Design Relation

Hee Weng Sheng 31226507

Crafting Weapons

When I'm implementing the crafting weapons, I had created classes which are ZombieClub, ZombieMace and CraftingAction. I will also modify the Player class to check whether the player inventory contains zombie arm / leg for the player to craft into weapon.

Player Class:

In player class playTurn() method, I will check whether the player inventory contains SimpleClub Object which may be zombie arm or leg. If the player inventory contains it, then the player can add the craft weapon action into the actions arrayList by creating new CraftingAction. So the player can choose to whether they want to craft weapon by entering the hotkeys.

CraftingAction Class:

CraftingAction class extends Action. It contains a one parameter constructor which takes in a SimpleClub Object. Eg. CraftingAction(SimpleClub weapon). Methods in this class:

- execute() – in this method we will create a new WeaponItem based on the SimpleWeapon Object whether it is a zombie arm or a zombie leg. If it is a **zombie arm** we will create a new **ZombieClub.** Whereas a zombie leg, we will create a new **ZombieMace.** After creating the WeaponItem, I will remove the zombie leg / arm based on the which one the player used to craft from the player inventory. If we are holding any weapon we will drop the item and switched to the new Weapon we have just created.
- menuDescription() – in this method we will display a string of description that shows the actor has crafted into which weapon.

ZombieClub and ZombieMace class:

ZombieClub and ZombieMace class, both classes extend WeaponItem. They both contain zero parameter constructor and called the super class constructor which takes in the name of the weapon, displayChar of the weapon, damage of the weapon and the verb of the weapon. There will be no any methods at this point like the **Plank** class. The name, displayChar, damage, and item we will set ourself.

That's how I would design for Crafting Weapon.

Advantages:

- CraftingAction is created so in the future if we want to craft multiple stuff into a single new stuff we can do that in CraftingAction class.
- ZombieMaca and ZombieClub class implementation is just like the Plank class.
- Can craft into stronger weapon make this game more interesting by also having multiple different weapon instead of just having one weapon (Plank).

Disadvantages:

- We only have increase in damage for crafting weapon. Include some interesting skill such as every attack will heal the player some health or gain some movement speed.

Farmer and Food

When I'm implementing the Farmer and Food, I will create 7 classes which are **Farmer**, **FarmingBehaviour**, **FertiliseBehaviour**, **HarvestBehaviour**, **ConsumeAction**, **Crop** and **Food** to handle the task given.

Farmer class is added which extends Human, and has an attribute called behaviors which is an array of Behavior Object. It contains FarmingBehaviour, FertiliseBehaviour, HarvestBehaviour and WanderBehaviour. The class Farmer has the following methods:

- playTurn(Actions actions, Action lastAction, GameMap map, Display display), to perform the action when is the player turn.

In this class, there's a one parameter constructor called Farmer(String name), which takes in the name of the farmer. Then it will call the super class constructor to set the name, displayChar and hitpoints of the Farmer instance.

In the playTurn method, it will have a for each loop to loop through each behaviour and get the action of each individual behaviour by behavior.getAction(actor, map).

FarmingBehaviour, FertiliseBehaviour and HarvestBehaviour classes are created which extends Action and implements Behaviour. In these classes they all have override same methods but with different bodies. These are the following classes with its methods:

FarmingBehaviour:

- getAction(Actor actor, GameMap map) – this method checks whether there is any Dirt beside the actor. If there is, it will return this action.
- execute(Actor actor, GameMap map) – this method set the Ground beside the actor to a new Ground called **Crop**(which is a newly added class).
- menuDescription(Actor actor) – this method return a String that display the action the actor done in the console.

Instance variable:

- destination: Location that store the location of where to sow the Crop.

FertiliseBehaviour:

- getAction(Actor actor, GameMap map) – this method checks whether the actor is standing on a Crop and the age of the Crop is < 20. If it is true, it will return this action.
- execute(Actor actor, GameMap map) – this method will fertilise the Crop and return a String of what the actor has done.
- menuDescription(Actor actor) – this method return a String that display the action the actor done in the console.

HarvestBehaviour:

- getAction(Actor actor, GameMap map) – this method checks whether the actor is a ripe Crop. If it is true, it will return this action.
- execute(Actor actor, GameMap map) – If the actor is Farmer, he/she will harvest the Crop by removing it from the ground and changing the Ground back to Dirt and finally drop the Food on the Ground. If the actor is Player, he/she will harvest the Crop by removing it from the Ground and changing the Ground back to Dirt and finally putting it into his inventory. Lastly, return a String of what the actor has done.
- menuDescription(Actor actor) – return a String that display the action the actor done in the console.

Instance Variable:

- destination: Location, that stores the location of which Crop to harvest.

Food class is newly added and extends Items. It has an instance variable which is Health and it is a constant that shows the amount of health the actor will gain when the actor eat it. It has the method of getHealth() which gets the amount of health will regenerate when the damaged Human and Player consumed.

ConsumeAction is newly added class and extends Action. In this class, it has instance variable of Food that initialize the food that the actor eat. In the execute method which is overridden, when the damaged human/ Player consumed the Food, it will heal the actor by the amount the health contains in the food. Then it will remove from the Ground if the damaged Human consumed it or it will remove from the player inventory when the player consumed it. The menuDescription method is just return a String to display what the actor done.

A **Crop** class is added and extends the Ground. The contains the following methods:

- tick(), which increase the age of the crop by one every turn
- fertilise(), which increase the age of the crop by 10
- getAge(), which returns the age of the crop.

Instance Variable:

- age: int, that store the age of the Crop.

In the fertilise method, I will increment the age of the crop by one and I will check whether the age of the crop is greater than or equal to 20. If is true, then it shows the crop is ripe by change the char to R.

There are few classes I have changed which are **Human**, **Player** and **Application**.

Human Class:
In the method of playTurn(), I will check if the human is damaged or not. If the human is damaged and he/she is standing on a Food, then it will return an Action called ConsumeAction that the Human will eat the food on the ground to regain some health back using the heal() method.

Player Class:
In the Player class I will add a new instance variable in the class which is an array of Behaviours. I choose array because if in the future the player can have more behaviour we can just add it in. I will add a HarvestBehaviour into it to allow the Player to have the HarvestBehaviour so that the player can harvest the ripe crop and put it inside the player inventory. Then in playTurn() method, I will check whether the player inventory contains food. If player inventory contains food then the player have the option whether the player want to consume the food to regain some health.

Application Class:
In the application class I will add Farmer object into the main method so that the farmer can be inside the game.

Advantage:

- Can display what farmer did in that turn in the console.
- Farmer can have more unique actions and behaviours to be implemented in the future.
- Farmer can farm crop and be eaten by player so they can gain some health back.

Disadvantages:

- The farmer can only do one task at one turn. Maybe if it can fertilise and wander at the same time will make the game more interesting.
- Did not display the health of the farmer health and the turns left for the crop to ripe in the console.