

Assignment 1: Design

Learning Outcomes & Materials

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;
- ✓ **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:

- read and understand UML design documentation for an existing Java system
- propose a design for additional functionality for this system
- create UML class diagrams to document your design using a UML drawing tool such as UMLet or Visual Paradigm – you are free to choose which one
- write a design rationale evaluating your proposed design and outlining some alternatives
- use git to manage your team's files and documents

The marking scheme for this assignment will reflect these expectations

Learning Materials

Please download this .zip file to familiarise yourself with the game engine, documentation, and more. The base code will be automatically available in your group's repository (Gitlab). We intentionally make this .zip file below doesn't have a game package. We don't want you to write code straightaway.

Repeat this mantra: Design, write code, test, fix design, fix code, repeat ☐

 [fit2099-assignment.zip](#)



Note: You **must NOT follow** demo apps' design decisions; they only show how to use the engine, **NOT** how to design a proper system with object-oriented principles.

Introduction

For the rest of the semester, you will be working in teams on a relatively large software project. You will design and implement new functionality to add to an existing system that we will provide to you.



IMPORTANT: A document explaining the FIT2099 Assignment Rules is available as the [Assignment Survival Kits](#) lesson. Please read it and make sure you understand **BEFORE** you begin the project – you are expected to follow what it says and will almost certainly lose marks if you do not.

Getting Started

The initial codebase will be available in a repository that will be created for you on git.infotech.monash.edu once teams are formed. In the meantime, design documents for the system are available for you under the Assessments section in Moodle. Download these and familiarise yourself with the design.

For each requirement, you will see **Scenario** and **Implementation Expectations** subheadings. Implementation expectations will be used in Assignment 2 later, but the provided information might be useful for you to grasp the feature as a whole.

Aiming to simulate the real-world conditions you may experience in the industry, a final product from this group project is a work of team effort. In other words, we will treat it as one product, not detached individual works. Since you are allowed to choose your team members, ensure that you are having similar goals. We won't create groups with students from different Applied Session classes as this makes it less clear who's supposed to mark you.

Requirements vs Group size



You do not need to submit an interaction diagram (e.g. sequence diagram or communication diagram) in **assignment 1**. However, you will need to submit these documents for assignment 2 and assignment 3. For assignment 1, you may still create these documents if you find them useful for designing the system.

If you are in a group of 3, you must complete all requirements. If you are working in pairs (group of 2), you must complete all requirements, except **REQ4**.

NOTE: The team is allowed to only complete REQ1-4 (or REQ1-3 for Group of 2), but maximum score will be capped to 80 out of 100 marks in total (i.e., unable to get HD). So, REQ5 is an HD requirement where it is worth 20 out of 100 marks.

In other words:

- Group of 3: REQ1, REQ2, REQ3, and REQ4 + *REQ5 [HD]*

- Group of 2: REQ1, REQ2, and REQ3 + REQ5 [HD]
- Group of 1 [*special case**]: REQ1 and REQ2 + REQ5 [HD]

*Generally, you should not work alone on this assignment. Group of 1 will only be assigned based on the decision of the admin TA and the lecturer for very exceptional circumstances.

General background

You will be working on a text-based “**rogue-like**” game. Rogue-like games are named after the first such program: a fantasy game named Rogue. They were very popular in the days before home computers were capable of elaborate graphics and still have a small but dedicated following.



If you would like more information about rogue-like games, a good site is <http://www.roguebasin.com/>. The initial codebase will be available in the repository mentioned above. It includes a folder containing design documents for the system.

```
.. nnnn..... ~~~~~
..... ##### ..... ~~~~~
.. nnnn..... # .. ____ .. ____ #..... ~~~~~
..... ____ #..... ~~~~~
..... ____ #..... ~~~~~
..... #..... _#..... ~~~~~
..... #..... ###.....
.....
~~~~~ ..... ### ____ ###
~~~~~ ..... _@ ____ #..... nnnn.....
~~~~~ ..... # ____ U ____
~~~~~ ..... # ____ K_#..... nnnn.....
~~~~~ ..... ### ____ ###
~~~~~ ..... # ____ #.....
.....
.. ####_ ## ..... 888 ..... ##### .. ## ..
.. #..... 888 ..... #..... ____ ..
.. # ____ ..... 888 ..... 888 ..... ____ # ..
.. ####_ ### ..... 888 ..... _..... # ..
..... 888 ..... ### .. _### ..
.....
```

In this assignment, we are inspired by [Elden Ring](#). We may use several similar names (characters,

items, locations) and concepts. The purpose of using an actual game's concepts is to help you visualise the required features. We also believe that using actual game references can **bring fun** while working on the assignments.

All of the linked game contents, videos, and images belong to the respective owners and are subject to copyright. We mainly use the concepts for educational purposes and provide credit to the original creators accordingly. We may also add, alter, and reduce the original content and features to make them more suitable to the game engine, unit outcomes, and assignments' time frame.

In this game, you will play as **Tarnished** (@) to explore The Lands Between, where the event of Elden Ring takes place!



The following slides (REQ1 to REQ5) are the functionalities that you need to design in Assignment 1, and they will be implemented in Assignment 2. Please go to [Submission Instructions](#) if you are not sure what you should do.

REQ1: Environments & Enemies

✓ If you are a group of 3, this requirement is **mandatory**.

✓ If you are a group of 2, this requirement is **mandatory**.

i If REQ4 is not mandatory for your group, i.e. if you are a group of 2, you can choose one weapon and one class from REQ4 and use it for testing your implementation and submission.

