Department of Computer Science

Summative Coursework Set Front Page

|  |  |
| --- | --- |
| Module Title | Programming in Java |
| Module Code | CS2PJ20 |
| Lecturer responsible | Prof Richard Mitchell |
| Type of Assignment (e.g. technical report, portfolio exercise, in-class test) | Coursework |
| Individual or Group Assignment | Individual |
| Weighting of the Assignment | 40% |
| Word count/page limit | Front Page + 4 pages + form in appendix |
| Expected hrs spent for the assignment (set by lecturer) | Weekly Lab sessions + 5 hours |
| Items to be submitted | ZIP file with report, video and code. |
| Work to be submitted on-line via Blackboard Learn by | 5pm Fri 8/12/23 |
| Work will be marked and returned by | Within 15 working days |
| Note  By submitting this work you are certifying that you have read the assessment guidelines which are displayed at the top of the Assessment Folder on the Blackboard course for this module, and that you have conformed to the associated policies and practises, including those on   * Submitting your own work, not that of other people or systems, and the associated penalties for Academic Misconduct * Submitting by the specified deadline, and the penalties associated with late submission (if allowed) * The exceptional circumstances system (for applying for extensions) * The use of a green sticker for students with relevant needs | |

# 1. Assessment classifications

| First Class (>= 70%) | Strong technical knowledge and skill shown through development, proving a strong grasp of object orientation and advanced programming. Report is well structured and fluently written. Design choices are validated in the report, and the work shows some research and novelty beyond the standard material |
| --- | --- |
| Upper Second (60-69%) | A solid grasp of the subject with a good selection of advanced programming methods. The report is well written and validates design choices. May show some elements of creativity and originality and makes use of existing literature to validate choices |
| Lower Second (50-59%) | A reasonable range of grasp of the subject, with few technical errors and written in plain English. On topic, relevant, and relatively well organised. |
| Third (40-49%) | Evidence of appropriate study showing success in progress towards providing a solution with most technical content corrects. The work relies on simple examples or uses methods inconsistently. |
| Pass (35-39%) | Shows some evidence of study, but may be largely unfinished, flawed, or irrelevant, whilst showing some attempt to present a coherent solution. |
| Fail (0-34%) | Insufficient evidence of study/programming |

# 2. Assignment description

You are to develop a simulation of a variety of different robots and other objects moving around an arena and display this in a graphical user interface (GUI). You are free to add features to your Robot Simulation code.

In weeks 3, 4, and 5, you will develop the classes needed to create a simulation of robots moving around an area, displayed in a console window. The classes to be written are specified in detail in the lab sheets for these weeks. The work is to be demonstrated in week 5 in the Lab (either to staff or a fellow student) and then you self assess by answering the test on Blackboard.

In Weeks 7 and 8 you will learn about the JavaFX GUI library and object inheritance.

By week 9 you should know about Java, JavaFX, Objects, Inheritance and GUIs.

Hence the coursework is to develop a simulation of robots and other objects, displayed in a GUI, with which the user can interact. The GUI should have menus, buttons and an information panel, as set out below. There should be a proper object oriented design of the robots and other objects.

The GUI is expected to have the following.

**Application Menu**

The application should have a suitable menu which allows the arena to be configured, saved and edited, that allows the simulator to run, and to provide help and instructions. You can make design choices about the menu items as long as you describe these choices in the design section of the report.

**File Handling and Configuration**

The application should allow users to save and load simulation configurations from files. A configuration is defined as the set of parameters required to set up and start a simulation. By default, the simulator should load from a user-defined configuration file, but if one is not there, then a default arena with at least three robots or different items should be provided.

**Toolbar**

Also required is a toolbar with buttons to control the simulation (e.g., start, pause, add robots, etc.).

**Information Panel**

As the simulation runs, information should be provided on the state of the arena and its contents – these could be textual descriptions, but graphs or other displays may also be appropriate.

**Classes and Inheritance**

The classes for the console simulation were tightly specified, but you have more freedom for the GUI version, so you should employ suitable design strategies. See below for hints.

The code should make use of at least one abstract class, which is inherited directly or indirectly by other classes – the abstract class could be used by robots and other devices in the arena. Pay special attention to your use of inheritance and use access modifiers correctly. You should also have a class for the GUI, one for a canvas and one for the arena.

**Animation**

You can use any method, as long as you use JavaFX, to achieve the animation. In the console version, robots move either North, South, East or West. For the GUI version, movement should not be constrained to these four directions. It is suggested that your design is inspired by the Ball examples in the course, but your robots should look like robots and have sensors which are visible and used to determine how robots act.

**Comments, documentation and version control**

Your code should be professionally laid out and commented on using Javadoc-style comments.

## Additional information

The classes are to be developed in the timetabled lab sessions where support is provided. Students are expected to have read what is required before the sessions, plan on what they will write, then write their code in the session. They should ask for support when needed. They should complete any unfinished tasks in their own time before the next session.

# Hints on Simulation

A rich simulation is expected, comprising a variety of robots with different behaviour and other items in the arena.

It is perhaps best to have a robot which is like that in the console; namely, if it cannot move in its current direction because of a wall or another object, it turns. In effect, it has a ‘bump’ sensor.

There should also be robots with different types of sensors – perhaps detecting obstacles a certain distance away. You could have sensors which detect lights or even charging stations.

You could have obstacles in the way, and so a robot would have to steer to avoid an obstacle or other robot.

You could have robots which are predators of other robots.

You could have a robot which the user of the simulation controls.

You might find it easier if you make all robots/obstacles circular as it is easy to work out if they overlap. You can then use some of the concepts in the ball simulation. However, the robots should look like robots – eg have wheels.

You are free to give a game-like theme if you may wish.

## Key point

You must have at least one abstract class, possibly one which defines an object in the arena which all robots and other obstacles extend, or a robot abstract class which more sophisticated robots extend (as per the ball example).

It is important, therefore, that you do a full object-oriented design early on, before implementing the different robots. You must submit the class design in your report showing the hierarchy of classes you have implemented. See also the comment below as regards the final report.

# 3. Assignment submission requirements

*You will submit a ZIP file containing*

*A report of your project, using the provided template*

*Note in the appendix is a table in which you self assess your program*

*A video of you demonstrating your working program providing evidence of your judgement*

*A folder with the java source files (that you wrote)*

*A folder with the associated Javadoc web pages*

Front page of the student’s report

(*the following are compulsory)*

Module Code:

Assignment report Title:

Date (when the work completed):

Actual hrs spent for the assignment:

Please use the attached template for the report – note the page limit – being front page, 4 pages you write and the self evaluation appendix.

Video of your program

This should be a short, 5 min max, video of you demonstrating your program – show your program running. You should include the about and/or help which should show your student number. It is recommended that you show also the robots moving around, robots being added/deleted and if possible show the file handling. There should be sound of you explaining what is happening.

# 4. Marking scheme

In the appendix of the report template is a table for you fill in where you self assess your work, allocating yourselves marks in the specified categories, with a brief justification.

This table in effect defines the available marks for the code (worth 70%).

The marker will use the table and the video and allocate a suitable mark in these 5 categories.

The remaining 30% will be assessed from the report and other submissions according to the following table.

| **Area** | **Max Available Mark** |
| --- | --- |
| Working Video + Javadoc files + Code files | 2 + 2 + 2 |
| Abstract | 3 |
| Introduction and Images of program | 3 |
| Class Hierarchy | 3 for description 3 for suitability |
| Discussion of results | 3 |
| Reflection | 4 |
| Novel features | 5 |