Assignment 3 Report

Group name: Brainsorts

- Youssef Nehad - 29801362
- Ashwin Sarith - 29940478

Possible Recommendations for change in the engine package

Negative Feedback

Problem: Throughout the engine package, some classes are using protected attributes rather than private attributes. This is not a good idea, as other classes can have direct access to these attributes and modify them without using setters and getters, and this interrupts the encapsulation theory and causes privacy leaks. For instance, in the World class especially inside the addGameMap method, the actorLocations is set by directly accessing the attribute. The fact that many attributes in engine package do not have accessors and mutators makes it impossible for code in the game package to access important attributes when needed.

Solution: all variables in the engine package must set to private and must have public setters and getters to make it possible for code in the engine or game package to access these variables and modify them without interrupting encapsulation and privacy leaks.

Advantages: allows code in the game package to access these instance variables and make the code much easier and understandable. This also allows us to write preconditions in setters and getters before using the variables. Furthermore, this reduces the scope of the variables and prevent unauthorized changes to them.

Disadvantages: sometimes it is not necessary that all attributes in the engine package must have accessors and mutators. Some might just be used within the scope of the package itself or even not used at all. For instance, the item variable in PickUpItemAction class is not used at all. So these type of variables are just needed to be private and avoid writing excess code that has no meaning and might confuse the developer.

Problem: One big problem that we faced was the many dependencies created as a result of the repetition in our code due to not having the ability to access everything in the engine package. However, the excess of dependencies is not good as it may easily cause the system to crash if one part of the code is modified. Also, the repeated code is not good, we are aiming to avoid repeating code. An example of dependencies would be the code having far too few getters and setters in the code. For example, the code to get the directions of the players inputs, was rather difficult considering how the directions were part of the init map, and to get those directions would require the creation of several getters which would violate DRY since we will be repetitively coding getters in all classes that require the need for a directional string.

Solution: A possible solution to this issue would be adding a setters and getters in the Actor class to be able to retrieve the directions. However, this could contribute in the reduction of repetition of code in each class when need to use this functionality. More to that, if there were a unified eco points system in the engine that allows a larger value that works for all classes with adding a few getters and setters to retrieve some variables, that could made it easy to avoid repetition and avoid falling into a bunch of dependencies that can cause the system to crash.

Advantages: classes in the game can access attributes in engine without interrupting encapsulation and repeating code as well as avoiding dependencies.