Scenario

In ***Elden Ring***, several hostile creatures inhabit the "*Lands Between*". Some of these creatures occupy different types of environments and will attack anyone that gets too close to their territory. Please read thoroughly; some features might be detailed in the latter sentences.

In the starting area, the following environments and creatures can be encountered by the player.

A. Environments

1. Graveyard n

- It is represented as n (lowercase N) and it is occupied by the "*Heavy Skeletal Swordsman*" creatures. They can be spawned from the graveyard with a 27% chance at each turn.

2. Gust of Wind &

- It is represented as & (ampersand) and it can spawn "*Lone Wolf*" with a 33% chance at every game turn.

3. Puddle of Water ~

- It is represented as ~ (tilde) and it has a 2% chance of spawning a "*Giant Crab*".

B. Enemies

One type of enemy could attack the other type. They, however, cannot attack their type, e.g. *Lone Wolf* can attack a *Giant Crab*, but *Lone Wolf* cannot attack another *Lone Wolf* (except for *Giant Crab* & *Heavy Skeletal Swordsman* since their attack damage anything in their surrounding).

If an enemy is close (one block away) to the player, they will follow the player. Otherwise, they will wander around the map. However, suppose an enemy of one type is close to another type of enemy. In that case, they will attack without following them, e.g. if a *Giant Crab* is close to a *Lone Wolf*, the

Lone Wolf will attack the *Giant Crab* without following it.

At each turn, *Giant Crabs*, *Lone Wolves* and *Heavy Skeletal Swordsmen* have a 10% chance of being despawned (removed from the map) unless they are following the player.

1. Heavy Skeletal Swordsman q

- A hostile creature, represented by q (lowercase Q), that has 153 hit points and carries around a weapon called *Grossmesser*.
- This creature has the following unique ability: it can become a "Pile of Bones" (represented by x (uppercase X)) for three (3) turns if killed by other enemies or the player. If the *Pile of Bones* is not hit within the three turns, the *Heavy Skeletal Swordsman* will be revived with full health.



Here is a video of the *skeleton* turning into a *pile of bones* and reviving in the game:

https://www.youtube.com/watch?v=_5CIRFaTF4I

2. Lone Wolf h

- A hostile creature, represented by h (lowercase H), that has 102 hit points and *bites* other creatures, including the player, dealing 97 damage with 95% attack accuracy.



Here is a video that can help you visualise how wolves can spawn from a gust of wind in the game:

<https://www.youtube.com/watch?v=1X5cxmOGWt4>

3. Giant Crab c

- Another hostile creature, represented by c (uppercase C), that has 407 hit points and *slams* other creatures, including the player, dealing 208 damage with 90% attack accuracy.
- Instead of only attacking one particular enemy in their surroundings, the *Giant Crab* may also decide to *slam* all creatures within their surroundings (all 8 locations), dealing the same amount of damage with the same accuracy as their targeted attack.
- Note that since their *slam* area attack hits anything in their surroundings, they **may hit** other *Giant Crabs* within the attack area.
 - Moreover, if a *Giant Crab* performs the slam area attack, actor A may get hit while actor B may not (the probability is independent between each actor, i.e. one actor getting hit does not mean another actor will also get hit)



Here is a video of a giant crab spawning from a puddle of water and slamming the player:

<https://www.youtube.com/watch?v=M5WfXbEK9Ws>

Edit: slamming all creatures in their surroundings is the special skill of Giant Crab.

C. Weapons

1. Grossmesser (?)

- A curved sword, represented by ? (question mark), carried around by the Heavy Skeletal Swordsman that deals 115 damage with 85% attack accuracy.
- This sword allows the user to attack a single enemy within their surroundings or to perform a *spinning attack*, which attacks all creatures, including the player, within the user's surroundings. The damage dealt and the attack accuracy for the targeted and the spinning attack is the same.
- Note that since the spinning attack hits anything in the user's surroundings, it **may hit** other actors of the same type (*Heavy Skeletal Swordsman A* performing the spinning attack may accidentally hit *Heavy Skeletal Swordsman B*)
 - If the user performs the spinning attack, actor A may get hit while actor B may not (the probability is independent between each actor, i.e. one actor getting hit does not mean another actor will also get hit)

Edit: Grossmesser will be dropped by Heavy Skeletal Swordsman when they are defeated by the player (after the pile of bones is destroyed).

Implementation Expectations (Assignment 2)



Although the following are mainly applicable for assignment 2, you may find them helpful when designing the system for assignment 1.

- These grounds can be directly added to the string representation of the map (i.e., ArrayList of strings to create a map).
- You have the freedom to design your map with these ground components if you can demonstrate all requirements.
- The grounds can only spawn the enemies if no other actor is standing at that location.
- If the player carries multiple weapons, they can decide which weapon they want to use for attacking the enemy if they have access to that weapon in their weapon inventory.
- The player can also attack the hostile creature with their intrinsic weapon. If they choose to do this, they will punch the enemy, dealing 11 damage.
- No actor should be able to enter the Wall, represented by # (the pound sign)
- Any enemies should not be able to enter a Floor, represented by _ (the underscore)
- Area attack cannot be chosen by the player or performed by the enemy if there is no one to attack in the surrounding.
- To simplify the implementation, if the enemy has multiple weapons, they will choose the first weapon in their inventory. They will use their intrinsic weapon to attack if they do not have any weapon.
- If the weapon the enemy holds has a special skill, the enemy has a 50% chance of using the weapon skill.
- The player should display their current hit points out of their maximum hit points, e.g. (42/100)
- You may use the following string to create your map:

```
..nnnn.....~
.....#####.....~
..nnnn.....#..__#.....~
.....__#.....~
```

[illegible]

REQ2: Trader & Runes

✓ If you are a group of 3, this requirement is **mandatory**.

