Department of Computer Science

Assessed Coursework Set Front Page

Module Title: Java

Module Code: CS2JA16

Lecturer responsible: Prof Richard Mitchell

Type of Assignment: Programming – Demonstration / Report

Individual / Group Assignment: Individual

Weighting of the Assignment: 90% of 40%

Page limit/Word count: 10 pages

Expected hours spent for this assignment: 25 hours

Items to be submitted: Electronic Submission due 22 Jan 2020

(ZIP file with JAR and Report – details below)

Preliminary class Design Week 10 Autumn

(Submit to Blackboard with weekly test)

Work to be demonstrated: During Lab Sessions Week 2 Spring Term

Work will be marked and returned by: 12 Feb 2020

# NOTES

By submitting this work you are certifying that it is all your own work and that use of material from other sources has been properly and fully acknowledged in the text. You are also confirming that you have read and understood the University’s Statement of Academic Misconduct, available on the University web-pages.

If your work is submitted / demonstrated after the deadline, *10%* of the maximum possible mark will be deducted for *each* working day (or part of) it is late. A mark of zero will be awarded if your work is demonstrated more than 5 working days late. You are strongly recommended to submit / demonstrate in by the deadline as one late piece of work can impact on other work.

If you believe that you have a valid reason for failing to meet a deadline then you should complete an Extenuating Circumstances form and submit it to the Student Support Centre *before* the deadline, or as soon as is practicable afterwards, explaining why.

# 1. Assessment classification

There is a detailed marking scheme for the final demonstration, and marks for correct submission of runnable code and documentation, as well as for the associated report. These reflect the general criteria as shown below.

| **Classification** | Associated Criteria |
| --- | --- |
| **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 is showing study and novelty beyond the standard material. |
| **Upper Second (60..70)** | 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..60)** | 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..50)** | Evidence of appropriate study showing success in progress towards providing a solution with most technical content correct. The work relies on simple examples or uses methods inconsistently. |
| **Pass (30..40)** | Shows some evidence of study, but may be largely unfinished, flawed, or irrelevant, whilst showing some attempt to present a coherent solution. |

# 2. Assignment description

You are to develop a simulation of a variety of different drones and other objects moving round an arena, as displayed in a graphical user interface.

The work is inspired by the code written in weeks 3, 4 and 5 in which you had one type of drone moving round the arena, shown in a console interface. However you will need to do a proper object oriented design with a suitable hierarchy of classes, including at least one abstract class.

You must utilise the JavaFx GUI library which you will learnt about in weeks 7 and 8. The GUI should have menus, a toolbar with buttons and an information panel – as set out below.

## 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. 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 the user 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 configuration file, but if one is not there then a default arena with at least three drones or different items should be provided.

## Toolbar

Also required is a toolbar with buttons to control the simulation (e.g., start, pause, add drones, 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, and 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 drones 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, drones move either North, South, East or West. For the GUI version movement should not be constrained to these four directions – see the Ball examples.

## Comments, documentation and version control

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

You should use GitHub for storing, version control of your program.

## 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, planning on what they will write, to then write their code in the session, asking for support when needed, and to complete any unfinished tasks in their own time before the next session.

## Hints on Simulation

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

Drones could operate at different heights, and so when showing the arena you draw all those at low levels, before those at higher ones. This could be done by repeatedly scanning through all the drones, until all have been drawn, but only drawing a drone if there is none below it – and then remembering it has been drawn. You could try drawing it in 3D if you wanted!

It is perhaps best to have a drone 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 drones with different types of sensor – perhaps detecting obstacles a certain distance away (such as to the front left/right as per the robots you saw in Begin Robotics). You could have sensors which detect lights, or even charging stations.

You could have obstacles in the way, and so a drone would have to steer to avoid an obstacle, or rise up to go over it.

You could have drones which are predators of other drones.

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

You might find it easier if you make all drones/obstacles circular as it is easy to work out if they overlap. You can then use some of the concepts in the ball simulation.

## Key point

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

It is important therefore that you do a full object oriented design early on, before implementing the different drones.

As such, as part of the week 10 lab assessment, you must submit a brief document (2 page max) showing the hierarchy of classes you are planning which together with that week’s test, will comprise the mark for that week. See also comment below as regards the final report.

## Suggested Development Plan

In weeks 7 and 8 you will learn about using JavaFX and about inheritance, and in week 9 there is more detail on inheritance including the use of abstract classes. In the lab session in week 7, you learn about JavaFX and in week 8 model a simple solar system with planets and moons. At that stage you will know about developing a GUI, with menus, mouse clicks, buttons and animation.

In week 9 you should start the GUI version of the drone simulator. It is strongly suggested that you create a GUI, based on that in the solar system, but for drones. So have classes for GUI, canvas, drone arena and drone, and ensure that you have a drone which moves around the arena, perhaps turning when it reaches a wall of the arena, showing itself in the canvas of the GUI. Then have an array list of drones, and now the drones turn if they are about to encounter a wall or another drone. If you want to, have an abstract class for an item in the arena, which is inherited by your drone class. You could also have static obstacles which are in the class which if encountered by drones cause the drone to turn.

For the session in week 10, perform an object oriented design covering the various drones and other obstacles that will be in your arena, including at least one abstract class. Produce a simple document showing the class hierarchy ready for submission to Blackboard before you answer that week’s test. This could be similar to the figure on slide 8 of the “JavaWeek 9 AbstractClasses” powerpoint – showing the classes, key data and inheritance.

Implement some of the classes in week 10, and the rest in week 11, adding functionality such as file handling (for which it is recommended that you make your classes serializable so you can load or save an arena of drones in one go).

