COM00141M

Department of Computer Science

**ALGORITHMS AND DATA STRUCTURES**

ASSESSMENT BRIEF

|  |  |
| --- | --- |
| **Assessment Author** | Shaun Shei |
| **Assessment** | Summative - Open |
| **Release** | Week 1 |
| **Submission** | Monday of Week 5 at 13:00 (UK time) \* |
| **Feedback** | Within 30 working days of submission |
| **Weighting** | 30% |

\* If this date falls on a UK public holiday or a University of York closure day, the submission date will change. Please check the submission point in the ‘Assignments’ area of the module in Canvas for the exact submission deadline.

# Module Learning Outcomes

* **MO1**: express a problem solution algorithmically using pseudocode
* **MO2**: analyse the time complexity of an algorithm
* **MO3**: construct computer programs to implement algorithms
* **MO4**: test a computer program against the specification.

This assessment will contribute to learning outcomes **MO3** and **MO4** for this module.

# Assessment Brief

You are required to develop a single Java program with a **graphical user interface (GUI)** to erase and display different shapes based on user input of specific commands. You must use the same development environment as the one used in this module (**Java 11** and **JavaFX)** to implement the logic and GUI components.

The single program must be created inside one and **only one** file with exactly this title – “Shapes.java” (*without* quotes) – and have the following functionality:

1. **[Component 1]** A graphical interactive component which has the following actions associated with it: the component can accept user input, where assuming the input is valid, a shape will be displayed on the main screen. It is assumed that the program will eventually support a large variety of shapes. The program should currently accept the following input: **an integer between 3 and 5 inclusive**. Depending on the input, the program should generate a shape with the number of sides corresponding to the input value. For example, an input of 5 should generate a shape with five sides.
   1. You need to ensure that the user input is validated, and feedback is given when an invalid input is entered.
   2. The program should only allow the user to enter one valid input at a time.
   3. The main screen should only display one active shape at a time.
2. **[Component 2]** A graphical interactive component which has the following actions associated with it: the component must accept user input, where assuming the input is valid, the currently displayed shape will be manipulated depending on the validated input. In this case, the colour of the shape should be filled according to the validated user input. It is assumed that the program will eventually support a large variety of colours. The program should currently accept the following inputs: **green**, **red** and **grey**.
   1. You need to ensure that the user input is validated, and feedback is given when an invalid input is entered.
   2. The program should only allow the user to enter one valid input at a time.
   3. The main screen should only display one active shape at a time.
   4. The currently displayed shape should only be filled with one colour at a time.
3. Any potential error states should not crash the program or generate generic error messages. The program should clearly indicate to the user that an error has occurred, with specific information on why the error occurred and what valid action(s) is/are available.

# Assessment Tasks

Given the brief stated above, your tasks are as follows:

* Create one and only one Java class called ***Shapes.java***and implement the functionality from the brief stated above, ensuring that you have covered all points.
* Create a **1200-word** report with the following sections (please follow the report template provided at the submission point):
  1. **Implementation:** Outline of your approach taken to meet the functionality specified in the brief, with justifications and supporting evidence, e.g. source code evidence for appropriate attributes, method signatures and data types. Design and justification of GUI components for prompting and displaying details.
  2. **Functionality and Testing:** Outline of your strategy with justifications and supporting evidence to demonstrate that **ALL** the functionality specified in the brief has been tested. You also need to show evidence that the one and only one Javaclass called ***Shapes.java*** can be compiled and run **from the command line**.

**List of references:** Provide a correctly structured list of references to all the resources used for this development and report. Your responses should be appropriately supported by references to the literature and relevant resources using the Computer Science Department’s referencing standard (see [IEEE referencing style - a Practical Guide](https://subjectguides.york.ac.uk/referencing-style-guides/ieee)).

**Report word/page limit: 1200 words maximum. 8 pages in total, excluding list of references. Any work exceeding the word or page limit will not be marked.**

# Deliverables

Please submit the following:

1. A single **report** in PDF (or Word) format. **(Word and page limit: 1200 words in total, up to 8 pages, excluding list of references. Any work exceeding the word or page limit will not be marked).**
2. A single **archive** (.zip extension), containing one and only one file called exactly “Shapes.java” (without quotes).

**You should not be identified anywhere on your submission.**

**General submission guidelines:**

* Present your answers on A4 pages, with a minimum 12-point font (14-point for headings), minimum 120% line spacing (what Word calls “Multiple 1.08”), and margins of at least 2 cm on either side.
* All images must be of a size/quality that the details are viewable within the document. Images will not be scaled to identify blurred or small text.

**Canvas submission guidelines:**

* This assessment requires you to anonymously upload your submission to Canvas. If you are submitting multiple files, you must upload **all** files simultaneously to ensure that they are marked as a single submission.
* If you want to resubmit one component of your work, you need to re-upload all other files at the same time: every submission must include **all** the deliverables listed above.

