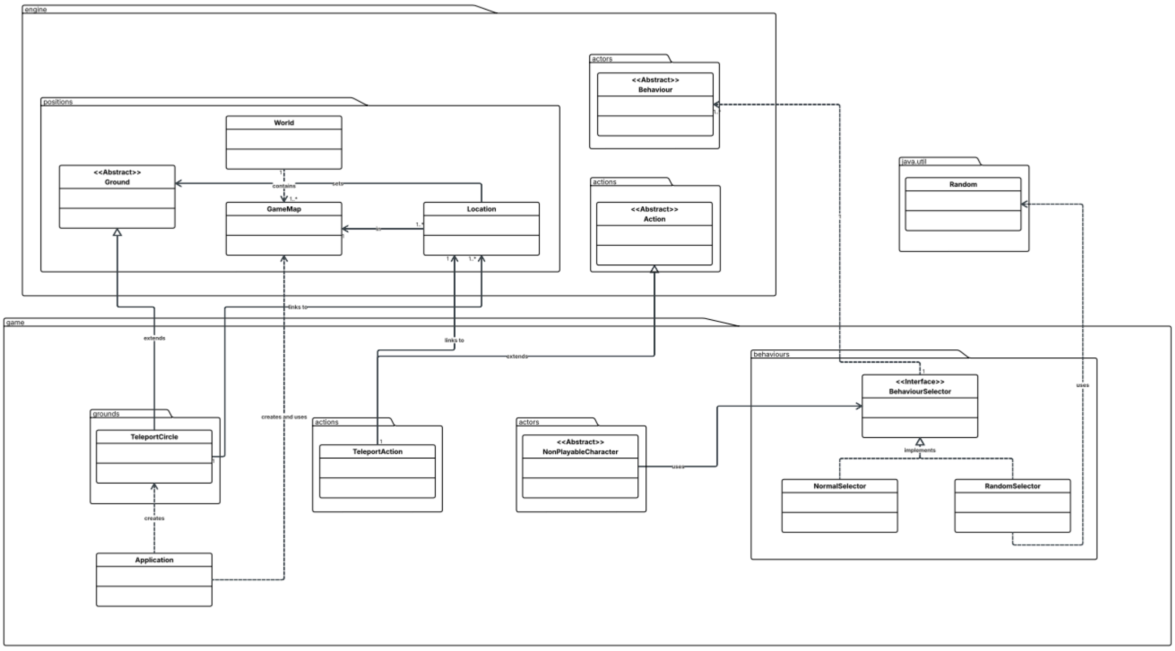
Assignment 3 Design Rationale

- Adrian Abdullah
- Nam Anh Tran

Requirement 1



A: Behaviour Selection

Design 1 (Chosen Option)

Interface for selecting behaviours and behaviour selection classes that implement it

Advantages	Disadvantages
Open Closed Principle: New selection strategies can be added without modifying existing NPC class logic.	Class bloat - each behaviour selection algorithm would be its own class, possibly leading to many classes if this concept is repeated with different systems
DRY: Behaviour selection logic is centralised in selector class, no need to duplicate behaviour checking code across NPC subclasses.	
Easily extensible - new behaviour selection method = 1 class which can be passed to	

