Assignment 3

Learning Outcomes

In this assignment, you will design and implement some new game functionality. This assignment is intended to develop and assess the following unit learning outcomes:

- **LO1.** Iteratively apply object-oriented design principles to design small to medium-size software systems, using standard software engineering notations, namely UML class diagrams and UML interaction diagrams.
- **LO2.** Describe the quality of object-oriented software designs, both in terms of meeting user requirements and the effective application of object-oriented design concepts and principles.
- **L03.** Apply object-oriented programming constructs, such as abstraction, information hiding, inheritance, and polymorphism, to implement object-oriented designs using a programming language (namely, Java).
- **LO4.** Apply effective programming strategies to refactor and debug object-oriented implementations systematically and efficiently using available programming language tools.
- **LO5**. Apply principles of software engineering practice to create object-oriented systems with peers using tools including integrated development environments (IDEs), UML drawing tools, and version control systems.

To demonstrate your ability, you will be expected to:

- design and implement further extensions to the system
- use an integrated development environment to do so
- update your UML class diagrams as required to ensure that they match your implementation
- use git to manage your team's files and documents

The marking scheme for this assignment reflects these expectations.

Introduction

Heavy penalty will apply for editing (add, modify, delete) any classes in the game engine as it breaks the purpose of the whole learning experience.

IMPORTANT: 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 this assignment, you will design and implement three (3) mandatory requirements **and either** the "survival mode" or the "creative mode" requirement (see below for more details). You **must** complete four new requirements to complete the assignment.

Assignment Modes

Since this unit is about design, we allow you to do some creative work! In this part, **you have two choices**:

- **Survival Mode:** we decide the features that you must work on (see [Survival Mode]).
- **Creative Mode:** you can go wild with your imagination (see [Creative Mode]). In other words, you can be creative in designing and implementing your own requirements. However, **you must strictly follow the rules and standards** that we set so that we can grade your work fairly.

If the rules are not followed, a heavy penalty will apply. So, please read the rules in the "Creative Mode" page carefully.

REQ1: The Ship of Theseus

Now that the intern has met their quota in collecting scraps, they are now ready to head back to the factory to sell their findings. The intern can park their spaceship on the factory's spaceship parking lot.

In addition to buying items, the intern can now interact with the computer terminal (`=`) to return to the factory.

A new moon (see REQ4) is now also available in the computer terminal. The intern may also decide to travel to the new moon to collect more scraps, which can be done by interacting with the computer terminal.

> **If the intern is near any computer terminal, they should be able to go to other places that they are not currently in**. For example, if they are currently in the factory, there should be only two options: travel to Polymorphia or the new moon (see REQ4).

After a software update, the computer terminal now also sells THESEUS (`^`), a portable teleporter that allows the intern to instantly move within a single map for free, for 100 credits each. THESEUS can be sold from the start of the game along with the other items from the previous assignment (see Assignment 2 REQ4).

The teleporter comes with the following instruction manual:

1. Put THESEUS down on the ground.
2. ~~Move away or anywhere within the map.~~
3. Select the option to teleport (i.e., "Teleport with THESEUS.")
4. THESEUS will now randomly teleport you.

Disclaimer: the factory cannot be held responsible in the event that you are teleported on top of a tree or a wall.

> THESEUS will **NOT** teleport the intern across maps. The intern will be randomly teleported to a location within the map that they are in.

You may use the following game map for the factory's spaceship parking lot:

```
".......",
".#####.",
".#___#.",
".#___#.",
".##_##.",
".......",
".......",
```

```
".......",
".......",
"......."
```

You may use the following game map for the new moon:

```
".........................~~~~",
".........................~~~~",
".........................~~~~",
"~........................~..",
"~~..........#####............",
"~~~.........#___#............",
"~~~.........#___#............",
"~~~.........##_##............",
"~~~..................~~.......",
"~~~~................~~~~......",
"~~~~...............~~~~......",
"..~................~~~~......",
"....................~~........",
".............~~..............",
"............~~~~............."
```

REQ2: The Static Factory

Once they arrive at the factory's spaceship parking lot, the intern is greeted by a humanoid figure (`H`). If the intern is within the surroundings of this entity, they can sell the scraps that they have collected so far.

