School of Systems Engineering

Assessed Coursework Set Front Page

**Module code: SE2JA11**

**Lecturer responsible: Dr Karsten Oster Lundqvist**

**Coursework description: Major Coursework #1**

**Work to be submitted on-line via Blackboard by 10:30 am on: 25/01/16**

**Work will be marked and returned by: 12/02/16**

**NOTES:**

This coursework should be submitted on-line through Blackboard Learn.

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 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 submitted more than 5 working days late. You are strongly recommended to submit work by the deadline as a late submission on one 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 Information Centre *before* the deadline, or as soon as is practicable afterwards, explaining why.

**MARKING 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 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. |

# Major Coursework #1

In this coursework you are required to implement an application, the “**Artificial Life Simulator**”, to manage, run and visualise simulations of artificial life forms on 2D maps. The horizontal and vertical dimensions of the 2D map should be user-defined parameters. A life form should be able to move, sense the environment (sight, smell, etc.), find and eat food to increase its energy/stamina level. By using inheritance several different types of life forms should be developed. Life forms may prefer some type of food over others (e.g., carnivores vs herbivores). Some life forms may have a complex behaviour and organise themselves in social groups or communicate with others to hunt. When the life form performs an action (moving, sensing) its energy/stamina level is decreased. The environment can contain life forms, food entities containing some amount of calories or poison. Complex food entities may be able to provide a continuous source of food (e.g.. grass, plant, fruit trees, etc.). The environment contains obstacles (such as rocks, trees, etc.). Some objects in the environment may be changed by the life forms (such nests, dens, etc.) and used for protection. Life forms and other entities should be shown on the map by means of icons or images.

Each student may determine the richness and complexity of the simulation and use creativity, however remember that richness impacts positively on the final mark.

***Graphical User Interface:*** A Graphical User Interface (GUI) based on JavaFX must be provided. You can use the Java console (stdin/stdout) only for displaying debugging information. The GUI has menus with menu items and a toolbar with buttons to control the simulation (e.g., start, pause, reset, etc.). ***IMPORTANT***: GUI based on GUI builder tools will not be accepted: the GUI must be hand-coded by using the JavaFX API directly.

***Configurations File:*** The application allows 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. The application should also remember the last configuration file used when the application is restarted and should automatically load the last configuration when started.

***Application Menu:*** The application should be organized in 5 menus as shown in Figure 1. A different structure of the menu items is acceptable as long as they provide an equivalent set of features.   
You can make design choices about what the menu items are, as long as you describe these choices in the design section of the report.

***Inheritance:*** the code should make use of either an abstract class or an interface. ***Generics*** must be used throughout the code.

***Animation:*** In the lecture of week 8 java animations were discussed. You can use this method, or any other you prefer as long as you use JavaFX, to achieve the animation.

|  |
| --- |
| **File**   1. New configuration (reset configuration, no file is created) 2. Open configuration file (using your own file format, e.g. CSV) 3. Save (using your own file format) 4. Save as (allows to change the file name) 5. Exit |
| **View**   1. Display configuration 2. Edit configuration 3. Display info about life forms (e.g., list of <ID, name, position, energy>) 4. Display info about map (size, obstacles, food) |
| **Edit**   1. Modify current life form parameters 2. Remove current life form 3. Add a new life form |
| **Simulation**   1. Run 2. Stop 3. Pause/restart 4. Reset 5. Display map at each iteration: ON/OFF (toggle between ON and OFF) |
| **Help**   1. Display info about application 2. Display info about author |

*Figure 1: Application Menu*

**Electronic submission of the Jar archive**

Eclipse you can easily export your project as jar archive. Please make sure that it contains the source code (**mandatory**) and can be executed (**mandatory**). 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 studentName\_ALS.jar

***INSTRUCTIONS***

1. **Demonstration**: You are required to demonstrate the final application and to get a demo mark sheet signed by an assistant 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 Karsten Lundqvist by e-mail (k.o.lundqvist@reading.ac.uk) to set up an alternative demonstration timeslot.
2. **Hard-copy submission**:
   1. You are required to submit your demo mark sheet signed by either the lecturer or an assistant to the SIC by the specified deadline,
   2. You are required to submit a “Statement of Original Authorship” that covers both hard-copy and electronic submission to the SIC by the specified deadline.
3. **Electronic submission**:
   1. You are required to submit an electronic **report in docx or odt format** on Blackboard by the specified deadline. The report should include:
      1. a short informal presentation of the application (including screenshots of GUI and user manual) in max 3 A4 pages,
      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 you were provided throughout the weekly practical sessions. (Remember to use appropriate software engineering methods) max 4 A4 pages,
      3. tests, discussion and short conclusions with your personal reflection on this project in max 3 A4 page,
   2. You are required to submit an electronic copy of your code (**runnable** **jar archive including source code**) in Blackboard by deadline specified in the header. This file should include a top-level folder named “javadoc” containing the Javadoc documentation as a website.

**Coursework #1: DEMONSTRATION MARK FORM**

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

|  |  |  |  |
| --- | --- | --- | --- |
|  | Work category | Helper’s observations during Demo | Mark range |
| 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 Javadocs comments, * using Generics throughout the code. * Abstract Class and/or Interface   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).-1 stop, reset simulationsses used and | 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 * Responsive * Lagging functionality   **This must be described in the report under the informal presentation** | Design quality:   * Professional looking * Understandable * Lacking clarity   Missing elements:   * Toolbar buttons * Menu * Animation view | 0-10 |
| 3. | **Animation**: the GUI shows the animation of the simulation: the map is updated at each cycle. The animation can be controlled from the Simulation menu and the toolbar buttons | * Animation attempted * Animation works * Start/stop * Pause/restart * Menu control * Toolbar buttons | 0-10 |
| 4. | **Artificial World**: does it support different entities   * Renewable food (e.g. fruit trees, grass, honey hives) * Poisonous food * Modifiable environment (nests, dens, etc.) * Herbivores * Carnivores * Social life-forms (living in groups) * Forms supporting other senses than sight/smell * Other (describe in “Helper’s comment)   **Design must be described in the report** | Total number of different artificial world entities:   |  | | --- | |  |   Other: | 0-20 |
| 5. | Quality of the submitted **report**.  Tests and discussion in report  Executable jar archive submitted electronically with embedded javadoc. | **Evaluation based purely on quality of submitted work.** | 0-35 |

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