EDITING EXISTING CLASSES

Application
Instantiate 4 farmers in main.

Zombie
INSTANCE VARIABLE

1. A ZombieLimbs class was added. This class represents the limbs that the zombie has.
   //SHould it be added to constructor???

METHODS

1. public IntrinsicWeapon getIntrinsicWeapon()

There'll a 50 percent chance that the zombie will bite. If it doesn't bite, it will punch instead.

2. public String zombieLoseLimbs()

This method calls the ZombieLimbs method called loseLimbs(). zombieLoseLimbs will return the String of the limbs lost.

3. public Action playTurn(Actions actions, Action lastAction, GameMap map, Display display)

A random double, "say" was created. If "say"<=0.1(10%), then Braaaain will be printed out. The rest of the method remains the same. If an action can be carried out, that action will be carried out.

//////Association between zombie and zombieLimbs.

AttackAction
METHODS

1. public String execute(Actor actor, GameMap map)

The actor will first get the weapon they have. If the actor is a zombie and the zombie's next attack is a bite, the zombie only has a 25% chance of a successful bite. If a successful bite occurs, then the zombie will gain 5 hitpoints. If the actor is a human or the actor is a zombie whose weapon is not a bite, there's a 50% chance that they will hit. The rest of the method remains the same. If there is a successful hit, their victims will lose hit points depending on the attack on them. If the target's hitpoint is less than 0, the target will turn into a corpse and all their items are dropped.

Instead of creating a new PortableItem for corpse, a Corpse object is instantiated to represent the corpse. This was changed because the corpse should turn into a zombie in 5-10 turns so we needed to keep track of the corpse's age to do so.

Human
HarvestBehaviour added to the collection of behaviours.

NEW CLASSES

Farmer

Inherits from the Human class because a Farmer object should have the same characteristics as any other Human but is able to plant, fertilize and harvest crops.

ATTRIBUTES

1. **Behaviour[] behaviours**
   A collection of Behaviour objects that allow the farmer to sow and fertilize. It will contain SowBehaviour and FertilizeBehaviour.

METHODS

1. **Farmer(String name)**
   Constructor to make a Farmer object. Calls super(name, 'F', 80). The displayChar is chosen to be 'F" so that one can tell the difference between an ordinary human and a farmer. It'll be good for the player to know if there are any farmers left in the game. The hitPoints we chose is higher than an ordinary Human because farmers shouldn't be so easily killed as they are essential actors that provide the opportunity to heal, but it is still lower than the player's to make it challenging.
2. **playTurn(**

Crop

Inherits from Ground class because it grows on the ground.

ATTRIBUTES

age - to be able to tell how ripe a crop is.

METHODS

1. **Crop()**
   displayChar starts off as 'c'
2. **@Override**
   tick(Location location)
   Implemented to change its displayChar as the crop ripens. Once its age turns to 10, the displayChar is changed to 'g'. After age turns 20, it is ripe so the displayChar changes to '8'. We wanted to use characters that were similar to each other but can also can be told apart. If we had simply used 'c' and 'C', it would be difficult to tell when it is ready for harvest.

SowBehaviour

Behaviour only exhibited by Farmer. Checks adjacent location around the farmer. If ground is dirt, there's a 33% chance that SowAction is created and a crop is planted.

SowAction

A new Crop object is created and the ground is set as Crop.

FertilizeBehaviour
Behaviour only exhibited by Farmer. Checks if Location where Actor is standing on is a Crop. If it is, then Fertilize Action is created.

FertilizeAction
If farmer is standing on an unripe crop, the farmer fertilizes it and so decreases the crop's time left to ripen by 10 turns.

HarvestBehaviour
Behaviour exhibited by Human. When standing next to or on a ripe crop, the actor may harvest the crop for food to heal damage.

HarvestAction
Food, a PortableItem object, is created. HealAction is created and added to the food's allowable actions.

HealAction
Heals the actor's damage by an amount specified.

ZombieLimbs
This class represents the limbs that the zombie has. By default, the zombie has 2 hands and 2 legs.

<u>INSTANCE VARIABLE</u>

1. private ArrayList<String> zombieLimbs= setZombieLimbs();

This represents the list containing all the limbs that the zombie has.

2. private int noOfHands=2;

This represents the number of hands the zombie has.

3. private int noOfLegs=2;

This represents the number of legs the zombie has.

4. private int noOfLimbs=4;

This represents the total number of limbs the zombie has.

<u>METHODS</u>

1. public ArrayList<String> setZombieLimbs()

This method will initialize the zombieLimbs ArrayList.

2. public String loseLimbs()

This method will randomly cause the zombie to lose its limb. The number of hands, legs, and total limbs will be updated. This method will return the limb that the zombie lost.

3. public int getNoOfHands()

This method returns the number of hands the zombie has.
public int getNoOfLegs()
This method returns the number of legs the zombie currently has.

4. public int getNoOfLimbs()

This method returns the number of legs the zombie currently has.

Corpse

Inherits from PortableItem because it can be picked up by the player.

ATTRIBUTES

1. **age**
   A counter for how many turns it has been a corpse
2. **chance**
   Chances (in decimal) of the corpse becoming a zombie. It starts at 0, and after five turns the chances increase by 20% meaning by 10 turns, the corpse will definitely have turned into a zombie.
3. **rand**
   Random double generator.

METHODS

1. **@Override**
   tick(Location currentLocation, Actor actor)
   Checks location around currentLocation and calls generateZombieAtLocation() with that empty location and the corpse should be dropped from the actor's inventory. If no empty location is found, the game will be over because the player would be trapped with a zombie.
2. **@Override**
   tick(Location currentLocation)
   Calls generateZombieAtLocation(currentLocation) because if the corpse was on the ground, the zombie should be generated at that same coordinate.
3. **generateZombieAtLocation(Location location)**
   age will increment. If age is more than 5, chance will increase by 0.2. If the random number generated is within the range, a Zombie object is created at the location specified.
   This method was implemented in regards to the Do Not Repeat Yourself Principle because no matter if the item is in an inventory or on the ground, a zombie has to be generated.