The intern cannot hurt this humanoid figure since it is a part of the factory.

The following is the list of scraps that the intern can sell to the factory:

- Large bolts can be sold for 25 credits each.
- Metal sheets can be sold for 20 credits each. However, when the intern attempts to sell a metal sheet, there is a 60% chance that the factory will ask for a discount, only paying the intern 10 credits.
- Large fruits can be sold for 30 credits each.
- Jars of pickles can be sold for 25 credits each. If the intern attempts to sell this item, there is a 50% chance that the factory will pay double the price, paying the intern 50 credits instead.
- Metal pipes can be sold for 35 credits each.
- Pots of gold can be sold for 500 credits each. However, if the intern attempts to sell this item, there is a 25% chance that the factory will take the item directly from the intern without paying anything.
- Toilet paper rolls can be sold for 1 credit each. However, if the intern attempts to sell the toilet paper roll, there is a 50% chance that the intern will be killed instantly by the humanoid figure.

Other items not mentioned above cannot be sold to the factory.

REQ3: dQw4w9WgXcQ

Don't get what the title and this requirement are referring to? Click here to find out more :)

With the new software update, the intern can now also purchase an AI device (`z`), called Astley, for 50 credits each from the computer terminal.

The intern must pay a subscription fee to use the services provided by the factory's AI device, i.e. if the AI device is in the intern's inventory. Every 5 tick, the intern will have to pay the factory 1 credit. If the intern fails to pay the subscription fee, the intern will not be able to interact with the device until they have more credits to pay the fee.

If the device is not in the intern's inventory, the subscription will be paused. Once the intern picks the device up again, the subscription will continue.

When the intern has the AI device in their inventory, the intern should be given the option to listen to the monologue of the device. What the device says will be chosen randomly from the list of monologue options below. Note that some options can only be performed when certain conditions are met.

The AI device's monologue options:

- **Available from the start:** The factory will never gonna give you up, valuable intern!
- **Available from the start:** We promise we never gonna let you down with a range of staff benefits.
- **Available from the start:** We never gonna run around and desert you, dear intern!
- **Available if the intern has more than 10 items in their inventory:** We never gonna make you cry with unfair compensation.
- **Available if the intern carries more than 50 credits in their wallet:** Trust is essential in this business. We promise we never gonna say goodbye to a valuable intern like you.
- **Available if the intern's health point is below 2:** Don't worry, we never gonna tell a lie and hurt you, unlike those hostile creatures.

Consider what happens if there are other actors/item/ground that can monologue? Can your design accommodate these changes easily?

[Survival Mode] REQ4: Refactorio, Connascence's largest moon

If you decide to work on this requirement, you must **not** work on the creative mode requirement.

The title of this requirement should give you a hint at what you should do to your existing design and implementation.

Also, the title of this requirement is a reminder that you should write about connascence in your design rationale :)

Once the intern has sold all scraps that they found in Polymorphia, they headed back to their ship, preparing to go to the second moon, Refactorio, which is one of the largest moon of the planet named Connascence.

Similar to Polymorphia, this moon also has Inheritrees, but instead of starting from its sapling stage, the Inheritree starts from "sprout" (`,`). At this stage, the tree cannot produce any fruits, but it can grow into "sapling" (`t`) after 3 turns.

Next, "sapling" can grow to a young Inheritree (`y`) after 6 turns. Inheritree sapling can still drop the same fruit at the same rate each turn.

At this stage, the young inheritree lays dormant and cannot produce fruit.

After 5 turns, the young inheritree can grow into a mature inheritree, which can drop the same fruit as before at the same rate each turn.

To make the implementation simpler, you must modify your existing implementation of Inheritree to accommodate the new stages. In other words, from this point onwards, the Inheritree in Polymorphia can also grow through all the stages.

Although this requirement seems pretty simple, think about whether your proposed design violates any of the principles taught in the unit. Some things you may want to consider:
- Does any of the classes of this requirement attempt to manage several responsibilities at once? For example, one class attempting to manage multiple stages of the tree.
- How easy is it to extend your design? For example, what if there is another tree stage? Or another type of tree? Would you need to modify any of the existing code?
- Does any of the classes implement unnecessary method? For example, trees that cannot drop fruit are forced to implement a dropFruit method that simply returns null.
- Are there code duplications between the classes? For example, similar logic for searching for an exit to drop a fruit between different tree stages.

