**Recommendations for change to the game engine**

In our opinions, the World class should be in the game package instead of engine package as it is easier to implement certain features such as Mambo Marie, which only occurs after a certain number of turns. In order to implement this features, we had to create a subWorld class which inherits from the World class and overwrite the run() method. Duplicated code is unavoidable as we have to make changes to the method so that it can keep track if MamboMarie is alive, whether or not it exists on the map.

For the Menu() class, it would be great if we can input Strings as the options instead of providing a collection of Actions. The class should give an optional choice where we can add a collection of Actions or a collection of Strings.

One of the main advantages of putting World class in the game package is that it will be easier for us to control the flow of the game. Features such as doubling the speed of the Zombie, freezing the time and other time-related features can be handled by the World class and would be easier to implement because World class has control over the actors’ play turn, and we can skip the other actors’ play turn if we want to make the Zombie moves 2x its speed for example. The disadvantage would be anyone might potentially change the code base and cause the flow of the game to be disrupted.

Besides this, we quite enjoy working with the engine. The parent classes needed for this assignment are provided in the engine, which makes it easy to implement the features and functionality required for this assignment.

Here are a few examples of inheritance we added to help reduce duplicated code, which make it possible forus to adhere to the DRY (Don’t repeat yourself) principle.

WeaponItem: ZombieArm, ZombieMace, ZombieLeg, SnipleRifle and Shotgun, etc.

Action: FarmAction, CraftAction, GroanAction, VanishAction, etc.

With the inheritance, we only have to declare a class and add additional features to it without having to worry if the actors would be able to pick up the WeaponItem and use it, or how the actors choose which action to perform in each turn.

In addition, since inheritance is easy to be implemented, it helps to maintain the Single Responsibility Principle where a class should only be responsible over one functionality. This is very helpful when we tried to understand each other’s codes as all the class and methods are separated which make them easy to be understood.

The tick method() in Item class is also very useful in the code as it helps to count the number of turns that have passed. This helps to reduce dependencies (RED principles) on the World class to count the number of turns.

Other than that, lots of the methods in the engine already contains exception handling, which gives us an error message whenever we make a mistake. This potentially help us to realise the mistakes we made so that we can debug our code before continuing coding. For example, if we were to add an actor in a certain location where an actor is not allowed to enter, the add() method in ActorLocations will throw an IllegalArgumentException() to warn us about the error. This will also save us time as we do not have to waste our time figuring out the error. The exception handling definitely follows the FF (Fail Fast) principle.