Muhammed Yusuf Temizer

2099356

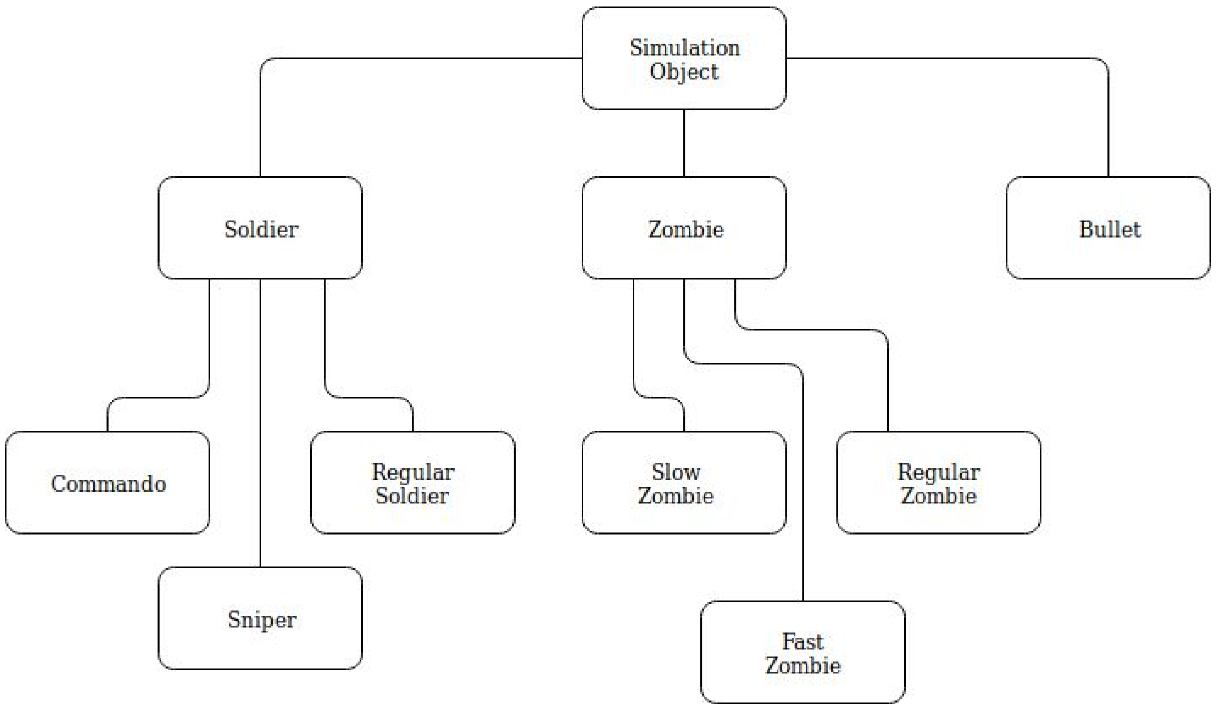
Ceng443 - Introduction to Object Oriented Programming Lang. and Systems

Homework 1 - Design Document

Design Choices:

<u>Design Overview :</u>

- Following diagram shows the general structure of the Simulation Object class and its subclasses.
- Simulation Object is the parent class of the all classes below of it.



Simulation Object :

- Simulation Object is an Abstract Class which has common behaviour and attributes of its sub-classes.
- There are 3 Class that extends Simulation Objects which are Soldier, Zombie and Bullet.
- The aim of creating Zombie and Soldier abstract classes is to be able add new types of soldier or zombie to the simulation easily.
- If we want to add new Zombie or Soldier type, we simply extend the related class and implement only the must be overridden function (step function).

Soldier :

- Soldier is an Abstract Class that contains all the common behaviours and attributes of the Soldier Types.
- Each Soldier has collision range, shooting range, speed and bullet speed.
- There are 3 different state that a Soldier can be in, SEARCHING, AIMING and SHOOTING.
- Regular Soldier, Sniper and Commando extend Soldier class.
- Sub-classes only implements their step behaviour.