[Creative Mode] REQ4: The illusion of freedom

❌ If any of the following rules is not followed, a heavy penalty will apply or this requirement will be ignored during the marking process, qualifying as "missing one requirement".

Welcome to the Assignment **creative** mode! Here is an opportunity for you to use your creative juices and "freedom" to design your own game scenario to complete this requirement. Please refer to previous and current assignments (Assignments 1, 2, and 3) to write appropriate scenarios and testing instructions.

❌ If you decide to work on this requirement, you must **not** work on the REQ4 -- Survival Mode.

However, there are two rules that you must follow:

1. Your TA must be informed that your group has decided to complete REQ4 in a *creative mode*, during your applied session **in Week 12**.

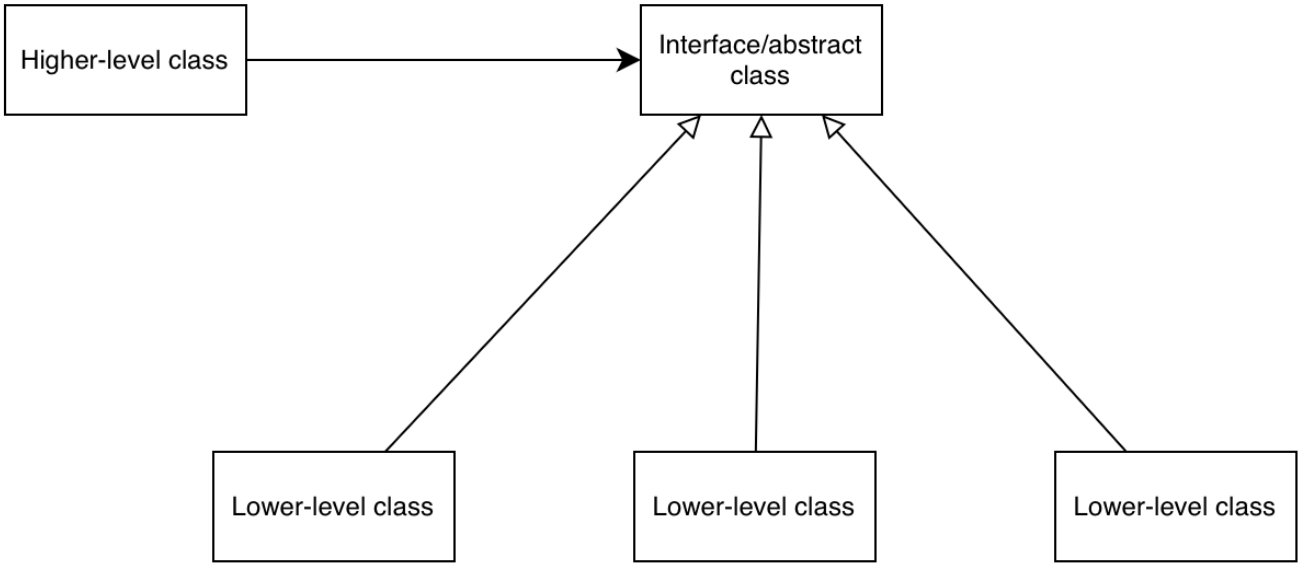
⚠️ If your TA was not aware that you are working on the creative mode requirement, a heavy penalty will be applied to the submission (i.e., REQ4 will be ignored during the marking process, **which is equivalent to missing one requirement**).

In addition, your TA must agree that your idea meets our standards. **You must draft your features in the form of scenario/story and write them down in Markdown format (`README.md`)**. **You must push this file to the repository before your applied session**. Then, you must pitch the scenario to your TA in week 12 of the applied session. If approved, your TA will record that your team decides to work on REQ4 as a creative requirement. Here, you can only add extra details (i.e., numbers, entities, testing instructions, etc.), and you are prohibited from reducing the complexity and amount of approved features.

❌ If the implemented feature is different than what was proposed, the implementation will be ignored in the marking process, which is again, equivalent to missing one requirement.

