UNIVERSITY of York

School of Physics, Engineering and Technology
Assessment 2024/25

Module Code	Module Name
ELE00055I	Software Design

Assessment Name	% contribution
Project Code	65
Individual Report	35

Accepted submission formats

Two files in the following accepted file formats:

Project Code: ZIP file containing code and associated files as indicated in the main assessment requirement.

Individual Report: Single Portable Document Format (PDF)

Submission is via VLE.

Please try and submit early as late submissions will be penalised.

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.

Software Design

Assignment 2024/25

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** directly use the Canvas and geometry classes developed in your "drawing" and "geometry" packages from the laboratories. You **must** use the Canvas class as-is from the module VLE – no modifications or subclasses are allowed. You may add to the "geometry" classes as appropriate, but any additions should **not** supersede the original functionality, particularly the method signatures and instance fields. Failure to use these Canvas and geometry classes as required will result in an overall mark of **zero** for the "Project Code" component of the assignment. You should also make use of the classes/code developed in the "turtle" package – you should adapt these as appropriate for the 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. Every 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. The implemented rules **must** control the individuals via the "`move(pixels)`" and "`turn(angle)`" methods in a similar way to the basic control implemented in the "`update`" method during the laboratories. The original "`move`" and "`turn`" method signatures **must** be as in the original laboratory template, keeping the parameters as integer values.

Your program **must** allow for the number of individuals and the flocking parameters to be varied by the user through interacting directly with your application's graphical interface.

There are three rectangular obstacles on the canvas: size (dx=120, dy=80) with top-left corner at (100, 300), size (dx=80, dy=150) with top-left corner at (350, 200), size (dx=150, dy=120) with top-left corner at (550, 100). These are impervious to any individuals – they cannot move into them or through them. You **must** implement these obstacles as part of the base specification.

To achieve high marks, the project should demonstrate good object-oriented design practices, such as appropriate 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 and/or diseased individuals which may or may not recover). 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.

Use of generative artificial intelligence is **not** allowed and you should consult the link below, which leads to the University policy on such.

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 laboratory computers used in the module, using the Java command line tools or Eclipse.

Submission will be via the VLE in two parts:

1. **Individual Report**: a PDF report of **maximum** eight A4 pages in length. Anything beyond the limit will **not** be considered. A title page is not required but, if included, will not count towards the limit. The report should cover all software design aspects including flocking algorithm, complex data structures / algorithms, testing, etc. – see the mark scheme below for more detail. **Not in 2-column format.**
2. **Project Code**: A zip file containing:
   - The Java source code for your program (appropriately commented and properly formatted).
   - A simple text file named README.txt that lists which source files should be compiled (simply stating “all” is acceptable if appropriate) and the entry point (main class) used to run your final program.

PLEASE NOTE: Anonymised marking

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 (in percentages). The **Project Code** will be marked under “Appropriate Object-Oriented Design Practices” 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 your design	10
Explanation of the program implementation	10
Summary of test procedures and results	5
Readability and formatting	5