✓ If you are a group of 2, this requirement is **mandatory**.

i If REQ4 is not mandatory for your group, i.e. if you are a group of 2, you can choose one weapon from REQ4 for the trading functionality **in addition to** other weapons and items specified below.

Scenario

Merchants/Traders are actors that allow the player to purchase and sell certain items or weapons. The player can find them scattered across the *Lands Between*.

The player can purchase items or weapons from the merchants using **runes**, the currency used in the game of *Elden Ring*. In this game version, hostile creatures are the main source of runes. Defeating them will award the player a certain amount of runes, as specified below.

A. Enemies

i The ranges specified below are **inclusive**, i.e. for Heavy Skeletal Swordsman, 35 and 892 should be included when determining the amount of runes dropped. The value of the runes should be generated randomly between the range.

Heavy Skeletal Swordsman is a hostile creature that could be spawned from a graveyard. If killed, this creature could generate any amount of runes within the range of 35 and 892.

Lone Wolf is another inhabitant of the *Lands Between* that could generate any amount of runes between 55 and 1470.

Finally, *Giant Crab*, if defeated by the player, generates 318 - 4961 runes.

i See REQ1 for more details about each hostile creature mentioned above.

NOTE that the player will only get runes from the hostile creatures if they defeat them directly. In other words, if enemy A kills enemy B, the player **will not get the runes** from enemy B's death. Additionally, if the enemy despawns, the player will not get the runes from the despawned enemy.

B. Trader K

The trader that could be found on the first map is [Merchant Kale](#), represented by K (uppercase K).

Merchant Kale sits around the building in the middle of the map. In the current version of the game, Kale cannot move around, and the player cannot attack them.

C. Weapons

Merchant Kale allows the player to purchase the following weapons:



Again, if REQ4 is not mandatory for your group, you should choose one of the following weapons (Uchigatana, Great Knife and Club) to implement the trading functionality. You may decide to implement all of them for developing a complete game, but your marks will not be affected (positively or negatively).

- Uchigatana for 5000 runes
- Great Knife for 3500 runes
- Club for 600 runes

Additionally, if the player carries any of these weapons in their inventory, the player could decide to sell them to Kale for the following price.

- Uchigatana for 500 runes
- Great Knife for 350 runes
- Club for 100 runes



For more details about these weapons (Uchigatana, Great Knife and Club), see REQ4.



Even if REQ4 is optional for your group, you **MUST** implement the requirements specified below.

Grossmesser is a curved sword carried around by the Heavy Skeletal Swordsman. This weapon cannot be purchased from Merchant Kale, but you can sell it to Kale if you have one or more in your inventory. Grossmesser can be sold for 100 runes.

Implementation Expectations (Assignment 2)



Although the following are mainly applicable for assignment 2, you may find them helpful when designing the system for assignment 1.

- To simplify the implementation, an unlimited amount of weapons or items can be purchased
- To simplify the implementation, the player can carry multiple weapons of the same type
- The player should display the number of runes they are currently holding.
- If the attacker can retrieve runes & the target can drop runes, display the number of runes dropped when the target is killed.
- If the actor does not have enough runes, allow them to choose a purchase action, but fail when the action is executed.
- Selling an item/weapon to a merchant will not add those items/weapons to the merchant's inventory.
- Enemies cannot purchase items from the merchant since enemies cannot enter the Floor.

- The enemy will not be able to attack the merchant since enemies cannot enter the Floor.

REQ3: Grace & Game Reset

✓ If you are a group of 3, this requirement is **mandatory**.

✓ If you are a group of 2, this requirement is **mandatory**.

Scenario

A. Flask of Crimson Tears

In Elden Ring, the enemies' attack deals a significant amount of damage, and the player does not have a lot of hit points. To give the player a fighting chance, **the player will start the game with an item, the Flask of Crimson Tears**. This item can be consumed **twice** (maximum uses for now), and each time the player uses it, their health will be restored by 250 points. Additionally, the player **cannot drop** Flask of Crimson Tears.

B. Site of Lost Grace

The Site of Lost Grace is a unique ground in the game of Elden Ring, represented by **U** (uppercase U). It allows the player to rest on it. When this happens, the entire game will be reset (see the next section for more details on game reset).

C. Game Reset

i If you implement this requirement, the death of the player will no longer end the game.

Since the concept of death has been removed from the Lands Between during the event of Elden Ring, nobody can truly die.

To implement this functionality, **when the player dies (or if they rest at the site of lost grace), the game will be reset.**

When the game is reset, the following happens.

- All enemies that can spawn from the grounds will be removed from the map/despawn (including piles of bone - see *REQ1: Heavy Skeletal Swordsman's unique ability*)
 - Assume all enemies are resettable, but enemies added in the future version of the game may have different behaviour, i.e. they may not be removed from the map when the game is reset.
- The player's hit point will be reset to the maximum
- The Flask of Crimson Tears will be reset to the maximum number of uses


Note that dropped weapons can be left on the ground if the world is reset (for instance, if the player killed a Heavy Skeletal Swordsman before the game is reset and the enemy dropped the Grossmesser, the Grossmesser will stay on the ground after the game is reset)

Also, note that when the player dies, their weapons and items will not be dropped.