2. You **must** use the following design pattern

⚠️ Not using the following design will incur a heavy penalty in terms of design quality.



This means:

1. You **must** have **at least** 5 classes (1 higher-level class, 1 interface/abstract class, and 3 lower-level classes) in this requirement.
2. You must justify why this design is recommended in terms of the principles taught in the unit so far (e.g. DRY, reduce dependency, SOLID principles, etc.)
    - **Briefly** describe what you have done for this requirement
    - Explain the benefits of using this design pattern (make sure to relate the justification to the SOLID principles) in implementing this requirement
    - Explain the drawbacks of using this design pattern in implementing this requirement
    - Explain potential alternative designs, i.e. what other designs could have been used? How does it compare to the design above? Is the alternative design better than the design above?
    - Include concrete examples of how the requirement can potentially be extended in the future, e.g. what happens if a new “thing” is added (give examples)? How easily can your design accommodate this change?
3. Your idea must be new and original; in other words
    - Not simply extending any of the previous requirements, e.g. adding a new fruit, adding a new consumable item, adding a new enemy that can spawn from crater
    - Not simply adding something very similar to the previous requirements

- Not doing the same thing as any of the previous requirements

Submission Instructions

⚠ Within this class, you are welcome to use foundation models (ChatGPT, GPT, DALL-E, Stable Diffusion, Midjourney, GitHub Copilot, and anything after) in a totally unrestricted fashion, for any purpose, at no penalty. However, you should note that all large language models still have a tendency to make up incorrect facts and fake citations, code generation models have a tendency to produce inaccurate outputs, and image generation models can occasionally come up with highly offensive products. You will be responsible for any inaccurate, biased, offensive, or otherwise unethical content you submit regardless of whether it originally comes from you or a foundation model. If you use a foundation model, its contribution must be acknowledged in the submission (in the form of a separate file within your Design folder, titled `FoundationModelsUsed.txt/.pdf`); you will be penalised for using a foundation model without acknowledgement.

Having said all these disclaimers, the university's policy on plagiarism still applies to any uncited or improperly cited use of work by other human beings, or submission of work by other human beings as your own. Moreover, missing contribution logs/GIT commits and/or failing any handover interview will be treated as a potential breach of academic integrity which will be further investigated.

⚠ We will mark your assignment based on the latest commit in the `main` branch in your GitLab repository by the due date, **not** in the `master` branch.

✕ Heavy penalty will apply for editing (add, modify, delete) any classes in the game engine as it breaks the purpose of the whole learning.

You are expected to produce the following artefacts in Assignment 3:

1. Revised (based on feedback received on Assignment 2 or if you consider changes were needed) + New UML design documents, which include
    - 4 UML class diagrams (1 for each requirement, similar to Assignments 1 and 2)
    - 4 Interaction diagrams (1 for each requirement: sequence diagrams or communication diagrams are equally acceptable)
2. Design rationale, which should include (4 of them, 1 for each requirement)
    - Justification for any changes made to the design proposed for Assignment 2
    - Justification for the new requirements
3. Implementation (the code) of the system.
4. Contribution Log (mandatory, useful for markers) - see the Contribution Logs template in the Assignment Rules module

⚠ For class diagrams, please make sure to distinguish between existing classes and new classes with different colours. Additionally, you should colour revised classes and newly added classes differently.

Additionally, you will be asked to complete the following forms:

- Peer Assessment Form (**mandatory**, useful for markers and for you to reflect on your group

dynamics)

As mentioned above, you can (and should) update the artefacts previously marked in Assignment 2 since the code must align with the UML diagrams.

We will mark your Assignment 3 on the state of the `main` branch of your Monash GitLab repository by the due date at your local time (see above). If you have done any work in other branches, make sure you merge it into `main` before the due date. Additionally, please, make sure at least one team member submits the assignment to Moodle. This step is compulsory.

Do not create a new copy of your work for Assignment 3. Continue working on the same files, in the same directory structure (you might like to add a tag to your final Assignment 2 commit before starting on Assignment 3, so you can find that version easily). *https://git-scm.com/book/en/v2/Git-Basics-Tagging*

As we said above, you may update your design and implementation if you find that your Assignment 2 solution is not suitable for extension or if you think of a better approach during Assignment 3.