# 3. Assignment submission requirements

## Electronic submission:

There are two parts to the submission, which you should combine into one **ZIP** file – note RAR files are **not** acceptable, whose name includes your name and student number. The two parts which should be submitted to Blackboard are:

1. An electronic **report in docx / pdf format**. The report should be a maximum of ten pages, excluding the front page (see later) but including appendices and figures, on paper size A4 with 12pt font. It should include:
   1. A brief abstract. A short introduction.
   2. A description of the OOP design of your application (list/explanation of classes, UML diagrams, data design etc.) with a critical analysis of the design used. This could be an expansion of the document you submitted in week 10, with a commentary of any changes as part of your criticism. (Remember to use appropriate software engineering methods).
   3. A brief user manual incorporating screenshots of your GUI.
   4. Details of tests done
   5. A discussion, a short conclusion, and your personal reflection on this project.
2. An electronic copy of your code (**runnable** **jar archive including source code**.

Note, using Eclipse you can easily export your project as a jar archive.

Please make sure that it contains the source code (**mandatory**) and can be executed on a Windows machine (**mandatory**).

The JAR file should also include a top-level folder named “javadoc” containing the Javadoc documentation as a website.

You can find more information on jar archives here: <http://docs.oracle.com/javase/tutorial/deployment/jar/build.html>

The jar archive must contain the source code and must be able to be run using the command:

java –jar studentNameandNumber.jar

Include also a configuration file if appropriate.

Test that your JAR works on a Windows machine (eg one in G56) before you include it with the report into the ZIP file. This is because it will be tested on a Windows machine.

Front page of the submission

Module Code:

Assignment report Title:

Student Number (e.g. 25098635):

Date (when the work completed):

Actual hrs spent for the assignment:

Assignment evaluation (3 key points):

## Final Demonstration:

You are required to demonstrate the final application where you get a demo mark sheet signed by a marker during your practical session in the **Week 2** of the **Spring Term**. Missing the demonstration will impact your final mark negatively.

If you have an officially approved extenuating circumstance then contact Richard Mitchell by e-mail (r.j.mitchell@reading.ac.uk) to set up an alternative demonstration timeslot.

The last page of this document sets out how the demonstration will be assessed.

## Demonstration and Mark

Each student will be given a time period during which they are due to demonstrate the work. They should arrive before the start of that time, set up their program, fill in the marking sheet with their name and their assessment of the work they have done. They will then be asked to demonstrate their program to a marker who will then ask them to explain the code. The marker will propose a mark and will give the mark and feedback to the student. A receipt at the bottom of the marked sheet will be detached and given to the student. The marks given in the session are provisional.

Then the electronic submission will be assessed, the remaining marks allocated, and the demonstrator marks will be moderated to ensure consistency. Your ZIP file includes your name so it is easy to associate its contents with the marks for the demonstration. A final mark will be generated and put onto Blackboard.

That mark excludes penalties for lateness etc, and is also provisional until the overall moderation process is complete, as per normal.

# 4. Marking scheme

See below for the demonstration mark scheme. This should be printed by the student with the top and left column filled in by the student before the demonstration. The marker will fill in the right had column during the demonstration, and fill in the section at the bottom. This will be torn off and used as a receipt.

The sheet also shows that there are marks available for correctly submitting a runnable JAR file, with good Javadoc web pages, for the report, and 10% for innovation – where marks awarded for going beyond the ideas suggested here.

# CS2JA16 Coursework #1: FINAL DEMONSTRATION MARK FORM

Print this out, fill in your assessment in the first column **BEFORE** the demonstration and be set up ready to demonstrate your program and explain your code to the marker

|  |  |  |
| --- | --- | --- |
| **Your full name (PRINT):** |  | **MARK :** |

|  | Students own assessment | Marker’s observations during Demo | Mark |
| --- | --- | --- | --- |
| 1. | Overall **OOP design**, API and code style:  following Java code conventions:   * + variable and class names   + method names   + indentation rules * using inline developer comments, * using Javadoc comments, * abstract class * good hierarchy for items in building   Overall OOP approach **must be described in the report under the OOP design** including abstraction, encapsulation, and information hiding choices. Appropriate use of inheritance, access modifiers for classes, attributes and methods (Information Hiding). | Number of Classes:   |  | | --- | |  |  * Inheritance used   Access modifiers check:   * Correct * Less than 5 incorrect * More than 5 incorrect * Static variables and methods used correctly | 0-25 |
| 2. | JavaFX **GUI** design and implementation: usability, robustness, quality, user-friendliness.   * Crashes * Feedback to user instead of crashing or recover * Runs smoothly without interruptions * Lagging functionality   **This must be described in the report** | Design quality:   * Professional looking * Understandable * Toolbar buttons * Menu * File handling | 0-10 |
| 3. | **Animation**: and control (tick all that applies)   * Animation attempted but not fully working * Animation works with Start and Stop * Controlled using Menu control * Has Toolbar buttons | Comments | 0-10 |
| 4. | **Arena**: does it support different entities   * Drones which turn when meet walls/drones/objects * Objects which don’t move * Drones with additional sensors * Panel listing arena contents * Other – specify: * Other – specify: * Other – specify:   **Design must be described in the report** | Total number of different types in arena:   |  | | --- | |  |   Comments: | 0-20 |
| 5. | Quality of the submitted **report**.  Tests and discussion in report  Executable jar archive + embedded javadoc submitted. | **Evaluation based on submitted work.** | 0-20  0-5 |
| 6 | Bonus marks for innovation / outstanding features | **Academic Evaluation** | 10 |

**Signed and dated (student): Signed (marker):**