Edit: when the player dies, they should respawn in the last site of lost grace that they visited.

D. Runes

In relation to runes, the following happens when the player dies.

 The following will not happen if the player rest at the site of lost grace.

- If the player "dies", their runes will be left on the location just before they died (e.g. if they are starting from (1,1) and then died at (1,2), the runes should appear in (1,1) instead of (1,2))
- When the player stands on top of the dropped runes, the player should be given the option to recover the dropped runes, e.g. `Tarnished retrieves Runes (value: 0)`
- If the player died again before grabbing the runes, the runes would disappear. **If the player rests at the site of lost grace, the runes will not disappear.**
- The runes dropped by the player should be represented by `$` (the dollar sign).

 Here is a video that you may find helpful in understanding how the runes are dropped when the player dies and how they can be retrieved back: <https://www.youtube.com/watch?v=V3W0iX-nZ4g> (although the video was made before the release of Elden Ring, the idea still applies)

Implementation Expectations (Assignment 2)

 Although the following are mainly applicable for assignment 2, you may find them helpful when designing the system for assignment 1.

- You should display the number of uses the Flask of Crimson Tears has left on the console.
- The player can recover their runes (it will be added to the number of runes that they currently hold)
- You should display the value of the runes dropped by the player when they died.
- You should print the "YOU DIED" message when the player dies.
- The first site of grace should be called `The First Step`.

REQ4: Classes (Combat Archetypes)



If you are a group of 3, this requirement is **mandatory**.



If you are a group of 2, this requirement is **optional**. You may decide to implement this requirement to develop a complete game. However, the marker will not assess the design and implementation of this requirement, i.e. no extra marks will be given, and no marks will be deducted if your design and implementation do not meet the marking criteria. If you, as a group of 2, work on this requirement and want some feedback on your design, please let your TA know (your mark will not be affected).

Scenario

A. The Player

In the game of Elden Ring, the player, represented by @ (the at symbol), is called Tarnished.

B. Classes/Combat Archetypes

Before the game begins, the player should be able to choose their starting class (not to be confused with the object-oriented concept of classes).

These classes determine which weapon the player will start the game with and the player's starting hit point.



For the current version of the game, the hit point of the player cannot be upgraded. Upgrading the total amount of hit points of the player, however, may be integrated into the game in the future version.

In the current version of the game, the following classes are available as options to the player when the game starts.

1. Samurai

If the player chooses the samurai class, they will start the game with *Uchigatana* as their starting weapon. Their starting hit point will be 455.

2. Bandit

If the player chooses the bandit class, their starting hit point is 414, and they start with the *Great Knife* in their weapon inventory.

3. Wretch

If the player chooses the *Wretch* class, they will start with 414 hit points and a *Club* as their weapon.

C. Weapons

1. Uchigatana)

- A katana type, represented by) (the close parenthesis), that is a starting weapon of the *Samurai* class. It deals 115 damage with 80% attack accuracy.
- This weapon allows the user to perform "*Unsheathe*", a unique skill that deals 2x damage of the weapon with a 60% chance to hit the enemy.

2. Great Knife /

- A dagger type, represented by / (the forward slash), that deals 75 damage with a 70% hit rate. This is the starting weapon of the *Bandit* class.
- This weapon allows the user to perform "*Quickstep*", a unique skill that deals normal damage to the weapon to the enemy. ~~After dealing with the damage, the user will move away from the enemy, evading their attack.~~

Edit: For simpler implementation, you do not need to make sure the player will evade their attack in the next turn. You can simply choose one location within the player's surroundings and move the player there as long as there is no actor in the new location.

3. Club !

- A hammer type, represented by ! (the exclamation mark), that deals 103 damage with an 80% hit rate. The *Wretch* class starts with this weapon.
- This weapon does not have any special skill.

Implementation Expectations (Assignment 2)



Although the following are mainly applicable for assignment 2, you may find them helpful when designing the system for assignment 1.

- For the implementation of quickstep, you can choose any location within the surroundings of the user to move them into (as long as there is no actor in the new location).

REQ5: More Enemies (HD requirement)



If you are a group of 3, this requirement should be completed to reach a maximum 100 marks (see Introduction and Marking Rubric).



If you are a group of 2, this requirement should be completed to reach a maximum 100 marks (see Introduction and Marking Rubric).

Scenario

To keep our implementation faithful to the original game, let's increase the difficulty of the game even more by adding **more types of enemies!**

Let's split the starting map into the Left (West) and Right (East) half.

On the West side of the map

- *Graveyard* spawns *Heavy Skeletal Swordsman*
- *A Gust of Wind* spawns *Lone Wolf*
- *A Puddle of Water* spawns *Giant Crab*

On the East side of the map:

- *A Graveyard* spawns *Skeletal Bandit*
- *A Gust of Wind* spawns *Giant Dog*
- *A Puddle of Water* spawns *Giant Crayfish*

A. Enemies



For more details on Heavy Skeletal Swordsman, Lone Wolf and Giant Crab, see REQ1

Similar to the enemies found in REQ1,

- At each turn, *Skeletal Bandit*, *Giant Dog* and *Giant Crayfish* have a 10% chance of being despawned (removed from the map) unless they are following the player.
- One type of enemy could attack the other type.
- If an enemy is close (one block away) to the player, they will follow the player. Otherwise, they will wander around the map. However, if an enemy of one type is close to another type of enemy, they will attack without following them.
- If the game is reset, *Skeletal Bandit*, *Giant Dog* and *Giant Crayfish* should be removed from the map/despawn.