It is highly recommended that you explicitly Indicate in the **design rationale** what changed in the design between Assignment 2 and Assignment 3. Explaining the changes would help you organise your ideas to explain how the design changed during your handover video recording. A quick summary of changes with some explanation regarding why you applied such changes would be really valuable.

Contribution Log

We require you to have a document to know how you plan to divide the work between team members and simultaneously capture individual contributions. We have prepared a template you can copy to your university's Google Drive and share with your team.

Since your markers need to access this information, make sure that you create a share link and put it in the README of your repository. Failure to do so will result in 0 mark.

In this matter, everyone must contribute a **FAIR amount of code AND documentation**. It means you cannot work only on the code or only on documentation. In other words, your team **MUST NOT** split the work by artefacts: a person working on a "class diagram", another on a diagram alone, and another working on design rationale alone. We will give a heavy penalty to your team as a whole.

You will meet up with your team ideally once a week for about 1 hour for team discussion and another 1 or 2 hours for integration and resolving conflicts (if any).

Peer Assessment Form

You should also complete the peer assessment form, which you can find here: https://forms.gle/BETvSkCoPCx4briHA, to help us understand and evaluate your (and other team members') contributions to the assignment better.

About Late Submissions

Unless a team member has applied for and received special consideration according to the Monash Special Consideration Policy, late submissions will be penalised at 10% per day late (including weekends). Special considerations are granted individually.

It is all 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.

Handover Video Instructions

For Assignment 3 (A3), we will NOT be conducting a handover interview. Instead, we kindly request that you INDIVIDUALLY record a short video where you succinctly describe your work, placing particular emphasis on the good design practices you applied in the last assignment.

Please adhere to the instructions outlined below:

1. Upload your video presentation as an unlisted video on YouTube. Alternatively, the video could also be uploaded to your student account's Google Drive, Panopto or Zoom Cloud.

2. Your presentation **should not exceed 4 minutes in duration per student**. So, if you are a group of 4, the presentation should not exceed 16 minutes in length, while If you are a group of 3, it should not exceed 12 minutes.

3. The primary aim is for you to INDIVIDUALLY demonstrate **how you contributed to any of the requirements of A3**. Additionally, you should explain the correctness of your solution **based on the design principles** discussed in the unit.

Just a friendly reminder that when submitting the handover video, please do not only run the game but also explain your code and demonstrate a good understanding of your implementation. If the video only shows the game without any explanation or understanding, the entire group will be flagged as an academic integrity case that may require further investigation.

4. During the entirety of your video presentation/recording, ensure that your webcam is enabled and **your face is clearly visible** without the use of background images or filters.

If the webcam is turned off or the face is not clearly visible in the video recording, we will also flag this as an academic integrity case since we cannot verify if it's you who explained the solution. This may require further investigation.

5. You must verbally discuss your solution and the rationale behind it while **screen-sharing your code in IntelliJ.**

6. There's no need for professional video recording and editing tools; **a simple recording via Zoom with screen share will suffice**.

7. Kindly create a text/pdf file and upload it to GIT (to your Docs or Design folder) and Moodle titled "**HandoverVideos.txt / .pdf**". This file should contain the links to the videos of each member of your team in the following format:

```
Student Name 1 - Link to Zoom/Google Drive/Panopto/unlisted YouTube video

Student Name 2 - Link to Zoom/Google Drive/Panopto/unlisted YouTube video
```

```
Student Name 3 - Link to Zoom/Google Drive/Panopto/unlisted YouTube video

... and so on.
```

After you've received your final scores for this unit, you may want to remove the video from the platform where you uploaded it.

Please make sure that the teaching team can access the video on the platform where you uploaded it to. For instance, if you upload the video on Google Drive, please make sure to give the correct permission so that the teaching team can correctly access the video.

Please note that the teaching team is not expected to contact you if the videos are not accessible.

Marking Rubric

Assignment 3 Rubric

Prerequisite for marking the whole assignment: Design documents (UML diagrams and design rationale) and implementation need to be submitted for the assignment to go through the marking process. Missing any of these will result in 0 marks.