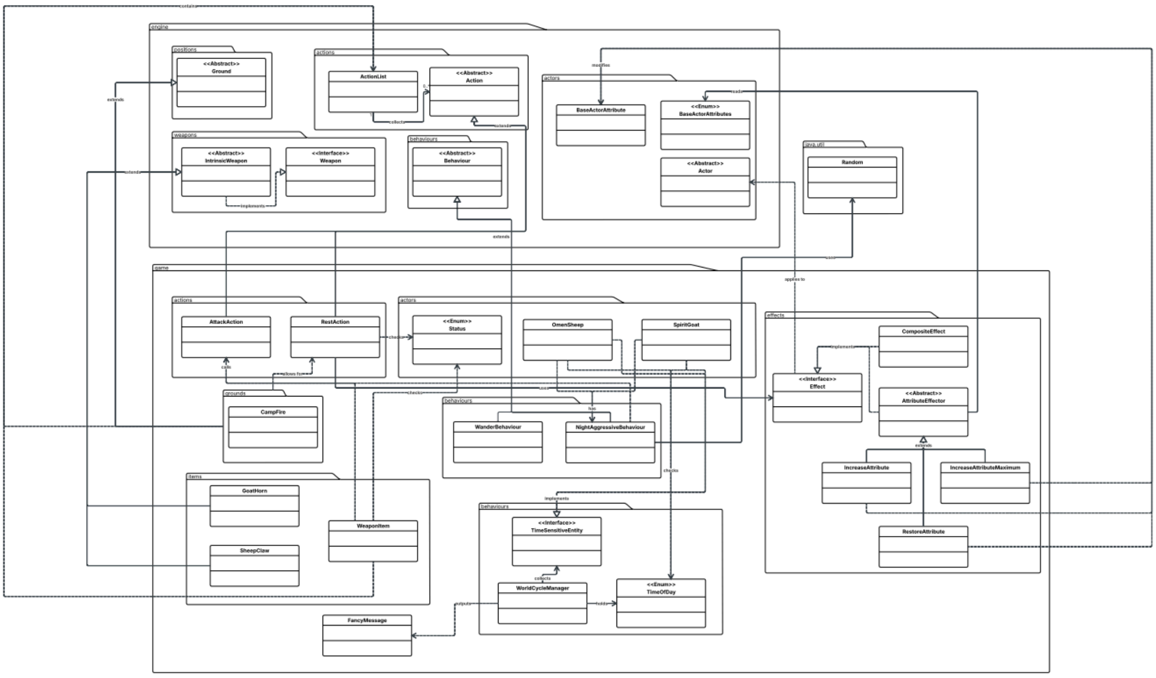
all npcs that want to use it	

Design 2
For each npc that can have multiple behaviours, create a new class for that npc with that behaviour instead

Advantages	Disadvantages
Straightforward extension - new behaviour, new class	A lot of repeating code
	Class bloat - one new behaviour selection method = many new npc classes for those that use that new behaviour selection method

In the end, design 1 was chosen as using classes for behaviour selection that implement a behaviourSelector interface allows for easy extensibility as each new behaviour selection class is only responsible for its specific behaviour selection method (SRP), which can be passed to any NPC that allows for a specific behaviour selection method.

Requirement 3



A: Day Time System

Design 1 (Only Considered Design)

Global time cycle that handles the effects on entities implementing the TimeSensitiveEntity interface.

Advantages	Disadvantages
Each entity implementing TimeSensitiveEntity interface can handle their own effect upon time changing	If there is a exact same effect between multiple entities when changing time, you would have to repeat yourself to implement their time changing code
Reusability: classes like NightAggressive Behaviour are modular and reusable across multiple actor types	
LSP: Time sensitive entities are abstracted and can be substituted with the interface	

B: RestAction and application of buffs

Design 1 (Chosen Design)

Application of buffs handled by classes implementing effects interface

Advantages	Disadvantages
SRP: RestAction only knows provides the system what buffs it wants to apply, while	Class bloat - instead of accessing directly and avoiding the creation of a class, each

effects handle actually applying the buffs to affected entities	effect needs a class created for it
Ease of readability is improved for anyone inspecting the code	
Modularity: new effect? can be added to a list of other effects to all be activated at the same time using CompositeEffect	
Reduces dependency of RestAction to actor	

Design 2
Manually buffing through accessing actor’s methods in RestAction

Advantages	Disadvantages
Ease of implementation - no need to create new classes for each buff	More cumbersome readability - instead of reading through composite effect you have to scan the modify attribute to see what it does
	More dependency between RestAction and Actor than alternative
	Less following SRP as the RestAction has to handle applying buffs as well

In the end, design 1 was chosen as it leans more into SRP with how the attribute modification classes implementing effects handle the buffing while RestAction only handles providing the effects-implementing classes what buffs it would like to give the actor. Additionally, with CompositeEffect, a list of attribute changes could be executed several times under one function call as opposed to copying and pasting the list of actor modifying attribute methods which follows the DRY principle.