1. [Skeletal Bandit](#)

- A hostile creature, represented by **b** (lowercase B), that has 184 hit points and carries around a weapon called Scimitar.
- This creature has the following unique ability: it can become a pile of bones (represented by **x** (uppercase X)) for 3 turns if killed by other enemies or the player. If the pile of bones is not hit within the 3 turns, the skeletal bandit will be revived with full health.
- They have a 27% chance of being spawned from the graveyard found on the East side of the map.
- If defeated by the player, they drop 35 - 892 runes.
- They are the same type as *Heavy Skeletal Swordsman*. Therefore, a *Skeletal Bandit* cannot attack another *Skeletal Bandit*, a *Skeletal Bandit* cannot attack *Heavy Skeletal Swordsman*, and vice versa (except for when an area attack is performed, as it may accidentally hit other creatures of the same type)



Here is a video of the skeleton turning into a pile of bones and reviving in the game:
https://www.youtube.com/watch?v=_5CIRFaTF4I

2. Giant Dog

- Another inhabitant of the *Lands Between*, represented by **G** (uppercase G), that has 693 hit points and slams other creatures (single target attack), including the player, with their head, dealing 314 damage with 90% accuracy.
- In addition to the targeted attack, the *Giant Dog* may also decide to slam all creatures within their surroundings (all 8 locations/squares), dealing the same amount of damage with the same accuracy as their targeted attack.
- Note that since their *slam* area attack hits anything in their surroundings, they may hit other enemies of the same type within the attack area.
- If a *Giant Dog* performs the slam area attack, actor A may get hit while actor B may not (the probability is independent between each actor, i.e. one actor getting hit does not mean another actor will also get hit)
- They have a 4% chance of being spawned from the *Gust of Wind* found **on the East side of the map**.
- If defeated by the player, they drop 313 - 1808 runes
- They are the same type as *Lone Wolf*. Therefore, a *Giant Dog* cannot attack another *Giant Dog*, a *Giant Dog* cannot attack *Lone Wolf*, and vice versa (except for when an area attack is performed, as it may accidentally hit other creatures of the same type)



Here is a video of the Giant Dog creature in the Elden Ring game: <https://www.youtube.com/watch?v=0eBcBkXNSVc>

Edit: slamming all creatures in their surroundings is the special skill of Giant Dog.

3. Giant Crayfish

- Another hostile creature, represented by **R** (uppercase R), that has 4803 hit points and slams other creatures (single target attack), including the player, with their *giant pincer*, dealing 527

damage with 100% accuracy.

- In addition to the targeted attack, the *Giant Crayfish* may also decide to slam all creatures within their surroundings (all 8 locations), dealing the same amount of damage with the same accuracy as their targeted attack.
- Note that since their slam area attack hits anything in their surroundings, they may hit other enemies of the same type within the attack area.
- If a *Giant Crayfish* performs the slam area attack, actor A may get hit while actor B may not (the probability is independent between each actor, i.e. one actor getting hit does not mean another actor will also get hit)
- They have a 1% chance of being spawned from the puddle of water found **on the East side of the map**.
- If defeated by the player, they drop 500 - 2374 runes.
- They are the same type as *Giant Crab*. Therefore, a *Giant Crayfish* cannot attack another *Giant Crayfish*, a *Giant Crayfish* cannot attack *Giant Crab*, and vice versa (except for when an area attack is performed, as it may accidentally hit other creatures of the same type)



Here is a video of the Giant Crayfish: <https://www.youtube.com/watch?v=d5DBoHr56GE>

Edit: slamming all creatures in their surroundings is the special skill of Giant Crayfish.

B. Weapons

1. Scimitar

- A curved sword, represented by **s** (lowercase S), carried around by the Skeletal Bandit that deals 118 damage with 88% accuracy.
- This sword allows the user to attack a single enemy within their surroundings or to perform a spinning attack, which attacks all creatures, including the player, within the user's surroundings. The damage dealt and the attack accuracy for the targeted and the spinning attack are the same.
- Note that since the spinning attack hits anything in the user's surroundings, it may hit other actors of the same type.
- If the user performs the spinning attack, actor A may get hit while actor B may not (the probability is independent between each actor, i.e. one actor getting hit does not mean another actor will also get hit)
- This weapon can be purchased from Merchant Kale for 600 runes.
- If this weapon is available in the player's weapon inventory, this weapon can be sold for 100 runes.

Edit: Scimitar will be dropped by Skeletal Bandit when they are defeated by the player (after the pile of bones is destroyed).

Implementation Expectations (Assignment 2)



Although the following are mainly applicable for assignment 2, you may find them helpful when designing the

- The grounds can only spawn the enemies if no other actor is standing at that location.
- No actor should be able to enter the Wall, represented by # (the pound sign)
- Any enemies should not be able to enter a Floor, represented by _ (the underscore)
- Area attack cannot be chosen by the player or performed by the enemy if there is no one to attack in the surrounding.
- To make the implementation simpler, if the enemy has multiple weapons, they will choose the first weapon in their weapon inventory. If they do not have any weapon, they will use their intrinsic weapon to attack.
- If the weapon that the enemy holds has a special skill, the enemy has a 50% chance of using the weapon skill.