Overview:

Basic requirements are worth 100% of the marks (REQs1-4)

26 points

- Feature implementation completeness (6 points)
- Implementation quality: design principles (8 points)
- Design rationale (4 marks)
- Integration with the existing system (2 points)
- UML syntax and clarity (2 points)
- Alignment and design consistency (2 points)
- Style & Javadoc (1 point)
- Format and directory structure (1 point)

Detailed rubric:

Feature implementation completeness (6 points)

6 marks - The system runs and perfectly meets **all functional expectations** as per the relevant requirement(s) with no runtime errors. All required classes and relationships are implemented, and the program's behaviour aligns perfectly with the specification.

5 marks - The system runs and meets **all functional expectations (with some minor errors or punctual omissions)** as per the relevant requirement(s). All necessary classes and relationships are included, and the program's behaviour largely matches the specification, except for **one or two minor unexpected behaviours**. No runtime errors occur.

4 marks - The system runs and **partially meets all functional expectations, displaying several minor functional errors or omissions** as per the relevant requirement(s). Despite this, the majority of necessary classes and relationships are included, and **only a few minor unexpected behaviours occur.** No runtime errors are present.

3 marks - The system runs and **all functional expectations were addressed to some extent,** (with one or two major functional errors) as per the relevant requirement(s). Most important classes and relationships are included but **at least one or two major unexpected behaviours** are observed. A major unexpected behaviour is that which affects considerably the completeness of at least one requirement. No runtime errors occur.

2 marks - The system runs but **several functional expectations were not addressed** as per the relevant requirement(s) (e.g. the notion of Behaviour is not included even though it is a part of the requirement). Alternatively, runtime errors occur during the execution of the system but they can be easily fixed.

1 mark - The system runs but **addresses only some functional expectations and shows major omissions** as per the relevant requirement(s) (important classes or relationships are missing). Alternatively, runtime errors occur during the execution of the system without a clear idea of the cause or cannot be easily fixed.

0 marks - The system runs but it **poorly addresses or doesn't address the functional expectations** as per the relevant requirement(s). Alternatively, the system might not run at all.

Implementation quality: design principles (8 points)

This item applies across all functional expectations as per the relevant requirement(s)

8 marks - The implementation of all functional expectations as per the relevant requirement(s) **flawlessly adheres to good design principles and concepts** (e.g., DRY, SOLID principles, connascence and design smells), making the design easy to extend and maintain. Any punctual violations are **convincingly** justified in the design rationale (e.g., using singleton or factory patterns to implement a feature). All relevant requirements have been addressed.

7 marks - The implementation follows good design principles **nearly perfectly** making the design is easy to extend and maintain (if some punctual principles are violated, the trade-off is **somewhat** justified in the design rationale. All relevant requirements have been addressed.

6 marks - The implementation involves **one or two minor violations of design principles** (could be easily fixed) across all functional expectations as per the relevant requirement(s) (e.g., some attributes are not set to private without any justification). All relevant requirements have been addressed.

5 marks - The implementation involves **minor violations of design principles in multiple places** (could still be easily fixed) across all functional expectations as per the relevant requirement(s). All relevant requirements have been addressed.

4 marks - The implementation involves **one or two non-severe violations of design principles that could be implemented in a better way** (no trade-offs are convincingly provided in the design rationale) across all functional expectations as per the relevant requirement(s) (e.g. some code

repetitions are found in the implementation that would require some refactoring to be fixed). All relevant requirements have been addressed.

3 marks - The implementation involves **non-severe violations of design principl**es **in multiple places** (no trade-offs are convincingly provided in the design rationale). All relevant requirements have been addressed.

2 marks - The implementation involves **one or two severe violations** of design principles that could be implemented in a better way across all functional expectations as per the relevant requirement(s), e.g. violating the SOLID principles. Fixing them would require substantial refactoring. Alternatively, not all relevant requirements have been attempted.

1 mark - The design/implementation can be considered **hacky or various instances of procedural programming are found**. For example, the implementation uses downcasting and 'instanceof' in various cases without a convincing justification. Alternatively, it can also be the case that abstraction is not used, making the design **difficult to extend and maintain**. Alternatively, only some relevant requirements have been attempted.

