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???
2. A new behavior called pickUpItemBehaviour will be added to the behaviors list.
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 zombie and its next attack is punch, we will first check the number of arms the zombie has. If the zombie has 2 arms, the chances of success remain the same which is 50%. If the zombie has 1 arm, the chances of hitting are halved, which means the chances of failure is 75%, If the zombie has no arms, it is confirmed that the zombie has failed in attacking. If the actor is a human, there's a 50% chance that they will hit. If the hit was a success, there is a 25% chance that the zombie will lose its limbs. It will call lossLimbs() in the zombie class and that method will randomly make the zombie lose one of its limbs. Whichever limb the zombie loses, the limb will drop at the zombie's location. If the zombie losses its arms, we will first check if the zombie has any weapon. If the zombie has a weapon and the zombie has 1 arm, there'll be a 50% chance of the zombie. If the zombie has a weapon and has 2 arms, it will definitely drop its weapon. If the zombie loses a leg, then the zombie can move once every two moves, but can still attack as usual. If the zombie loses both legs, the zombie cannot move at all, although it can still attack.

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 using tick to do so.

Human
Add HealBehaviour and PickUpItemBehaviour.
Change playTurn method so that it iterates through 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, FertilizeBehaviour and HarvestBehaviour.
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(GameMap map)**
   Calls super to inherit other Human behaviours. Should iterate through collection of Behaviours to return an Action for that turn.

Crop
Inherits from Ground because it should not be able to be picked up by player.
ATTRIBUTES
1. **int age**
   Number representing the number of turns since the crop has been sown. Initialised to 0.

METHODS
1. **Crop()**
   Calls super('v') because displayChar starts off as 'v'.
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 'V'. After age turns 20, it is ripe so the displayChar changes to 'Y'. We wanted to use characters that were similar to each other but can also be told apart.
3. **@Override**
   allowableActions(Actor actor, Location location, String direction)
   Calls super. But if age of crop is less than 20 and if actor is Farmer, then allowableActions include FertilizeAction. If the age of the crop is more than 20 and the actor is Farmer or Player, then allowableActions include HarvestAction.

SowBehaviour

Inherits from Behaviour and behaviour is only exhibited by Farmer. Must have random number generator for the chances a crop is sown.

METHODS

1. **@Override**
   getAction(Actor actor, GameMap map)
   Checks if the ground is dirt at each exit from the actor. If it is dirt and rand is less than 0.33, SowAction is created.

SowAction

Inherits from Action, instantiated by SowBehaviour.

ATTRIBUTES

1. **Location cropLocation;**
   The coordinates where the crop is to be sown.

METHODS

1. **SowAction(Location location)**
   Initialise cropLocation to location.
2. **@Override**
   execute(Actor actor, GameMap map)
   Creates a new Crop object. Set ground at plantLocation as Crop.
3. **@Override**
   menuDescription(Actor actor)
   Return a string saying actor sows a crop.

FertilizeBehaviour

Inherits from Behaviour and behaviour is only exhibited by Farmer.

METHODS

1. **@Override**
   getAction(Actor actor, GameMap map)
   Checks if location where actor is standing on is a Crop. If it is, then FertilizeAction is returned, else, null.

FertilizeAction

Inherits from Action, instantiated by FertilizeBehaviour.

ATTRIBUTES

1. **Location cropLocation**
   The location of the crop to be fertilized.

METHODS

1. **FertilizeAction(Location location)**
   Initialize cropLocation as location.
2. **@Override**
   execute(Actor actor, GameMap map)
   A loop calling cropLocation.tick() 10 times (The crop's time left to ripen is decreased by 10 turns).

3. **@Override**
   menuDescription(Actor actor)
   Return a string saying actor fertilizes a crop.

HarvestBehaviour
Inherits from Behaviour and behaviour is only exhibited by Farmer.
METHODS
1. **@Override**
   getAction(Actor actor, GameMap map)
   Checks if location where actor is standing on and location of exits is a ripe crop. If it is, then HarvestAction is returned, else, null.

HarvestAction
ATTRIBUTES
1. **Location cropLocation**
   Location of ripe crop.
METHODS
1. **HarvestAction(Location location)**
   Initialise cropLocation with location.
2. **@Override**
   execute(Actor actor, GameMap map)
   A Food object is created. If the actor is a Farmer, the Food is dropped at cropLocation. If the actor is Player, the Food is added to Player's inventory. The ground at cropLocation is set to Dirt.
3. **@Override**
   menuDescription(Actor actor)
   Return a string saying actor harvests a crop.

HealBehaviour
Inherits from HuntBehaviour (re-use of code follows Do Not Repeat Yourself principle) and behaviour is exhibited by Human.
If Human is at 50% health, then they will 'hunt' for Food type objects.

PickUpItemBehaviour
Inherits from Behaviour and will have a target class. Used by Zombie to pick up WeaponItems and used by Humans to pick up Food (Re-use of code follows Do Not Repeat Yourself Principle).
METHODS
2. **@Override**
   getAction(Actor actor, GameMap map)

Checks if location actor is standing on has instance of the target class. If it does, a PickUpItemAction will be returned, otherwise, null.

HealAction
Inherits Action and is called when an item heals an actor.
<u>ATTRIBUTES</u>
1. **Item healingItem**
   The item which does the healing.
2. **Int pointsHealed**
   The number of points the actor is going to get healed by.
<u>METHODS</u>
1. **HealAction(Item item, int points)**
   Initialise healingItem to item and pointsHealed as points.
2. **@Override**
   execute(Actor actor, GameMap map)
   Calls actor.heal(pointsHealed), then removes healingItem from the actor's inventory.
3. **@Override**
   menuDescription(Actor actor)
   Returns a string saying actor healed by pointsHealed.

Food
Inherits from PortableItem and adds HealAction to allowableActions.

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. A bit of random chance when the corpse will turn into a Zombie but within 5-10 turns.

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.

CraftAction

This is a class that inherits from action class. This class allows the player to craft weapon from zombie limbs.

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

The player will first check whether it has a zombie limb in the inventory. (List<Item> inventory=actor.getInventory. Iterator iterator=inventory.iterator().while(iterator.hasNext()) if(iterator.next.equals("rightHand")||iterator.next.equals("leftHand")||iterator.next.equals("right Leg")||iterator.next.equals("leftLeg")){
}
)
)