Submission Instructions



You do not need to submit an interaction diagram (e.g. sequence diagram or communication diagram) in **assignment 1**. However, you will need to submit these documents for assignment 2 and assignment 3. For assignment 1, you may still create these documents if you find them useful for designing the system.



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.

You are expected to produce the following design artefacts in Assignment 1:

- Class diagrams: distinguish between existing classes and new classes with different colours
- Design rationale: why this approach is good? what's the alternative?
- Contribution Log (mandatory, useful for markers)

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)
- Feedback Request Form (to be completed in order to upload your files to Moodle, see instructions [here](#)).

You will implement (code) your design later, but for Assignment 1, you are not required to write any code. Instead, you must produce preliminary *design documentation* to explain how you will add the specified new functionality to the system. The new functionality is specified in the previous sections/slides (REQ1 to REQ5).

Design Diagrams

We expect you to produce **several** *UML class diagrams* following **the FIT2099 Assignment Rules**. These Rules are available on [EdLesson](#).

Your class diagrams should not show the entire system. The sample diagram in the base code shows the whole system to help you understand how the `game` works with the `engine`. But, in this assignment, you only need to show the following:

- the new parts,
- the parts you expect to change, and
- enough information to let readers know where your new classes fit into the existing system.

As it is likely that the precise names and signatures of methods will be refactored during development, you do not have to put them in these class diagrams. Instead, the overall responsibilities of the classes need to be documented *somewhere*, as you will need this information to begin implementation. This can be done in a separate text document (`.md` markdown format) or

spreadsheet, which you should put inside the `docs` directory. We will not assess this document, but we believe it will be handy during the implementation phase.

Design Rationale

To help us understand how your system will work, you must also write a *design rationale* to explain your choices. You must demonstrate how your proposed system will work and **why you chose to do it that way**. Here is where you should write down concepts and theories you've learnt so far (e.g., DRY, coupling and cohesion, etc.). You may consider using **the pros and cons** of the design to justify your argument.

The design (which includes *all* the diagrams and text that you create) must clearly show the following:

- what classes will exist in your extended system
- what the roles and responsibilities of any new or significantly modified classes are
- how these classes relate to and interact with the existing system
- how the (existing and new) classes will interact to deliver the required functionality

You are not required to create new documentation for components of the existing system that you *do not* plan to change.



IMPORTANT! We will not accept any Word document because it cannot be opened/displayed in Gitlab.

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. Please follow the detailed instructions [here](#).



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/mTK8WM6LZ345y8aQ7>, to help us understand and evaluate your (and other team

members') contributions to the assignment better.

Summary

In summary, you must put your design and text documents in the `docs` folder of your Monash GitLab repository and contribution logs inside README (in a shared link to Google Spreadsheet). We recommend you combine your diagrams and design rationale text in **one PDF file**. So:

- A PDF document that has class diagrams along with the design rationale.
- A link to the contribution log document.
- A class responsibilities document (OPTIONAL)

The due date for this assignment is in the Introduction part. We will mark a snapshot of your repository as it is at the due time. During the interview, you will need to notify your marker if you finished late and tell them which version they should check.

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 (including weekends). **The special consideration only applies to the individual who applied, not the whole team.** Details about special considerations can be found in the Unit Information section of Moodle.

It is **ALL** team members' responsibility to ensure that the correct versions of the documentation are present in the repository by the due date and time. Once all teammates have agreed on a final Assignment 1 submission, please do not commit further to the `main` branch until the due date has passed unless your teammate agrees to accept the penalty. If you want to continue to make changes to the repository for some reason, please create another branch.

Please check our [survival kits](#) for more helpful tips.

Marking Criteria

This assignment will be marked on the following:

Design completeness. Does your design support the functionality we have specified?

Design quality. Does your design take into account the programming and design principles we have discussed in workshops? Look in workshop slides, and check the Object-Oriented Fundamentals documents for principles like

Practicality. Can your design be implemented as it is described in your submission?

Following the brief. Does your design comply with the constraints we have placed upon it — for instance, does your design leave the engine untouched, as required?

Documentation quality. Does your design documentation clearly communicate your proposed changes to the system? This can include:

- UML notation correctness, appropriateness, and consistency
- consistency between artefacts
- clarity of writing
- level of detail (this should be sufficient but not overwhelming)

Explanation. Can you adequately explain your design and the reasoning behind it, both in your rationale and in response to interview questions from your marker? Here are the marking criteria:

1. Clearly communicated the information – using suitable language, tone and pace.
2. Used questioning techniques to encourage views and opinions.
3. Used active listening to confirm understanding.
4. Provide possible suggestions and comments for improvement.

Marking Protocol & Rubric

For this assignment, you have been asked to prepare design documents. Here are some important instructions for completing the submission.

1. Try to **put together all your design documents (UML diagrams) into a single PDF document** instead of creating various documents. You should:

- **Create more than one class diagram** (our recommendation is to create one diagram per requirement), so the diagrams do not end up too complex.



Do not submit a Word document!

2. The team's contribution log can be in a separate document.

- You could create a Google Sheet document with the template we have given here: <https://edstem.org/au/courses/10331/lessons/30074/slides/211805>, and put the link in the README file in your repository.

3. Put both documents into a `docs` or `design-docs` folder in the GitLab project space created for your team. Update the design documents and the contribution log as often as you need and make sure to commit each change to git.