0 marks - The implementation mainly follows a non-OO paradigm; or the UML class diagram(s) or the implementation is missing. Alternatively, most relevant requirements have not been attempted.

Design rationale (4 points)

4 marks - The design rationale includes a description of **what has been done and why**, focusing on the principles and concepts taught in the unit (such as DRY, SOLID principles, - **with a particular emphasis on connascence, and code smells**), and not in terms of game design. The rationale discusses the **advantages and disadvantages** of the design (pros and cons), **the reasons** for choosing the current design, and **ways in which it can be easily extended** (e.g. my design achieves OCP because if a new character is added in the future ...). All relevant requirements have been addressed.

3 marks - The rationale describes **what has been done and why**, based on the principles and concepts taught in the unit. It also includes **some discussion** of the pros and cons of the design and the reasons for the current design choice but **lacks examples of future extensibility.** All relevant requirements have been addressed.

2 marks - The rationale describes **what has been done and why**, aligning with the principles and concepts taught in the unit. However, it **lacks a discussion on the pros and cons of the design and/or examples of future extensibility**. All relevant requirements have been addressed.

1 mark - The rationale **primarily describes what has been done** without a thorough explanation of the decision-making reasons. For instance, it may mention design principles ad concepts without providing convincing explanations, discussion of pros and cons, or examples of how the system can be extended. Alternatively, not all relevant requirements have been attempted.

0 marks - The design rationale is either very challenging to read, is missing, or the submitted work omits more than one relevant requirement.

Integration with the existing system (2 points)

This item applies across all relevant requirements.

2 marks - The implementation **effectively uses the engine classes and classes created in Assignment 2** (e.g. the submitted work demonstrates that the students understand the difference between actions and behaviours). All relevant requirements have been addressed.

1 mark - The implementation **does not use some engine classes or classes created in Assignment 2 as intended** (e.g. behaviours are created for some actions that do not need it, such as actions performed by the player) or includes **custom classes for functionality that could be implemented with the engine class or previously** (e.g. created a custom ground class instead of using the ground class given in the engine package). Most relevant requirements have been addressed.

0 marks - The engine has been modified in a minor or significant way OR the UML class diagram OR the implementation is missing OR the classes created in Assignment 2 were not properly re-used/extended.

UML syntax and clarity (2 points)

This item applies across all relevant requirements.

2 marks - Relevant (static and/or dynamic) diagrams are **perfect in terms of syntax**, with no missing multiplicities, correct arrowheads for all relationships, and appropriate usage of realisation for classes implementing interfaces, generalisation for classes or interfaces inheriting others, etc. Diagram formatting is **consistent and clear**. All requested diagrams have been submitted.

1 mark - Diagram(s) contain **some minor syntax errors**, such as missing multiplicities for several associations, inappropriate use of generalisation for classes implementing interfaces, realisation for classes extending others, or classes extending multiple classes, etc. Nonetheless, the design can still be understood by the TA with a little effort. Alternatively, there are several **inconsistencies across diagrams**. All necessary diagrams have been submitted.

0 marks - Diagram(s) are significantly hard to understand or show major inconsistencies in formatting and clarity, OR more than one required diagram is missing, OR they do not resemble a UML diagram as required.

Alignment and design consistency (2 points) - incl. Design rationale, code and UML diagrams

2 marks - The design rationale and UML diagrams are in **perfect alignment** with each other and

with the code implementation across all functional expectations, as per the relevant requirement(s). All relevant requirements have been addressed or attempted.

1 mark - There are **some small inconsistencies** (that can be easily fixed) between the design documents and the implementation, but all relevant requirements have been addressed or attempted.

0 marks - There are **major inconsistencies** (which would necessitate significant changes for correction) or **numerous smaller discrepancies** distributed throughout the design documents and the implementation. Alternatively, more than one required diagram or implemented requirement is missing, or some submitted design documents do not resemble the required UML diagram format.

Style & Javadoc (1 point)

Applicable to basic requirements (1-4) only

1 mark - The code is properly documented with Javadoc across all functional expectations as per the relevant requirement(s), including class-level and method-level documentation. The Google Java Style guide is followed properly (e.g. package names are written in lowercase, attributes and variables names are written in lowerCamelCase, class names are written in UpperCamelCase, etc.). All relevant requirements have been addressed or attempted.

