Computing

Assessment Setting Moderation Form

and

Assessment Brief

To be completed by the Module Leader:

Module name and CRN		Object Oriented Programming 21540			
Module Leader		Duncan Mullier			
Semester	2	**Level**	4	**Approx No of Students**	160

To be completed by the module leader:

Assessment components	**Part 1: Phase Test** - 40% **Part 2: Practical Work (program)** - 60% **Reassessment**: Complete either/both failed part(s) (cap of 40%). Coursework due by 16:00 on Monday 1 July 2024. **Deferral**: Opportunity to complete Parts 1 and 2 – full marks available.
Is the Assessment reused from the previous year?	Y and N It has been reworked. The library given has been totally rewritten.

To be completed by the internal moderator – please enter comments on each piece of assessment listed above:

Internal moderator's comments:-

I think your aim is to have students using github and regularly doing work. Github should be the one-stop show all for this, rather than a need for Google Forms.

There are 9 weeks of exercises. For the Portfolio of Exercises you are required to:

- have ONE completed and viewable on GITHUB
- to include an associated README.md file which contains the output of the code when run
- a commit timestamp needs to be within 2 weeks of the allocated Week (i.e. week 1 needs completing before the end of Week 3 etc)

Name of internal moderator Patrick Ingham		**Date 10/1/2024**	

To be completed by the module leader:

Response and action on internal moderator's comments (if required):- I have made the use of GitHub clearer and altered the form.			
Actioned by D Mullier		**Date 10/1/2024**	

May be completed by course director:

Comments:-			
Name		**Date**	

To be completed by the module leader (if necessary):

Response and action on comments (if required):-			
Actioned by		**Date**	

<ADD FURTHER RESPONSE FIELDS, IF NECESSARY>

To be completed by the external examiner:

External examiner's comments:-			
Name of external examiner		**Date**	

To be completed by the module leader:

Response and action on external examiner's comments (if required):-			
Actioned by		**Date**	

School of Computing, Creative Technologies and Engineering

Assessment Brief

Component 1 COURSEWORK

Module name and CRN		Object Oriented Programming 21540			
Module Leader		Duncan Mullier			
Term	3	**Level**	4	**Approx No of Students**	160

COMPONENT TITLE: **Phase Test**

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

HAND-OUT DATE: **at time of individual phase tests**

SUGGESTED STUDENT EFFORT: **2 hour session**

SUBMISSION DATE: **Scheduled session w/c 15/4 Week 10**

SUBMISSION INSTRUCTIONS:

- The phase test will take place on the VLE.

FEEDBACK MECHANISM:
Immediate feedback from 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 you fail to attend the test at the scheduled date and time (or arrive more than 30 minutes late) without agreed mitigation, 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 in July 2024 (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:

The phase test comprising of multiple choice/multiple answer questions of varying degrees of difficulty.

The test will be taken in your usual lab. The test is open book in that you may use your notes, myBeckett resources and Eclipse/IntelliJ (or other IDE). No communication systems such as email, Facebook, instant messaging are permitted and you are not allowed to use Google. If you fail to attend the phase test in your scheduled session and do not present acceptable mitigation to the mitigation pane you will be marked as "not submitted". Please note: Tutors will follow up any suspected unfair practice as per University policy.

Phase test 90% of component 1 marks

In addition there is a small test for each lecture. These will be enabled at the beginning of each lab session and a password provided.

Post lecture tests 10% of component 1 marks

NOTES: The usual University penalties apply for late submission. 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.

Student Instructions for Submission of Phase test

Your allocated time will be displayed at the top of screen, it is your responsibility to monitor your allocated time and submit your work within the time allocation. Once submitted; the work will be automatically marked and you should be able to see your grade.

Questions will be presented one at a time.

You can use your IDE, your notes and the Java documentation. You are not allowed to "Google" answers or use any communication system, such as email or instant messaging.

You will be given 90 questions each worth 1 mark (adding up to 90)

11 Questions from Variables

13 Questions from Selection and Iteration

11 Questions from Classes and Objects

14 Questions from Designing Classes

14 Questions from Inheritance

14 Questions from Polymorphism

13 Questions from Arrays

Questions are drawn at random for a larger pool.

MARKING SCHEME / CRITERIA

Questions will be marked as right or wrong by the VLE.

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 4pm 7th 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***

FEEDBACK MECHANISM:

Verbal feedback at demonstration.

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 2024 (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

The Library has information on your YouTube account.

https://libanswers.leedsbeckett.ac.uk/faq/179494

Your GitHub repository should be on 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 uses OOPGraphics, 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 OOPGraphics.jar. Your project should also have a main class (MainClass.java) which sets up the OOPGraphics class correctly. You should call the “about()” method in OOPGraphics which will output a simple graphic.
(10 marks)

You should now make another class called TurtleGraphics.java that uses inheritance to extend the OOPGraphics 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

The OOPGraphics panel has a text field and a button. You should be able to respond to text typed into this text field. The OOPGraphics 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 15 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 OOPGraphics. 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 OOPGraphics object. You should look at the documentation for OOPGraphics.

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.
`turnleft <degrees>`	Turn <degrees> to the left
`turnright <degrees>`	Turn <degrees> to the right.
`forward <distance>`	Move forward the specified `distance`.
`backward <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 (5 marks)

"penup" (1 mark)

"pendown" (1 mark1)

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

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

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

"backward" (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)

35

Requirement 3 Validating Commands. 10 Marks

Reports invalid commands (i.e. something not on the commands list above). (2 marks)

Detects valid command with missing parameter. (2 marks)

Detects non numeric data for a parameter. (3 marks)

Correctly bounds parameters (i.e. negative or non sensible values are reported as errors). (3 marks)

45

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)

Loads a text file of commands and executes them. (5 marks)
Warning if the current image/commands is not saved. (2 marks)
Use of GUI dialogues. (2 marks)

60

Requirement 5 – extending OOPGraphics using inheritance. 20 marks

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

You have already used inheritance because you have extended the OOPGraphics class. This is because you have been forced to add a processCommands(String) method. Without putting it you would get a syntax error because OOPGraphics 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.

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.

80

Requirement 7 Your Portfolio of Exercises 20

The weekly exercises on myBeckett and in the GitHub portfolio template are worth 20 marks in total. You should attempt as many as you can. You must edit the README.md file in the root directory of the portfolio to show how many of the exercises you have completed. You must also put your exercises in your GitHub repo on the GitHub Classroom. Failure to submit the Google form or GitHub repo will result in zero marks for this section.

For the Portfolio of Exercises you are required to:

- have ONE repo using the provideo portfolion link (in the Assessment Information directory) completed and viewable on GITHUB
- to include an associated README.md file which contains the output of the code when run
- a commit timestamp needs to be within 2 weeks of the allocated Week (i.e. week 1 needs completing before the end of Week 3 etc)
  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 OOPGraphics.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"
"turnright"
"turnleft"
"forward <distance>"
"backward <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.