Assignment 3

Recommendations

The engine code provided to use was easy to understand and we only came across a few minor problems when implementing the code. Therefore we would like to write about the design principles that we have learned from the engine code

The problems occurred that we were not able to check capabilities of a weapon object, it was troublesome as there were different categories of weapon (range & melee). The other small problem was that we were able to implement functions from the interface classes even though the interfaces are in another package.

The first design principle is **Don't repeat yourself.** There was very little duplicated code in the engine package. Some of the classes that were abstract classes such as Action class, Actions class, Item Class, Ground class etc. Most of the extra classes that were created were extended from these abstract classes. For example, the MoveActorAction and DropItemAction both extend from the Action class where they override the abstract methods and turn them into concrete methods. This is useful as it makes the code easier to maintain or change without introducing a bug.

The second principle will be the **open close design principle**. We can see the classes in the engine class are open for addition and close for modification. All the current methods in the classes should not be changed and only new methods are added for each feature introduced into the game. This would prevent the existing code from breaking as they are already tried and tested.

Furthermore, we also learned and applied the **single responsibility principle**. Each class has only one responsibility. For example, the MoveActorAction is only responsible for moving the actor to a designated location while the other functionality of the game is handled by other classes. Each Action that extends from Action class only carries out a specific function. This is very important , if we put more than one functionality in one class then there will be interdependence(coupling) between the two functionality, changing one functionality would most likely lead to the breaking of the other one. This principle would also make it easier for finding a certain functionality , if there is a problem for example with the drop item feature we will know that the error came from the DropItemAction class and not anywhere else.

We also implemented the **fail fast principle**. Try and catch statements are used for constructors in the engine code .For example, the FancyGroundFactory has a try and catch statement in its constructor . This will show the source where the error occurred during the instantiation of a new FancyGroundFactory.

Lastly we also ensure that the **variables are in the tightest possible scope**. Variables are all set as private attributes of a class or protected if necessary. This prevents methods from other classes from accessing and modifying these variables and thus makes the engine code less error prone .

As a conclusion, by following the design principle we have learnt , the code in the engine package is easier to understand, maintain and has no errors.