0.5 marks - Some classes and methods are missing Javadoc documentation OR some classes do not follow the Google Java style guide

0 marks - Most classes and methods are missing Javadoc documentation OR The Google Java style guide is not followed.

Project formatting and directory structure (1 point)

Applicable to basic requirements (1-4) only

1 mark - If the directory structure suggested in the specification has been followed. All the documents are easily found where expected.

0 marks - The suggested directory structure was not followed and some files may be hard to find.

Individual Contribution (100%) & Handover Video (compulsory)

We will, in principle, assume all team members equally contributed to the assignment. Yet, this individual contribution component will be used to weigh the marks awarded at a group level (see above) for cases where individual students did not contribute to the group task fairly (see details below) or did not show evidence that they understand their own work during the handover interview

(see details under Handover Interview, below).

If you contributed to the group task and demonstrated that you understand your own work you should get (100%) the full marks awarded at a group level as per the rubric above.

Individual Contribution (100%) Only for A2 and A3

100-80% - The student has made substantial contributions to the submitted work, evident by numerous **commits** (over 10) with meaningful commit comments (note: default comments from web UI are not considered). For full marks, there should be **multiple contribution log entries** (over 10 entries) with comprehensible explanations for an external reader (e.g. the TA).

Additionally, the student's contributions reflect a reasonable understanding of OOP principles, whether in programming or designing UML diagrams. The student's input spans **both design and code**.

Additionally, other team members have given positive feedback or there are no complaints regarding the student's contributions (as **per peer assessment**).

70-40% - The student has made some contributions to the submitted work, evident by **some commits** (between 5 and 9) with meaningful commit comments OR there may be **commit messages that don't match the commited content**.

Alternatively, there are **some contribution log entries** (between 5 and 9) with comprehensible explanations for an external reader (e.g. the TA).

Alternatively, the student's input focused **only on the design or the implementation**.

Alternatively, there are **some complaints** regarding the student's contributions (as per peer assessment).

30-20% - Peer assessment, contributions logs, and git logs show there are **insufficient contributions from an individual**; the student received **a negative review** from other team members which usually happened due to last-minute work, difficult communication outside of labs, or any other kind of difficulties (evidence are collective from peer assessment, emails, contribution logs, and git logs). The student committed to git, with **at least four commits** and meaningful comments OR there are various commits that are not meaningful and that do not explain the purpose. **Some contribution log entries** (at least 4 entries) with meaningful explanations that can be understandable by a new team member (e.g. the TA).

10-0%% - There is no or little evidence of contribution to the team from Git logs or contribution logs. If applicable, the student only writes comments (inline or Javadoc) without any appropriate programming contribution. Clear evidence of academic misconduct of an individual would be marked as zero. Note: submitting files as attachments to the release notes would be marked as zero.

INDIVIDUAL Handover Video (compulsory)

COMPLETION If the student can briefly describe what parts of the system they created and briefly describe design decisions in a video **no longer than 4 minutes**. The short video needs to demonstrate that the student understands the various parts of the assignment (as per the logged commits) and at least some awareness of those parts created by other team members.

NOT COMPLETED If the video is not presented (see specifications) or it is unclear whether the student understands their own work front he video.

IMPORTANT: Failing to have meaningful commits (i.e. showing that the task was progressively completed) and/or failing to provide the handover video would automatically flag this as a potential case of plagiarism, it will be further investigated using a similarity check software, and **zero marks would be awarded.**

IMPORTANT: Any inquiry (e.g. potential conflict within a team) should be submitted via the emails below (not your AdminTA, Lecturer nor CE). Emails sent in other ways will not be processed in time.

FIT2099.Clayton-x@monash.edu if you are based in Clayton
FIT2099.Malaysia-x@monash.edu if you are based in Malaysia

Assignment 3 Q&A Session

Direct Link: https://monash.au.panopto.com/Panopto/Pages/Viewer.aspx?id=2a20f7ff-6ee4-4fa7-8991-b1790183d9b7

Meeting chat:

GMT20240523-071526_RecordingnewChat.txt