FIT2099 Object Oriented Design and Implementation
Assignment 1
Design Rationale

Group Name: LYYandTSM
Group Members: Lai Ying Ying(30526361)
Tan Sook Mun(30695759)
Lab: Monday,10am

Zombie

Zombie has an AttackBehavior class, and Zombie have many attack actions. The attack actions can be class into AttackBehaviour. AttackBehaviour will call AttackAction in getAction() so it will return the AttackAction that is available to the actor. The pickUpItemAction will allow Actor to pick up Item, which includes WeaponItem. The inheritances used enable all Zombie Actor to pick up Item, which prevents repeating code and adheres to the DRY (Don't Repeat Yourself) principle. Zombie must have 0 to 2 arms and 0 to 2 legs, otherwise throw an exception as this follows the FF (Fail Fast) principle. Zombie has a 30% chance of losing its limbs and this is managed in AttackAction to reduce dependencies from other classes (RED principle). If the target is a Zombie, execute() will call damaged_zombie() to decide whether or not the zombie will lose its limbs. Zombie has an enum of playerAttack. This is to ensure when calling AttackBehaviour, it only calls the attack behaviours that are eligible to Zombie. A bite attack action is added to Zombie's getIntrinsicWeapon method, so that the Zombie can either punch or bite if it does not have any other WeaponItem. The other features of Zombie such as movement speed, picking up item and groaning are added to playTurn method. GroanAction is inherited from Action and called by playTurn().

Player

Inherits from human because a player is human but with more of its own extra functionality The player has an AttackBehaviour. These classes allow the player to have many types of attack. One of which a player can craft weapons. Player is allowed to pick up items as it is inherited from Actor which is associated with the class PickUpItem. When the zombie drops its limb, it can be used as a simple club, therefore ZombieLeg and Zombie Arm are inherited from WeaponItem. Player is able to pick up ZombieLeg and ZombieArm, and either use them as a simple club or craft them into ZombieClub and ZombieMace respectively, which can cause more damage to the target.Player is able to choose when he/she wants to craft. Likewise for food. Player is able to harvest.A class is created called AroundLocation where it returns an array list of available location within range of map. This is for(Dont repeat yourself) this class can be use for HarvestAction.SowAction. In player, the method getWeapon() is override from the actor class. This is so player is able to choose the weapons. Inheriting from a class and overriding a method prevents repeated code (DRY principle).

AttackAction

Both Player and Zombie have AttackBehaviour, therefore they are associated with AttackBehaviour which consists of AttackAction. AttackAction is a class consisting of special Action for attacking other Actors. The damaged_zombie method is added to the AttackAction to determine if the Zombie should lose its limbs. The code to be reused as both zombie and player are able to attack other actors, this is in line with the DRY principles.

ZombieArm, ZombieLeg, ZombieClub & ZombieMace

ZombieArm, ZombieLeg, ZombieClub and ZombieMace classes are created and inherited from WeaponItem as players are able to use ZombieArm and ZombieLeg as it is or craft them into more powerful weapons(ZombieClub and ZombieMace). CraftAction inherits from Action and it is called in Player's playTurn and gives Player options to craft WeaponItem. Zombies can only pick up and use ZombieArm and ZombieLeg as a simple club.

Corpse

A Corpse is inherited from PortableItem. If a Zombie dies, execute() in AttackAction will create a new PortableItem, whereas when Human dies, a new Corpse() will be created. Only Corpse() are able to rise as a Zombie after 10 turns. The tick() is overridden and a counter is added. Inheriting from a class and overriding a method prevents repeated code (DRY principle). After 10 turns, if an actor is carrying the corpse, it will not rise as a Zombie till the actor drops the Corpse down.

Farmer

Inherits from a human but with more of its own extra functionality. Farmer extra functionality is creating food, fertilizing crops, harvesting crops.These extra functionality is added into FarmBehaviour. This FarmBehaviour is an inheritance of the behaviour interface. This FarmBehaviour checks if Farmer is standing next to dirt,crop or ripe crop and return the SowAction, HarvestAction, or FertilizeAction. When a farmer harvest a crop it turn into a Food. Crop extends from ground this way it can inherite the ability that no items can be drop there. Also it is able to inherite the tick() which is useful for counting when it will ripe. This is the Dont Repeat Yourself. This way you dont need another new method for counting the turns. With these inheritance Food inherits the Item functionality and in a way is able to treat it as an Item. This allows the player to pick it up. This also reuses the code and following DRY. When a farmer stands over a patch of dirt it can sow a crop in it. When the farmer harvest the crop, it will drop the food. Because Farmer is inherited from Human class which is also inherited from ZombieActor. ZombieActor is able to drop an Item with the connection to DropItemAction. This will allow the Farmer to drop the food after it is harvested. Before sowing a crop, the code will ensure the farmer is standing next to a dirt if not it will throw an exception. That is why Farmer can interact with the dirt. Also, it is the same for harvesting, it ensures that it has already ripe or the crop is there before harvesting. The SowAction and FertilizeAction is only accessible to the Farmer and no one else for example Player or Zombie. This is following the FF(Fail Fast) principle.

Crop and Food

A Crop has CropCapability which is either Ripe or Unripe. tick() is overridden so a counter is added to count the number of turns. Food is inherited from Item and is portable. A ripe Crop will be replaced by Food and this is managed in the HarvestAction.

MamboMarie

Inherits from ZombieActor. It has a tick function to count the number of turns(for ChantAction and VanishAction). MamboMarie is controlled by SubWorld as SubWorld has full control over the actors' turn. In the run() method, it will check if the MamboMarie is in the map, otherwise it has a 5% chance of appearing. The SubWorld also keeps track if the MamboMarie is alive. If it is dead it would not appear in the map again.

AimAction, TradeAction, UnsowAction, VanishAction

Actions that inherit from the Actions class. This is to follow the Single Responsibility Class principle.

ChooseWeapon & ChooseZombie

This is necessary so that the player can choose the Weapon to use if there's more than one weapon and choose the zombie to aim at.

SniperRifle & Shotgun

This is a weapon item. This class has its own class variable such as bullets,aim,probability, etc. This is to store the items specific values. This is to follow Single Class Responsibility. Because this item is responsible for keeping the number of bullets, whether it is aiming and the probability of shooting.

ShootAction

Is an action that is for sniper rifle and shotgun to shoot. Player is unable to use the attack option if they want to shoot the gun. Because is a shoot action. This class is used for both sniper rifles and shot gun to reduce duplicated codes. This is following the Dont Repeat Yourself rule. The shoot class will search for zombies based on the weapon constraint(eg: shot gun only look for 3 direction for 3 spaces, rifle search the whole map).For getting the shotgun fire direction, it is in the getDirection() method. This method has a try and catch exception. This is the fail fast approach. This method will get 3 direction in the list based on a random number. This method ensures it the random number is a valid index. This class has a method call sniperRifle which basically provide a sub menu for the player to choose the target and aiming action. For this class when the bullet is "fired" and target is chosen it will call for attack action. This is to reuse the class that it already there. Attack Action will then handles the probability of hit or miss

End Game

This class is to end the game. It also inherits from action class. This is to follow Single Responsibility Class principle. Because it is an action for the player to choose. The method checkAlive() will check the map if there is any zombies or humans left. This method is called in attackAction every time when an actor dies. This is so the program does not need to check every actor turn or everytime the loop in the subworld is executed. My interpretation if the question is if any of the map have only zombies then player lost. Same goes for when there is no zombies, the player wins.