Zombie :

- Zombie is an Abstract Class that contains all the common behaviours and attributes of the Zombie Types.
- Each Zombie has collision range, detection range and speed.
- There are 2 different state that a Zombie can be in, WANDERING and FOLLOWING.
- Regular Zombie,Slow Zombie and Fast Zombie extend Zombie class.
- Sub-classes only implements their step behaviour.

Bullet :

- Bullet class directly extends Simulation Object.

Simulation Controller :

- This is the class that contains all the information about simulation. All the relations and connections between objects are handled by this class.

There is a composition relation between Simulation Controller and Simulation Objects. Controller owns any type of Simulation Objects.

- All the zombies, soldiers and bullets are stored in the 3 different Array List in Simulation Controller.
- Since it is the controller, all the adding, removing and stepping all the objects are done in this class.
- And in order to name the bullets properly, there is a bullet count that counts total bullets that have been fired.

Constants :

- This class contains all the constants of the simulation.
- It is created in order to make the code easy to understand.
- With this way, if a constant value is changed, developer doesn’t have to change the value from all the used lines. Instead, just change the value of the constant from constant class.

SOLID (DESIGN) PRINCIPLES :

There are 5 different design principle which are :

- S — Single responsibility principle
- O — Open closed principle
- L — Liskov substitution principle
- I — Interface segregation principle
- D — Dependency Inversion principle

And I used 3 of them in this project.

1 - Single Responsibility Principle :

- A class should have one and only one reason to change, meaning that a class should only have one job. In the subclasses of Soldier and Zombie, they have only one job which is stepping. In each subclass, only step function is implemented so that each of them has only one job to do.

2 - Open Closed Principle :

- In this project, most of the base method and attributions are added and it is fault tolerant.Since most of the base methods and states are handled, there cannot be breaking changes that affect base code. Therefore, the system is closed for modification.

- We are able to add new features or components to the simulation without breaking core system. Therefore, we can say that the system is open for extension. For instance, we can add new type of Soldier, Zombie or Simulation Object without changing the core module.

3 - Liskov Substitution Principle:

- In this project, Simulation Object assures that any subclasses can execute the common methods and Simulation Object class provide those method without knowing the type of subclasses.

OOP Principles:

Inheritance :

- Inheritance concept is applied to the project in the several classes.
- There are 3 abstract classes which are Simulation Object, Zombie, Soldier.
- Each “is a” relation is in the inheritance concept.
- In the simulation:
    * Soldier is a Simulation Object.
    * Regular Soldier is a Soldier.
    * Sniper is a Soldier.
    * Commando is a Soldier.
    * Zombie is a Simulation Object.
    * Regular Zombie is a Zombie.
    * Slow Zombie is a Zombie.
    * Fast Zombie is a Zombie.

Abstraction :

- Simulation Runner has Simulation Controller instance and when the controller is created, runner assumes that all the functionalities of the controller can be used without considering its implementation.
- Simulation Controller has Simulation object instance Arraylists and controller calls their step function without knowing implementation details and assumes they do their job correctly.

Polymorphism :

- When the simulation object is tried to be added to the related list of controller class, each object calls the abstract add class that overridden in each subclasses. With this way, each object can add itself to the related

ArrayList in the Controller class. Therefore, controller is capable of adding objects to the desired list without knowing the type of the object. It is same in the remove method. Remove and Add functions are overridden in each subclasses of Simulation Object (Soldier, Zombie and Bullet).

- There is also a step function that is implemented in the each subclasses of Soldier and Zombie, also in the Bullet class. This is because each subclass has their own unique movement based on their attribution and states.

Encapsulation :

- In the Simulation Object there are private and protected attributions and methods that only the object class can reach and execute. Its subclasses cannot reach the private methods and attributes of Simulation Object.
- In the Zombie and Soldier classes, there are private attributions and methods that only the Soldier or Zombie class can reach and execute. Its subclasses such as Sniper or Fast Zombie cannot reach the private methods and attributes of their parent classes (Soldier or Zombie).