UNIVERSITY *of York*

Department of Electronic Engineering

ELE00041I Java Programming Coursework Assessment 2020/21

SUMMARY DETAILS

This coursework for this module consists of the following two components:

Project Code contributes **65%** of the assessment for this module.

Individual Report contributes **35%** of the assessment for this module.

Clearly indicate your **Exam Number** on every separate piece of work submitted.

Submission is via the VLE module submission point. **The deadline is 12:00 noon on 13/5/2021, Summer Term, Week 4, Thursday.** Please try and submit early as any late submissions will be penalised. Assessment information including on late penalties is given in the Statement of Assessment.

ACADEMIC INTEGRITY

It is your responsibility to ensure that you understand and comply with the University's policy on academic integrity. If this is your first year of study at the University then you also need to complete the mandatory Academic Integrity Tutorial. Further information is available at http://www.york.ac.uk/integrity/.

In particular please note:

- Unless the coursework specifies a group submission, you should assume that all submissions are individual and should therefore be your own work.
- All assessment submissions are subject to the Department's policy on plagiarism and, wherever possible, will be checked by the Department using Turnitin software.

Java Programming

Assignment 2020/21

Dr Stuart J Porter

1 Task: Flocking Simulation

You are to design and implement a flocking simulation in Java using object-oriented programming techniques and Java libraries that have been introduced in lectures and labs. You **must** use directly the canvas and geometry classes developed in your "drawing" and "geometry" packages from the laboratories – failure to use these will result in an overall mark of **zero** for the "Project Code" component of the assignment; you may add to these classes as appropriate. You should also make use of the classes/code developed in the "turtle" package – you should modify these as appropriate for the new task.

Figure 1: Flocking birds: the flock seems to be an entity in and of itself but is really a collection of individuals, each following a set of simple rules.

Your application **must** simulate flocking behaviour in a two-dimensional world displayed graphically on the computer screen. Each individual in the flock should be able to move around and interact with others by following a set of simple rules that result in emergent flocking behaviour.

Your program **must** allow for the number of individuals and flocking parameters to be set and/or varied by the user through interacting directly with your application.

There is a rectangular obstacle of dimensions (dx=200, dy=100) pixels with its top-left corner at (250,150) on the canvas. This is impervious to any individuals – they cannot move into it or through it. You **must** implement this obstacle as part of the base specification.

To achieve high marks, the project should demonstrate good object-oriented design practices, such as inheritance and polymorphism, along with modular well-written code with good documentation and a well written and well-structured report.

To achieve high marks the program should be extended to include other complexities, such as control over simulation speed, obstacles, collision detection or other types of individuals for the flock to interact with (perhaps add predators into the simulation from which prey flee). However, a simple program that works is better than an unfinished, overly ambitious attempt that does not compile or execute correctly.

2 Academic Misconduct

This is an individual assignment – you **MUST** work on the program and report on your own. You are welcome to consult any sources of information you can find if you provide appropriate reference and attribution to the authors. If you use code written by anybody else (including your lecturers) you must make it clear which part is yours and which has been written by the other person. You should make this clear both in your report and in the code, itself.

Wholesale copying or reworking of large code chunks or a complete program written by someone else is not acceptable.

If the rules on what constitutes correct academic conduct are not clear, please consult the University guide on Academic Misconduct here:

https://www.york.ac.uk/students/studying/assessment-and-examination/academic-misconduct/

3 Submission

You are to write your program in standard Java, version 11 and it must compile and execute correctly on the departmental laboratory computers using the Java command line tools or Eclipse.

Submission will be via the VLE in two parts:

1. **Individual Report**: a PDF report (no more than eight A4 pages in length)
2. **Project Code**: A zip file containing:
   - The Java source code for your program (fully commented and properly indented).
   - A simple text file named README.txt that explains which source files should be compiled and the entry point used to run your final program.

PLEASE NOTE: Anonymised marking - use your exam number only

Marking will be done anonymously so **DO NOT** put your name on or in any of the submission parts. Instead, your report, code comments and filenames should identify you using your exam number only.

4 Marking Guidelines

The marks are broken down into the categories below (all marks in percentages). The **Project Code** will be marked under “Object-Oriented Design” and “Program Implementation”, the **Individual Report** will be marked under “Report”.

Appropriate Object-Oriented Design Practices	**(30)**
Use of multiple classes	6
Use of composition within your classes	4
Use of inheritance within your class structure	4
Use of polymorphism when referring to your objects	4
Use of interfaces within your class structure	4
Use of inner and/or anonymous classes	4
Use of public, private and protected	4
Program Implementation	**(35)**
Compiles, executes and runs without error	5
Execution (how well does the actual program work)	6
Completeness (how complete is the submitted program, how functional)	4
Modular, non-duplicated code (short, reusable, single-purpose methods)	4
Program documentation (commenting, formatting, suitable names of variables, etc)	4
Clarity and simplicity of program (good program structure, logical flow)	4
Extended functionality (going beyond the basic specification)	8
Report	**(35)**
Explanation of the flocking algorithm you used	5
Explanation of design	10
Explanation of program implementation	10
Summary of test procedures and results	5
Readability and formatting	5