Design Document

(Turhan Huseynov 1702232)

Firstly, I would like to mention that I have grouped all soldier types (regular, commando and sniper) under Soldier abstract class and all zombie types (regular, slow and fast) under Zombie abstract class because they had their own specific behaviours that could be grouped and also they were logically same type of objects too. Also I inherited SimulationObjects abstract class by Soldier, Zombie and Bullet abstract classes and defined all the new fields as private and gave access of fields with get and setters of them to subclasses.

Then, I have used 3 arrayLists at SimulationController class to keep active instances of 3 main classes that has been mentioned above. I have used those active instances to for distance calculations, removing dead objects and so. In order to get the type of active class instances at runtime, I have used Reflection API. I created a dummy instance of the class I wanted to check and used “getClass” method on it to get its class and finally checked if it is same with my obj class with equals method.

Most of the SimulationObjects classes have very similar implementation inside them, but zombies have a unique method called commonZombieStep. Since all 3 Zombie classes have a set of shared instructions at the beginning of their step method, I have moved them all those instructions to a seperate method in the parent inherited class.

I have used override on step method of child classes to prevent any potential risk of getting overridden by any parent class. Also i have moved general methods of Soldier and Zombie classes to SimulationObjects such as get random direction,find euclidean distance, find direction, update position and find closest. Since most of those methods are also used by Bullet class, there was no need to have another abstract class between SimulationObjects and Zombie/Soldier classes.