# Marking Criteria

The marks for each section are detailed below. The maximum mark is 100. This will be scaled to 30% for the assignment overall.

|  |  |  |  |
| --- | --- | --- | --- |
| Learning Outcome | Section/Task | Criteria | Available Marks |
| MO3 | 1. Implementation | Outline of your approach taken to meet the functionality specified in the brief, with justifications and supporting evidence, e.g. source code evidence for appropriate attributes, method signatures and data types. Design and justification of GUI components for prompting and displaying details. |  |
|  |  | Functionality approach/justification: Component 1 | 10 |
|  |  | Functionality approach/justification: Component 2 | 10 |
|  |  | GUI design approach/justification: Component 1 | 10 |
|  |  | GUI design approach/justification: Component 2 | 10 |
| MO4 | 2. Functionality and Testing | Outline of your approach taken, with justifications and evidence that all functionality in the *Shapes* class and the options in the GUI have been tested and that they fully meet the requirements of the brief. |  |
|  |  | Overview of strategy for testing/demonstration of functionality | 10 |
|  |  | Testing/evidence of functionality: Component 1 | 10 |
|  |  | Testing/evidence of functionality: Component 2 | 10 |
|  |  | Testing/evidence of error handling: Component 1 | 10 |
|  |  | Testing/evidence of error handling: Component 2 | 10 |
| MO3 | General | Source code is well presented and self- documenting, with comments. Appropriate class, method and attribute naming – follows coding convention with appropriate indentations. | 10 |
|  | **TOTAL**: | | **100** |

**NOTE**: Failure to meet **ANY** of the following will result in **0 (zero)** marks for the assignment **as the Java file is tested to verify the contents in your report:**

**1.** A single report in PDF (or Word) format**. [Without the report, we cannot assess the submission as the code alone doesn’t provide a discussion around what decisions you took or justifications for why you have chosen your specific approach]**

**2.** Submission of one and only one file called exactly “Shapes.java” (without quotes) inside a single archive. **[Without the Java source code, we cannot compile and run the code to verify the contents in your report]**

# Grading

The pass mark for postgraduate modules is 50. For more information about grades and assessment criteria, please review the ‘Assessment and award’ section of the *York Online Programmes Handbook*, which is available to view or download from your **Orientation Module**.

# Assessment submission

You will submit your assessments in the ‘Assignments’ area of the module in Canvas.

Please check your Canvas module for the specific submission date for this assessment.

For general assessment guidelines consult your Canvas Module and Orientation Module, and for Academic Regulations, please see the University of York’s website.

Any queries regarding the details of your assessment should be directed to your module tutor. Any queries regarding assessment procedures should be directed to: [studentsuccess@online.york.ac.uk](mailto:studentsuccess@online.york.ac.uk)

# Submission deadline

The deadline for submitting the Practical Programming Assignment is the **Monday of Week 5, at 13:00/1pm (UK time).**

If this date falls on a UK public holiday or a University of York closure day, the submission date will change. Please check the submission point in the ‘Assignments’ area of the module in Canvas for the exact submission deadline.

# Submission of student work to Turnitin®

Turnitin® is a third-party text-matching software programme that compares written assessments with an online database of articles, books, websites, and pieces submitted by other users. To ensure the highest levels of academic integrity and in line with University Regulation 5.7b, any work submitted for this summative assessment will be submitted to Turnitin®. In accepting the University Regulations on admission, students have agreed to the University’s use of this software package.

You can also use Turnitin® as part of your writing process, to help you check your use of source information and improve your understanding of academic integrity. To access the Turnitin® submission points, or for more information on this software tool, see the **Turnitin Training** module in Canvas.

# Exceptional Circumstances Affecting Assessment

If unforeseeable and exceptional circumstances take place that prevent you from submitting a summative assessment by the deadline, or that negatively affect your performance in an assessment, then you can submit an Exceptional Circumstances (EC) claim. If approved, you may be granted a deadline extension, a late penalty waiver or the opportunity to sit a new version of the assessment 'as if for the first time' (known as a ‘SAIFFT’).

If unforeseeable and exceptional circumstances do occur, you must seek support **as soon as possible**, ideally before the affected assessment deadline. You must submit your Exceptional Circumstances claim form within **7 days of the assessment deadline**. If you do not submit by the deadline indicated without good reason, your claim will not be considered.

Full details of the Exceptional Circumstances Policy and the online application form are available on the [Exceptional Circumstances affecting Assessment webpage](https://www.york.ac.uk/students/studying/progress/exceptional-circumstances/).

