Player

For the Player Class, we added three new actions for the Player to perform and those are Consume Action, Craft Action and Harvest Action. This follows the design principle “Single Responsibility Principle” as if the Player has to handle multiple responsibilities, more reasons have to be change for the Player Class at some point. Hence this these three Action Classes will be a class on its own.

Human

The Human Actor will contain a new functionality which is to consume items of ConsumableItem Class whenever they are damaged or hurt. It will consume food if it is already in its inventory or pick it up if is not in its inventory.

CraftAction

This class will give players the ability to convert materials into useful items. It will contain methods to accept materials and return an Object of the player’s choosing. It also applies an “Open Closed Principle” so long as the argument given is of MaterialItemClass and it should cater for all types of materials but the core functionality of converting it remains the same.

ConsumeAction

This class will provide players the ability to consume any items of ConsumableItemClass. This class will utilise an item of ConsumableItemClass and take an action with it. Currently, any humans have the ability to consume it. Players can use the action at will. When this action is used, the Consumable Item will be removed from their inventory and the health will be replenished by the item’s value.

Farmer

Farmer is a new type of Human and it extends the Human Actor Class much like the Player. Due to it being an extension of the Human Class, it applies the principle “Don’t Repeat Yourself” meaning it will have the ability to wander as well as consuming items.

Farmer Class will be associated with the FarmerBehaviour Class and this FarmerBehaviour Class will allow the Farmer Actor to act upon crops or ripen crops.

FarmerBehaviour

This behaviour is for the Farmer actors to act upon when approaching crops or ripen crops or standing upon them. In essence, it contains three Action classes which are HarvestAction, SowAction and FertilizeAction. This will return any one of the action or null if nothing is available.

HarvestAction

This action class allows the Player and Farmer Actors to harvest crops. The player will utilise this action on ripen crops which the food will directly add to their inventory. Farmer will utilise this on ripen crops which cause the food to drop on the ground. Once Farmer and Player harvests the crops, the location at which the actors are standing upon will be turn back to a regular Dirt Class.

SowAction

This action class allows the Farmer Actors to sow a crop in the patch of dirt. This Action Class will take in the location of the patch of the Dirt and convert it into a Crop Class. The probability of preforming this action is 33%.

FertilizeAction

This action class allows the Farmer Actors to fertilize unripe crops when standing on top of it. This action will decrease the time left for the crop by 10 turns. This Action Class will take in a Crop Object and increment its age by 10 turns.

Crop

This is a new class created when the Farmer Actors performs the SowAction on a patch of dirt. It will contain an instance variable called “age” to tell the time left for it to be ripe. It will contain four new methods which are getAge(), setAge() (used for the FertilizeAction), isRipe() and changeToRipe(). The tick method has been overridden

to increment the age of the crop and will turn to Ripe when age is 20. Crop will be represented on the Field as “C”.

LegCapability

For our current design and implementation, we have added two new classes which are associated the Zombie Class. These classes are called LegCapability Class and ArmCapability Class.

Adhering to the design principle - “Classes should be responsible for their own properties”. This is because the responsibilities of Zombies are to attack humans and pick up Item. If Zombies have the additional responsibilities to only attack when they have arms or move when they have legs, more changes have to be made to be Zombie Class if additional requirements state so.

LegCapability Class is responsible for the Zombie’s movement in the system where having both legs will assume the Zombie’s normal movement speed, one leg for half its normal speed, and no legs will permanently restrict its movement.

ArmCapability

ArmCapability Class is responsible for the Zombie’s ability to attack depending on the how many arms the Zombie has left. Although attacks that do not utilise the arm will have no effect.

Both these classes will make use of the existing functionality of playTurn method and this complies with the design principle “Don’t Repeat Yourself”. Furthermore, since these classes depend on one another, they have encapsulated in the same package which follows the design principle “Group elements that must depend on each other together inside an encapsulation boundary”.

Since LegCapability and ArmCapability are associated in the Zombie, they will be mostly likely declared as private instance variable which satisfies the design principle “Group elements that must depend on each other together inside an encapsulation boundary” which in this case is in the form of Zombie Class.

Zombie

Zombie Class is equipped with the extended functionality of picking up weapons on the floor if the action is permitted. This functionality did not modify nor change the structure of how Zombie should be but merely added the ability to pick up weapons on sight. This comply to the design principle “Don’t Repeat Yourself” as the functionality to pick up items was already embedded in the Actor Class which extended in Zombie Actor and extended in the Zombie Class. As such this also follows another design principle “Open Closed Principle” where the method for allowing more actors should be open for extension but closed for modifications.

Zombies also have the additional ability to bite instead of their usual punch and have 10% chance of saying brains. Also using the design principle “Avoid excessive use of literals”, the literal constant 10% is used as a variable for easier maintainability.

MaterialItem

Three new classes will be created to support the additional features of what Items can be and there are MaterialItemClass, ConsumableItemClass and CorpseClass.

MaterialItem class is an abstract class which is a base class for items that have the capability to be craftable. The MaterialItem has an additional argument which is a WeaponItem class type which will be used during the crafting action. The WeaponItem is the weapon that is crafted with this MaterialItem. The design to make it an abstract class was so that it could allow its subclasses to have the basic properties of being a material item. This will also follow the design principle “Don’t Repeat Yourself” as any of its subclasses can use their derived class’s method for their functionality. Abstract methods for the class will also be kept to a minimum to allow flexibility for its subclasses when they extend this class. This introduces two new classes which will extend this MaterialItem Class.

ZombieArm

ZombieArm is a material item drop from the Zombie when one of its arms is knocked off. It can be used to craft the Zombie Club as a weapon. This Object is associated with the actions PickUpItemAction, DropItemAction and CraftAction where Player can choose to pick it up, drop it or used it to craft something.

ZombieLeg

ZombieLeg is a material item drop from the Zombie when one of its legs is knocked off. It can be used to craft the Zombie Mace as a weapn. Along with ZombieArm, it is associated with the actions PickUpItemAction, DropItemAction and CraftAction where Player can choose to pick it up, drop it or used it to craft something.

ConsumableItemClass

ConsumableItemClass is an abstract class which is a base class for items that are consumable. Again, the design to make it an abstract class was so its subclasses could inherit its properties and so such complies with the design principles “Don’t Repeat Yourself”. This will introduce a new class to extend this class.

Food

Food Class is a new class to be created which serves as a consumable item and can be used by players or human to heal themselves. This class is created when a Farmer or a Player harvests a ripe crop in that location in which for Farmers, it will drop to the ground where it was harvested. For Players, it will be added directly to their inventory.

Corpse

The corpse class is a unique type of Item Class in a sense that this class is only created when a Human Object is declared DEAD. This class is subclass of PortableItem Class as the corpse can be carry around and place in inventory. So, by making use of existing functionality, we can still comply to the “Don’t Repeat Yourself” principle. Also, any Corpse Object will eventually become a Zombie Actor in 5-10 turns and this can be achieved by applying a method to create new Zombie Actors and place them in the same location as the Corpse Object in the GameMap.

ZombieClub

In addition to ConsumableItem Class and MaterialItemClass, our design also features two new classes which inherit the WeaponItemClass

ZombieClub Class is a subclass of WeaponItemClass and it is was designed based on the “Don’t Repeat Yourself” principle by using the inherited features. This Object

Class can be created using the CraftAction as well as the ZombieArm Object. Features such as damage and attack verb are already present in the system functionality.

ZombieMace

Along with the ZombieClub Class, this makes use of WeaponItemClass by extending it. This Object Class can be created using the CraftAction as well as the ZombieLeg Object.