4. Also, **upload the same documents to Moodle** before the deadline.



Don't panic if the Moodle submission is done **slightly after the deadline**, as your marker will look at the version in your repository.

5. This means **each member of your team should upload the same copy both to GitLab and Moodle**, but you will still be able to keep modifying them for assignments 2 and 3 in GitLab in the future.

6. When using git,

- All team members **must** commit their work (not relying on one member to commit everything)
- Add commit comments that are effective in describing the changes (not just the GitLab defaults)

7. You should also complete the peer assessment form, which you can find here:

<https://forms.gle/mTK8WM6LZ345y8aQ7>, to help us understand and evaluate your (and other team members') contributions to the assignment better.

Rubric (100 Marks)

Each requirement is worth 20 marks (REQ1-5: 20 marks each, and scaled depending on your

group size). It can be broken down as follows:

UML syntax (2 marks)

2 marks - The presentation and syntax of the UML diagram are perfect (no missing multiplicities, correct arrowhead for all relationships, correct relationship, e.g. realisation for classes implementing interfaces, generalisation for classes inheriting other classes, generalisation for interfaces inheriting other interfaces, etc.). **The design is easy to understand in terms of syntax.**

1.5 marks - The presentation and syntax of the UML diagrams are *almost* perfect, but there are minor syntax errors (e.g. perhaps using the wrong arrowheads, neglecting to include multiplicities, messing up the layout, or the UML syntax is incorrect in some parts.). **Still, with a little work, the design can be understood by the TA in terms of UML syntax.**

1 mark - The syntax is inconsistent. Perhaps some associations are missing multiplicities, dependency vs association error. **The design is unclear.**

0.5 marks - Diagrams are barely comprehensible in terms of syntax, and critical syntax requirements might have been left out. There are major syntax errors in the diagram (no multiplicities for most associations, generalisation for classes implementing interfaces, realisation for classes extending other classes, classes extending multiple classes, etc.). **There are big gaps in UML knowledge, and further consultation is recommended.**

0 marks - the diagram for the requirement is missing or does not resemble a UML diagram as required.

Requirement coverage (3 marks)

3 marks - If everything's been modelled well enough and ***all* functionality will be able to be supported for the specific requirement.** It's okay if the design isn't good from the point of view of design principles. The design meets all expectations etc. (e.g. all important classes and relationships are included in the diagram).

2 marks - **Some classes or relationships are missing**, but the requirement can still be implemented to some extent (e.g. Behaviour is not included in the UML class diagram even though it is a part of the requirement)

1 mark - Most of the critical classes or relationships are missing. The design has been attempted, but the description is so vague or poorly depicted that **it is unclear how the required functionality will be supported.**

0 marks - The UML diagram is missing or does not resemble a UML diagram as required.

Design quality (7 marks)

7 marks - The design clearly shows the application of key design principles, so it seems that it will be **straightforward to implement and easy to extend** later in Assignment 3. The proposed design

effectively uses the engine classes, follows good design principles (if some principles are violated, the trade-off is explained in the rationale, e.g. using singleton to implement a feature), and the design is easy to extend.

6 marks - The proposed design effectively uses the engine classes, follows good design principles (**no explanations are provided for violated principles**), and the design still is easy to extend.

5 marks - The proposed design effectively uses the engine classes and follows some design principles, but the design is **somewhat easy to extend**.

4 marks - The proposed design effectively uses the engine classes, but **it is not easy to extend**.

3 marks - The proposed design **somewhat uses the engine classes** (e.g. even though the engine code has provided an easy way to implement certain functionality, a customised solution is used instead).

2 marks - There is **no application of principles in the design** (potential code duplications, no abstractions, etc.).

1 mark - If any **modifications to the engine** are made (e.g. any relationships coming out of the classes in the engine package are made, such as adding an outgoing association from a class in the engine package to the game package) OR if the design does not use any of the engine classes.

0 marks - the diagram for the requirement is missing or does not resemble a UML diagram as required.

Design rationale (7 marks)

7 marks - The rationale contains a brief description of **what** has been done and **why** it has been done that way (NOT in terms of game design, e.g. I designed it that way to make the game easier to play). The explanation relates to principles taught in the unit, e.g. DRY, SOLID, etc. The rationale also includes some discussion of the **pros and cons** of the design, why the current design is chosen, and how it can be easily extended (e.g. my design achieves OCP because if a new character is added in the future, ...).

6 marks - the rationale contains a brief description of what has been done and why it has been done that way (NOT in terms of game design, e.g. I designed it that way to make the game easier to play). The explanation relates to principles taught in the unit, e.g. DRY, SOLID, etc. The rationale also includes some discussion of the pros and cons of the design and why the current design was chosen, **but it does not have any examples of how it can be extended in the future**

5 marks - the rationale contains a brief description of what has been done and why it has been done that way (NOT in terms of game design, e.g. I designed it that way to make the game easier to play). The explanation relates to principles taught in the unit, e.g. DRY, SOLID, etc. **However, it does not have any discussion on the pros and cons of the current design & no examples of how it can be extended in the future**

4 marks - the rationale **describes what has been done** with some mentions of principles (e.g. my design follows SRP) **with an insufficient explanation of why some decisions were made.**

3 marks - The rationale describes what has been done without in-depth explanation (class A has an association with class B because A will have an attribute of type B). **Simple correct explanations related to OOP principles are present, but there is an unclear connection with or mention of OOP design principles** (the actual names).

