CENG443 – HW1: DESIGN DOCUMENT

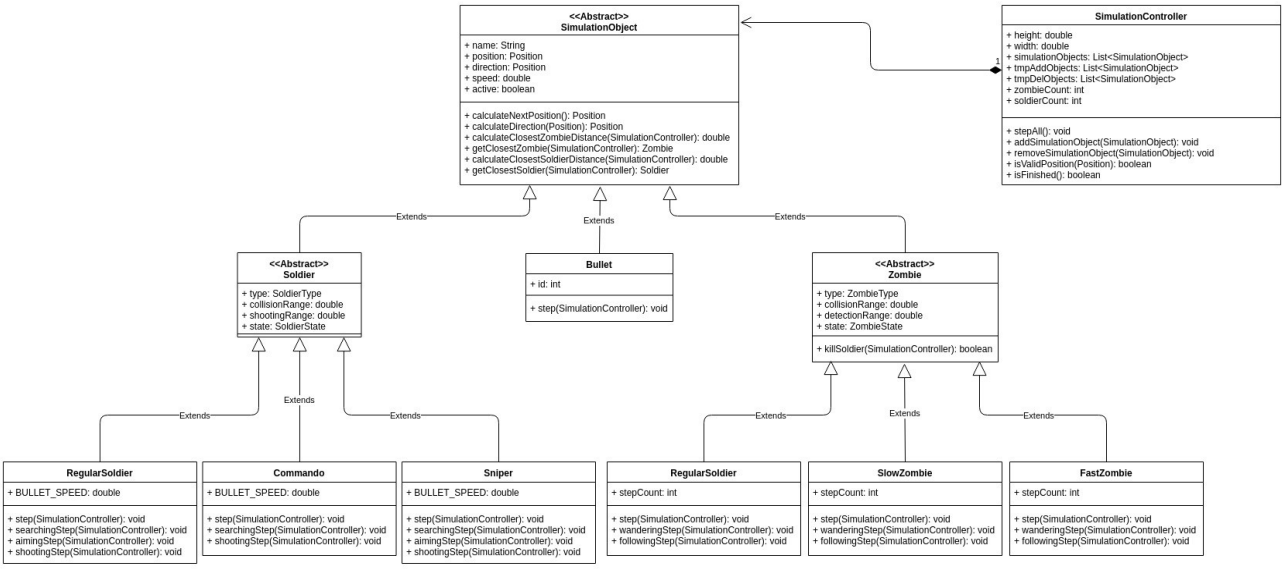

Figure 1: Class diagram of homework 1

As in the class diagram above, I have three **abstract** classes which are SimulationObject, Soldier and Zombie classes. They're abstract because of ***step()*** function as only concrete classes implement it. SimulationObject is **extended** by Soldier, Zombie and Bullet classes. Abstract Soldier class is also extended by concrete soldier classes such as RegularSoldier, Commando and Sniper classes. They implement ***step()*** function with respect to their type and state. Abstract Zombie class is also extended by concrete zombie classes such as RegularZombie, SlowZombie and FastZombie classes. They also implement ***step()*** function with respect to their type and state. Bullet class is concrete class as it implements ***step()*** function. SimulationController class controls SimulationObject instances and their activeness. It has SimulationObject containers and manage the container with add and remove methods.

SimulationObject and SimulationController remind me **MVC** architectural pattern. SimulationObject is **model**, and SimulationController is **controller**. Currently, it lacks viewer but it can be introduced easily in case of need such as GUI etc.

Abstract SimulationObject class contains **final** members: name, speed. They're final as they're initialized at construction and their values won't/cannot be changed after that. However, other members such as *position*, *direction* and *active* are not final because they will definitely change later. The class also contains instance related methods. One of them is *calculateClosestZombie* method and it has similar version for the soldier class. There are also very similar methods for getting the closest zombie and soldier. I could use **polymorphism** for those similar functions. I could also put these methods in SimulationController as it holds all the objects but I didn't design this way. I think it would be better in that way in terms of MVC architectureal design.

Abstract Soldier class contains **final** members: *type*, *collisionRange*, *shootingRange*. They're final because they're initialized at construction and their values won't/cannot be changed after that. However, *state* member cannot be final as it constantly changes.

Concrete soldier type classes contain **static** member *BULLET_SPEED* for each class. I realized that I've made mistake by not moving *step()* function to their parent class. In this way, I would've kept from code repetition.

Abstract Zombie class contains **final** members: *type*, *collisionRange*, detection*Range*. They're final because they're initialized at construction and their values won't/cannot be changed after that. However, *state* member cannot be final as it constantly changes. It also contains *killSoldier()* method that checks whether the closest soldier is killed by that zombie instance.

Concrete zombie type classes don't contain **static** member like soldier type classes do. As it was in concrete soldier type classes, it is my mistake that not having *step()* function to the parent class.

Bullet class is not abstract as it doesn't have any type but it extends *SimulationObject* class. As it is not an abstract class, it implements *step()* method.

In general, I do think that it will be easy to introduce new zombie or soldier types to my design. Their step behavior can be implemented without problem. However, introducing new class such as soldier, zombie or bullet might not be that easy as it will require similar functions to other classes. With newer classes that extends *SimulationObject* class, it would be better use polymorphic methods for similar tasks such as getting closest object, calculating closest object distance etc.