# 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 is from the game package. It is a child class of ZombieActor. Zombie has an AttackBehavior class. Because Zombie have many attack actions. The attack actions can be class into AttackBehaviour. This AttackBehaviour class consists of Attack actions such as bite(),punch(),slash(),hit(). The relation between AttackBehaviour and AttackAction is a dependency. Zombie is able to pick up a weapon. So ZombieActor which is the parent of Zombie is able to access pickUpItemAction that is a class in the engine package. The pickUpItemAction will take an Item. Weapon is a child of Item. From this connection, the zombie is able to pick up a weapon. The ZombieActor is associated with the pickUpItemAction because Zombie,Human,Farmer and Player inherit from ZombieActor. This way all these classes are able to pick up items through inheritance. This is also to ensure we do not need to repeat code and following the DRY (Don't Repeat Yourself) principle. When creating a Zombie, ensure that it has 2 arms and 2 legs in the beginning, otherwise throw an exception. Whenever there's a successful bite attack, we should restore 5 health points to the Zombie and reduce the health points when there's a damage. Zombie has an enum of playerAttack. This is to ensure when calling AttackBehaviour, it only calls the attack behaviours that are eligible to Zombie.

## 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. A player has an arraylist of inventory that stores what the player has picked up eg: food, weapon and etc. Player inherit from Human, which is inherited from ZombieActor. Zombie Actor is associated with the class PickUpItem. This is to allow Player to pick up items. When the zombie drops limbs it becomes an item. So when Player is able to pick up zombie limbs and make them into weapons. Player has an enum of playerAttack. This is to ensure when calling AttackBehaviour, it only calls the attack behaviours that are eligible to Player.

## 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 calls the

FarmAction to sow, harvest, or fertilize the crop or dirt. When a farmer harvest a crop it becomes a Food because the Crop is inherited from Food. 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. The Dirt class can have crop on it. Thus it also connected to the Crop class. 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 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 class. Also, it is the same for harvesting, it ensures that it has already ripe or the crop is there before harvesting. The FarmBehaviour is only accessible to the Farmer and no one else for example Player or Zombie. This is following the FF(Fail Fast) 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. Currently, it only consists of the normal punch attack, so adding in other types of attack actions such as bite, slash and hit would enable the code to be reused as both zombie and player are able to attack other actors, this is in line with the DRY principles.

## ZombieClub & ZombieMace

*WeaponItem* is an abstract class which represents items that can be used as a weapon. Therefore both ZombieClub and ZombieMace classes are created and inherited from WeaponItem as players are able to craft new weapons using Zombie's limbs. Only the players can craft ZombieClub & ZombieArm, the code should throw an exception when other Zombies or other Humans try to craft a Weapon. Zombies can only wield their own cast-off limbs as simple clubs.

## Damage

Damage class is created as an attack action may cause damage to the Zombie, causing Zombie to lose its limbs and potentially affects the zombie's movement such as movement speed and attack actions. It is dependent on the AttackAction class because different attack actions can cause different types of damage to the zombie. Damage class is also dependent on DropItemAction because certain arm damages may cause the zombie to drop its weapon. When damage is caused and Zombie drop limbs, ensure that Zombie has less limbs by checking the Zombie class variable limbs.