Design Rationale

Jiten Verma and Ayesha Ali

Zombie attacks

Feature Number	Feature description
1	Zombies should be able to bite. Give the Zombie a bite attack as well, with a 50% probability of using this instead of their normal attack. The bite attack should have a lower chance of hitting than the punch attack, but do more damage – experiment with combinations of hit probability and damage that makes the game fun and challenging. (You can experiment with the bite probability too, if you like.)
2	A successful bite attack restores 5 health points to the Zombie
3	If there is a weapon at the Zombie's location when its turn starts, the Zombie should pick it up. This means that the Zombie will use that weapon instead of its intrinsic punch attack (e.g. it might "slash" or "hit" depending on the weapon)
4	Every turn, each Zombie should have a 10% chance of saying "Braaaaains" (or something similarly Zombie-like)

Feature 1

The bite feature will be implemented by creating a new IntrinsicWeapon object with the name "Bite" and damage of 15. The getInstrinsicWeapon method in Zombies class will be modified to have a 50% chance of selecting punch and 50% chance of selecting bite. These probabilities will be stored as private integers.

Furthermore, the IntrinsicWeapon class would be modified to have a new attribute named accuracy. This will be used to determine if an attack resulted in a hit or a miss. The reason for this design decision is implementing accuracy will become a lot easier and more consistent. This gives us the ability to allocate the accuracy for all individual attacks.

Feature 2

Add an attribute to the IntrinsicWeapon class called heal. Heal is an integer which stores the number of hit points that should be restored when the weapon successfully hits. The constructor of the IntrinsicWeapon class will be modified to allow heal to be passed through. If heal is not passed into a constructor it has a default heal value of 0. Finally, the IntrinsicWeapon class would require a method which returns the value of heal.
Actor already has an existing method called heal which we can call in AttackAction to heal the attack user.

The reasons for this design choice is to clearly define the heal amount for certain attacks so that it is easily accessible and stored in a consistent manner.

Feature 3

Modify the playTurn method in the Zombie class so that it iterates through all possible actions. If one of the possible actions is pickUpItemAction, then this action will be executed (using the execute method), and the zombie will pick up the weapon and add it to the inventory (this will not count as the Zombie's turn).

The getWeapon method in Actor is already optimised to give weapon selection priority over IntrinsicWeapon.

Feature 4

Create an array called ZombieNoise which stores a few noises that zombies make.
Modify the playAction method in the Zombie class so that it produces a random integer between 1-100. If the integer is between 1-10 inclusive, then produce a random integer between 0 to the size of ZombieNoise to select a random noise for the zombie to make. This can be printed onto the game.

This design choice allows for multiple zombie noises to be stored so that there is variation and the gamer experience is enhanced.

Beating up the zombies

Feature Number	Feature Description
1	Any attack on a Zombie that causes damage has a chance to knock at least one of its limbs off (I suggest 25% but feel free to experiment with the probabilities to make it more fun)
2	On creation, a Zombie has two arms and two legs. It cannot lose more than these.
3	If a Zombie loses one arm, its probability of punching (rather than biting) is halved and it has a 50% chance of dropping any weapon it is holding. If it loses both arms, it definitely drops any weapon it was holding.
4	If it loses one leg, its movement speed is halved – that is, it can only move every second turn, although it can still perform other actions such as biting and punching (assuming it's still got at least one arm)
5	If it loses both legs, it cannot move at all, although it can still bite and punch
6	Lost limbs drop to the ground, either at the Zombie's location or at an adjacent location (whichever you feel is more fun and interesting)
7	Cast-off Zombie limbs can be wielded as simple clubs – you decide on the amount of damage they can do

Feature 1

Create an override method in the Zombie class for “hurt” and pass the map and the amount of damage. In this hurt method we can get a random integer between 1-100. If the integer is between 1 - 25, then a method named dropLimb will be called.
The dropLimb method will also be implemented in the Zombie class. A random integer will be generated between 1 - 20. If the number is between 1-10, only one limb will be knocked off. If the number is between 11-17, then 2 limbs will be knocked off. If the number is between 18-19,then 3 limbs will be knocked out. Finally, if the number is 20 then all 4 limbs will be knocked out.
It will randomly select either an arm or leg, depending on which body parts the zombie still has, and knock it out.

The reason for this design choice is that it allows for a unique hurt method in zombie, which will affect zombie actors only and allows us to program unique features for zombies.

Feature 2

Two new attributes are introduced to the Zombie class; Arms and Legs. Upon initialisation these have an integer value of 2 to represent 2 arms and 2 legs.
Two other attributes are also introduced; minArms and minLegs which will be initialised to 0. This will help ensure that the number of limbs never falls below zero.

Feature 3

A method in the Zombie class called accountForLostLimbs will be called and all modifications needed due to lost limbs will be fulfilled.
A method in the Zombie class will be added called modifyIntrinsicWeaponProbability which will modify the probability of punch to 25% and bite to 75% if the attribute arm is equal to 1. If the attribute arm is equal to 0 then modifyIntrinsicWeaponProbability will modify probability of punch to 0% and bite to 100%.
If the attribute arm is equal to 0 then dropItemAction will be called to drop any weapons being held.