**Please take proper precautions to safeguard your work and remember to make backup copies of your data. The University provides all its students with storage space on the University server and you should save and back up any work in progress on this server on a regular basis. Computer failure and theft of your equipment or storage media are not considered mitigating circumstances and extensions cannot be granted for work lost for these reasons.**

# Marking Criteria Grade Breakdown

## 1. Implementation (50%)

|  |  |  |
| --- | --- | --- |
| **Functionality approach/justification: Component 1 (10%)** | | |
| 0-39% | Fail | No approach or justification. Constructs are missing/inappropriate to cover all the functionality specified in the brief. |
| 40-49% | Comp Fail | Approach described but not justified. Constructs are appropriate but incomplete for covering all the functionality specified in the brief. |
| 50-59% | Pass | Approach is described with minimal justifications. Mechanisms are appropriate and cover all the functionality specified in the brief. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **Functionality approach/justification: Component 2 (10%)** | | |
| 0-39% | Fail | No approach or justification. Constructs are missing/inappropriate to cover all the functionality specified in the brief. |
| 40-49% | Comp Fail | Approach described but not justified. Constructs are appropriate but incomplete for covering all the functionality specified in the brief. |
| 50-59% | Pass | Approach is described with minimal justifications. Mechanisms are appropriate and cover all the functionality specified in the brief. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **GUI design approach/justification: Component 1 (10%)** | | |
| 0-39% | Fail | No approach or justification. Constructs are missing/inappropriate to cover all the functionality specified in the brief, i.e. no/incorrect shapes. |
| 40-49% | Comp Fail | Approach described but not justified. Constructs are appropriate but incomplete for covering all the functionality specified in the brief, i.e. only some shapes/colours. |
| 50-59% | Pass | Approach is described with minimal justifications. GUI components are appropriate and fully meet the functionality specified in the brief. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **GUI design approach/justification: Component 2 (10%)** | | |
| 0-39% | Fail | No approach or justification. Constructs are missing/inappropriate to cover all the functionality specified in the brief, i.e. no/incorrect shapes. |
| 40-49% | Comp Fail | Approach described but not justified. Constructs are appropriate but incomplete for covering all the functionality specified in the brief, i.e. only some shapes/colours. |
| 50-59% | Pass | Approach is described with minimal justifications. GUI components are appropriate and fully meet the functionality specified in the brief. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **General (10%)** | | |
| 0-39% | Fail | Unstructured code/no coding convention followed, no or few comments, inappropriate variable/method names. |
| 40-49% | Comp Fail | Structured but doesn’t clearly follow a coding convention. Mostly appropriate variable/method names. Sparse comments. |
| 50-59% | Pass | Structured and follows a coding convention. Appropriate variable/method names. Comments throughout. |
| 60-69% | Merit | As Pass. Well-documented/self-documenting. Comments show purpose (e.g. explanation, not just labelling). |
| 70-100% | Distinction | As Merit. Professional/production level documentation and structure (e.g. Javadoc). |

## 2. Functionality and Testing (50%)

|  |  |  |
| --- | --- | --- |
| **Overview of strategy for testing/demonstration of functionality (10%)** | | |
| 0-39% | Fail | No strategy or justification. Descriptive outline of some of the functionality or testing. |
| 40-49% | Comp Fail | A strategy is described but not justified. Descriptive outline of how the functionality or testing is demonstrated to meet the brief. |
| 50-59% | Pass | A strategy is described with minimal justifications. Overview of approach(es) to demonstrate how the brief is fully met. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **Evidence of functionality: Component 1 (10%)** | | |
| 0-39% | Fail | No approach or justification. Descriptive outline of how this construct works with no reference to the functionality specified in the brief. |
| 40-49% | Comp Fail | Approach described but not justified. Descriptive outline of how the functionality or testing is demonstrated to meet the brief. |
| 50-59% | Pass | Approach is described with minimal justifications and evidence to demonstrate how the brief is fully met. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **Evidence of functionality: Component 2 (10%)** | | |
| 0-39% | Fail | No approach or justification. Descriptive outline of how this construct works with no reference to the functionality specified in the brief. |
| 40-49% | Comp Fail | Approach described but not justified. Descriptive outline of how the functionality or testing is demonstrated to meet the brief. |
| 50-59% | Pass | Approach is described with minimal justifications and evidence to demonstrate how the brief is fully met. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **Evidence of error handling: Component 1 (10%)** | | |
| 0-39% | Fail | No approach or justification. Descriptive outline of how testing was carried out with no reference to the functionality specified in the brief. |
| 40-49% | Comp Fail | Approach described but not justified. Descriptive outline of how testing is demonstrated to meet the brief. |
| 50-59% | Pass | Approach is described with minimal justifications and evidence to demonstrate how the brief is fully met. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |

|  |  |  |
| --- | --- | --- |
| **Evidence of error handling: Component 2 (10%)** | | |
| 0-39% | Fail | No approach or justification. Descriptive outline of how testing was carried out with no reference to the functionality specified in the brief. |
| 40-49% | Comp Fail | Approach described but not justified. Descriptive outline of how testing is demonstrated to meet the brief. |
| 50-59% | Pass | Approach is described with minimal justifications and evidence to demonstrate how the brief is fully met. |
| 60-69% | Merit | As Pass. Justifications are clear and supported by references to the reading covered within this module to justify decisions. |
| 70-100% | Distinction | As Merit. Reference to a broad range of reputable literature (for example, conference papers, textbooks and white papers) to justify decisions. |