FIT2099 Assignment 3 Recommendation

Introduction

After going through the engine code, we didn't find many codes that seemed hard to understand but there are still some parts that can be improved. We would also like to write about some principles that the engine code adheres to that greatly impacted the understandability of the codes.

What can be improved

- There are a limited number of interfaces in the engine code and hence causing methods to clump in a single interface. There are only 4 interfaces such as ActionInterface, ActorInterface, GroundInterface and ItemInterface. For example, if we want to implement a new method for WeaponItem we have to do it through ItemInterface where WeaponItem is a subclass of Item class. By doing this, Items that are not weapons are forced to implement the methods thus causing interface segregation. We suggest implementing additional interfaces such as separating ItemInterface into ItemInterface and WeaponItemInterface. Once additional interfaces are added, classes that do not need a method will not have to implement them which reduce interface segregation. The disadvantage of creating more interfaces is the interfaces might not be used at all in future projects.

- Not enough getter and setter methods in the engine code. Getter and setter methods are not provided which cause us to create our own getter methods through the interface methods. By doing so, it will cause the methods that implement the interface methods to be messy if they did not use the methods implemented in the interface. Once setter and getter methods are added in the engine code, we will not implement other methods in the interface which will improve encapsulation of the code. The disadvantage of this may cause privacy leaks if it is not implemented correctly.

Principles that the engine code adhere

- Don’t Repeat Yourself(DRY)

  There are classes written in as an abstract class which help to reduce the duplicate code. The abstract class of the engine code include Action, Actor, Ground and Item. There are also classes which extend these abstract classes and implement all abstract methods inherited from abstract classes.

- Good usage of polymorphism

  There are different Action classes which override the execute method so actors can perform different actions depending on the Action class. For example there are DoNothingAction, MoveActorAction, DropItemAction in the engine file that will execute in different ways by making the abstract methods into concrete methods.

- Fail Fast

  Fail Fast is used in the engine package. For instance , a try and catch statement is used in the constructor of the FancyGroundFactory class. This try and catch statement will terminate the program immediately when there is any error during the instantiation of a new ground object.

- Single Responsibility Principle

  Each of the classes inside the engine package do not have more than one responsibility. Behaviour class is implemented for handling actor behaviour and Actions class are implemented to handle actions taken by actors. This is important because it ensures the user of the engine code to know which class to use when a certain object is needed as the code is easy to understand.