Feature 4

An attribute will be added to Zombie which stores a boolean value called canMove. This will be initialised as True. If this value is true, then the zombie will be able to perform WanderBehaviour actions (moving around). If this value is False, then the zombie would be unable to move around essentially.

A method will be added to the Zombie class called modifyMovementSpeed. If the number of legs is equal to 1. Then the value of canMove will be inverted (False now becomes True, and vice versa). This means that canMove will be True every second turn. If the number of legs is 2 then ensure canMove is True.

Feature 5

If the number of legs is 0 then ensure canMove is False using the modifyMovementSpeed method.

Feature 6 and 7

The dropLimb method created previously initialises a new arm or leg as a weapon and drops this onto the map where the zombie is standing. These weapons can be picked up and used as simple clubs. The damage an arm does is 10 and the damage a leg does is 15.

Crafting weapons

Feature Number	Feature Description
1	If the player is holding a Zombie arm, they can craft it into a Zombie club, which does significantly more damage.
2	If the player is holding a Zombie leg, they can craft it into a Zombie mace, which does even more damage

Feature 1

Create a new class called Craft which extends Actions.
The Player holds a zombie arm which is an object of Weapon. We need to create a class called CraftZombieClub which extends Craft which is an action for humans, if they have a zombie arm as a weapon in their inventory then they can craft a Zombie club. If they select this action, then we create a new weapon object in Human class called Zombie club with 25 damage and remove the zombie arm from the player's inventory without dropping it to the map.

Feature 2

Use the created class called Craft.
The Player holds a zombie leg which is an object of Weapon. We need to create a class called CraftZombieMace which extends Craft which is an action for humans, if they have a zombie leg as a weapon in their inventory then they can craft a Zombie mace. If they select this action, then we make a weapon object in Human class called Zombie mace with 35 damage and remove the zombie leg from the player's inventory without dropping it to the map.

Rising from the dead

Feature Number	Feature Description
1	As everybody knows, if you're killed by a Zombie, you become a Zombie yourself. After a Human is killed, and its corpse should rise from the dead as a Zombie 5-10 turns later.

Feature 1

In the human class and the zombie class, a new method called isHuman will be added. This returns False in the Zombie class and returns True in the human class.
A new class called Corpse will be created within the game package which extends PortableItem. The existing Corpse Item will be refactored so that it is produced by this class instead. The Corpse class will have an additional attribute called revivalTime which stores the number of turns remaining until the Corpse can be revived. We can create an override tick method in corpse to reduce the number of turns each turn.

In AttackAction, a random value between 5-10 will be passed into the constructor of Corpse if a human dies to initialise the value of revivalTime.

Farmers and food

Feature Number	Feature Description
1	When standing next to a patch of dirt, a Farmer has a 33% probability of sowing a crop on it
2	Left alone, a crop will ripen in 20 turns
3	When standing on an unripe crop, a Farmer can fertilize it, decreasing the time left to ripen by 10 turns
4	When standing on or next to a ripe crop, a Farmer (or the player) can harvest it for food. If a Farmer harvests the food, it is dropped to the ground. If the player harvests the food, it is placed in the player's inventory
5	Food can be eaten by the player, or by damaged humans, to recover some health points

Feature 1

Create a new class Crop which is an extension of Item and a new class called Farmer which is an extension of Human. Determine location using the Location class to get the coordinates of farmer's location and use that to check if there is a valid spot in which a crop can be sown near the farmer which is the one with just Dirt. For the probability, generate a random number from 1 to 100. If the number is between 1 and 33 and a valid location is found then make an object of crop in farmer represented by character "U" and make a variable to check if crop is ripe or not.By default set that variable to False.

Feature 2

Create a variable in the Crop class which will keep track of the number of turns taken for the crop to be ripe. This variable will be initialised as an integer value of 20. Each time a turn is played, the variable is decremented by 1, using a method called grow and once the variable becomes 0, the variable cannot be decremented anymore. Change the variable which keeps track of whether the crop is ripe or not to true and change the character of the crop to "R".

Feature 3

Add a new class called Fertilize which extends Action for the Farmer class which a farmer would be able to perform if they are standing on or next to an unripe crop. This action will call a method in the Crop class called executeFertilize, which decreases the number of turns remaining by 10 rather than 1.
In the playTurn method in farmer modify it so that if this action exists in it's list of actions then it automatically gets performed (without counting as a turn for the farmer)

Feature 4

Create a new class called HarvestedCrop which extends Item. Create a new class called Harvest which extends Action for Farmer, if he stands near a crop which is ripe, he will be able to harvest it. In the playTurn method in the Farmer class, iterate through all actions and automatically call the action mentioned above which will create a new HarvestedCrop object and drop it on the map with the letter "F" and remove the existing Crop object.
Repeat this process for the Player class so they can also Harvest crops.
If a player comes next to a HarvestCrop object they will have an Action to pick it up using PickUpItemAction.

Feature 5

A method called heal already exists in the Actor class. Use PickUpItemAction to place the harvested crop in the human or player's inventory. A new class called consumeHarvestedCrop which extends Action must be added allowing them to consume the harvested crop to regain some hit points. This action will appear in the list of doable actions and the player will need to use a turn to consume the crop. If this action is called then the heal method in Actor will be used to give some number of hitPoints which is less than maxHitPoints.