Assignment 2: First Iteration

Dead And Deader

This assignment follows on from Assignment 1. In that assignment, you designed some extensions to the functionality of a roguelike game that simulates a post-apocalyptic zombie world. In Assignment 2, you will implement those extensions.

You may change your design if you find that you need to. It is normal for the design to evolve alongside the codebase as the developers' understanding of the requirements improves. If your design changes, you should document the changes. This includes your Design Rationale as well as your diagrams.

In Assignment 3, you will be given further requirements to design and implement, as a well as a written task in which you will consider the overall system architecture.

Project requirements

In this assignment, you must implement all the functional requirements stated in the Assignment 1 specification.

Design is important

One of the primary aims of this unit is for you to learn the fundamentals of object-oriented design. In order to get a high mark in this assignment, it will not be sufficient for your code to "work". It must also adhere to the design principles covered in lectures, and in the required readings on Moodle.

To make this completely clear: **code that functions but is not well-designed according to the principles you have seen in lectures and readings, will not be enough to get you a good mark**.

Updating your design

As you implement your design, you may find that it has problems. You might also think of a better approach to some of the problems you've faced — your understanding of the problem will improve as you progress. **If you want to update your design, you may do so**; just make sure that you update your design documents so that they match the code. It is also a good idea write a brief explanation of your changes and the reasons behind them, to help your marker understand the thinking behind your code.

Coding and commenting standards

You must adhere to the Google Java coding standards that were posted on Moodle earlier in the semester.

Write javadoc comments for *at least* all public methods and attributes in your classes.

You will be marked on your adherence to the standards, javadoc, and general commenting guidelines that were posted to Moodle earlier in the semester.

Bonus marks

In Assignment 3, there will be bonus marks available for groups that come up with ideas for new features for the game, and then design and implement them. These could be new kinds of actor or entity, with new behaviours. They could be new kinds of location. They must be different from any features that have been requirements in FIT2099 in previous semesters.

A feature must be quite complex to qualify for a bonus mark. You must discuss your plans for bonus mark features with your tutor and gain approval before beginning work on them. Bonus marks will be awarded after the submission of Assignment 3, but can be started now.

A maximum of three bonus marks are available (these are whole marks for the unit). To make sure that bonus marks are awarded consistently, unit staff will decide how many bonus points each team's proposed feature is worth. No one has to attempt bonus marks, and it is possible to get full marks for the unit without any bonus marks.

Submission instructions

The due date for this assignment is *Friday, May 22nd, at 11:55pm (your local time)*. We will mark your Assignment 2 on the state of the "master" branch of your Monash GitLab repository at that time. If you've done any work in other branches, make sure you merge it into master before the due time.

The FIT2099 Assignment Rules apply to this assignment. Please read this document and make sure you comply, especially as regards the Work Breakdown Agreement.

You do not need to create a new copy of your work for Assignment 2. Git lets us easily roll back to any previous version of your repository, so keeping a separate "Assignment 1" folder only creates unnecessary clutter. If you like, you can add a tag to your final Assignment 1 commit before starting on Assignment 2, so you can find that version easily.[1]

You may update your design if you find that you need to, provided you update your documentation as well. You may also update your Work Breakdown Agreement if your initial version didn't cover Assignment 2, but make sure the other members of your team accept it. We will take your Work Breakdown Agreement into account when marking. If you choose to reallocate tasks, make sure you keep your WBA up to date and that both team members sign off on the updated version.

Unless a team member has applied for and received special consideration according to the Monash Special Consideration Policy,[2] late submissions will be penalized at 10% per day late.

It is both team members' responsibility to ensure that the correct versions of the documentation and code are present in the repository by the due date and time.

Marking Criteria

This assignment will be marked on:

- Functional completeness
- Adherence to design principles
- Code quality
  - readability
  - adherence to Java coding standards
  - quality of comments
  - maintainability (application of the principles discussed in lectures)
- Quality of updated documentation, if any
- Correct use of GitLab

Marks may also be **deducted** for:

- late submission
- inadequate individual contribution to the project
- academic integrity breaches

[1] `https://git-scm.com/book/en/v2/Git-Basics-Tagging`
[2] `http://www.monash.edu/exams/changes/special-consideration`

Learning outcomes for this assignment

This assignment is intended to develop and assess the following unit learning outcomes:

1. Iteratively construct object-oriented designs for small to medium-size software systems, and describe these designs using standard software engineering notations including UML class diagrams (in conceptual and concrete forms), UML interaction diagrams and, if applicable, UML state diagrams;

3. Implement object-oriented designs in an object-oriented programming language such as Java, using object-oriented programming constructs such as classes, inheritance, abstract classes, and generics as appropriate;

5. Use software engineering tools including UML drawing tools, integrated development environments, and revision control to create, edit, and manage artifacts created during the development process.

To demonstrate your ability, you will be expected to:

- implement the extensions to the system that you designed in Assignment 1
- use an integrated development environment to do so
- update your UML class diagrams and interaction diagrams as required, to ensure that they match your implementation
- justify any design changes in an updated Design Rationale
- use git to manage your team's files and documents

The marking scheme for this assignment reflects these expectations.