2 marks - The rationale describes what has been done with justification in terms of game design (without OOP design explanations), or the rationale only mentions principles **without any explanation or evidence of understanding the OOP principles.**

1 mark - The rationale only describes what has been done without any justification (a description of the diagram, e.g. Class A has an association with class B).

0 marks - either the design rationale is missing, it is extremely hard to read, or it does not refer to the submitted work.

Alignment between design & rationale (1 mark)

1 mark - The design diagrams and rationale perfectly match

0 marks - Either the UML class diagram or the design rationale is missing, or there is a major mismatch between the design and rationale (e.g. classes mentioned in the rationale are not available in the diagram)

Individual Contribution & Handover Interview (100%)



This individual contribution is 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 under Individual Contribution, 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 demonstrate 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 (80%)

80% - The student did sufficient contributions to the team, i.e., multiple **commits** (+2) with helpful commit comments (note: default comments from the web UI don't count.), ideally, one commit per week previous to the submission deadline; and multiple **contribution log entries** (more than 3 entries) with meaningful explanations that can be understandable by a new team member (e.g. the TA). The student shows a reasonable quality of contributions, such as a fair understanding of the OOP knowledge in programming or designing UML diagrams; If applicable, the student contributes to BOTH design and code. There are positive reviews or zero complaints from other team members (via **peer assessment**);

50% - 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 one commit and comment OR a single commit found without any comment or various commits that are not meaningful and that do not explain the purpose. Some contribution log entries (at most 2 entries) with meaningful explanations that can be understandable by a new team member (e.g. the TA).

0% - There is no evidence of contribution to the team from Git logs or contribution logs. Submitting files as attachments to the release notes would be marked as zero. 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.

Handover Interview (20%)

20% If the student **can answer all questions during the handover interview** satisfactorily. The responses demonstrate that the student understands the various parts of the assignment.

15% If all the questions **but one** is responded to adequately and sensibly. The responses still demonstrate some knowledge about the student's own work.

10%, If the student is unable to answer most of the interview questions (more than 2)

0% If two or more questions are not responded to adequately and sensibly. The remaining question(s) is/are partly responded to, but it is unclear whether the student understands their own work.



IMPORTANT: If **one student** in the team is awarded this mark, this would automatically flag this assignment as a potential case of plagiarism for the whole team and it will be further investigated using a similarity check software and **zero marks will be immediately awarded for the team.**

If you completed the assignment by yourselves, there is nothing to worry about during these interviews. It will be an opportunity to receive some feedback to consider in assignment 2, which involves implementing your own design. You will receive additional feedback once the marks are awarded. Remember, this is a team task, so you should be aware of how the parts designed or coded by others work.

IMPORTANT

Failing to have meaningful commits (i.e. showing that the task was progressively completed) and/or **failing the handover interview** 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.

Contribution

We will assume all team members equally contributed to the assignment (i.e. 50-50% for a team of two or 33.33% each for a team of three). Once again, answering our Peer Review form will help us evaluate your (and other team members') contributions to the assignment qualitatively:

<https://forms.gle/mTK8WM6LZ345y8aQ7>.



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

Recommendations on task splitting strategy

Although we don't expect you to work on any code/implementation in Assignment 1, We highly recommend completing this assignment iteratively to make sure that your proposed design works as expected, i.e.

Design -> Implement the design (Code) -> Repeat

A clear work division is also highly recommended for working in a team. We recommend the following breakdown to achieve equal contributions. You are welcome to adjust this recommendation as you wish.

Ideally, each person will be in charge of completing & creating a core object instance in the system (i.e., actor, item, or ground). Since each person learns one core part deeper than the others, they should better understand it and be ready to be an expert on that part. Hence, in the group tasks, each person will lead one shared task and coordinate with other team members to complete it.

Since there are some dependencies between requirements, the team must decide on proper class names and their relationships in Assignment 1. At the initial implementation stage, these design documents (as communication tools) will be used by other team members to complete their tasks. Clear design enables your other team members to prepare empty classes to be completed by the relevant team member in the next iteration/meeting.

So, in all assignments, ensure that each of you has a chance to work on: class diagrams, interaction diagrams, design rationale, code, and comments.



Please note that since REQ5 is an HD requirement, it is team's responsibility to ensure that everyone can contribute to this task. If there's clear evidence that an individual doesn't contribute to this requirement, the mark won't be awarded to that student. For example, Team Member 3 only worked on REQ 4 , and didn't join any meeting, discussions, or work on REQ5 (evidence is collective from peer review, contribution logs, and interviews). Team member 3 will get 0 in REQ5 but will be marked normally in REQ4. This is subject to the markers' decision. Yet, this may not apply if there's sufficient balance of workload in the team.

Group of 3

- Requirement 1:
 - Enemies: Team member 1
 - Environments: Team member 2
- Requirement 2:
 - Trader: Team member 2 and Team member 3
 - Runes: Team member 1
- Requirement 3:
 - Grace & Game Reset: Team member 1, Team member 2 and Team member 3

- Requirement 4:
 - Classes: Team member 3
- Requirement 5:
 - More Enemies: Team member 1, Team member 2 and Team member 3

Group of 2

- Requirement 1:
 - Enemies: Team member 1
 - Environments: Team member 2
- Requirement 2:
 - Trader: Team member 2
 - Runes: Team member 2
- Requirement 3:
 - Grace & Game Reset: Team member 1 and Team member 2
- Requirement 5:
 - More Enemies: Team member 1 and Team member 2