**Zombie Attacks**

Created new classes ZombieAttackBehaviour and BiteAction. Change Zombie class to use ZombieAttackBehaviour instead of AttackBehaviour. Added constant attributes (dialogue and chance to return dialogue) to Zombie class and changed playTurn() method to have a chance to print dialogue. Added new methods to prevent repetition to AttackBehaviour and AttackAction to follow by DRY principle.

**ZombieAttackBehaviour**

* Inherits AttackBehaviour
* Has a chance to perform either AttackAction or BiteAction
* Before beginning turn, uses map.locationOf(actor).getItems() to find the items available at the location where the zombie is standing on. If the list is not empty, use item.asWeapon() method to check if items are weapons. First non-null Item use PickUpAction.execute(Actor actor, GameMap map)

**BiteAction**

* Inherits AttackAction
* Has constant miss chance, bite damage and health restored

Before beginning each turn a zombie will reuse methods (map.locationOf().getItems()) from Location class to obtain items on the spot it is standing on to pick up weapons.

**Beating up the Zombies**

Created new classes ZombieArm and ZombieLeg.

**ZombieArm**

* Inherits WeaponItem

**ZombieLeg**

* Inherits WeaponItem

**Zombie**

* Added 3 new attributes to Zombie class (armsNumber, legsNumber and isSecondTurn)
* Add new method to either reduce number of arms or number of legs
* Override the hurt() method from Actor parent class to add 25% chance to call method to reduce limbs
* Add addItem() to hurt() to drop limbs on the map accordingly
* Add getDropAction() to hurt() method to drop its weapon depending on the number of arms remaining
* Change ZombieAttackBehaviour constructor to accept number of arms as a parameter
* Change ZombieAttackBehaviour to increase miss rate and use of BiteAction accordingly
* Change Zombie class playTurn() to check legsNumber before going into HuntBehaviour and WanderBehaviour. Use a Boolean isSecondTurn to swap between True and False to choose when to move. Completely remove movement if both legsNumber == 0

**Crafting weapons**

Created new class CraftingAction, ZombieClub and ZombieMace. Added craftable item as an attribute of ZombieArm and ZombieLeg respectively.

**CraftingAction**

* Inherits Action
* Constructor takes in the original item and the craftable item (from the attribute of the original item)
* Overrides execute() from parent class Action to remove original item from inventory of actor and adds craftable item to the inventory of actor

**Rising from the dead**

Created new class ZombieCorpse. Changed ZombieAttackAction to produce ZombieCorpse instead of corpse Item when knocking other actors unconscious.

**ZombieCorpse**

* Inherits Item
* Has two attributes conversionTurn and conversionCounter
* conversionTurn is a random integer between 5-10
* Override tick() method from parent class Item to increment conversionCounter every game tick
* Has comparison in tick() to check if conversionCounter == conversionTurn
* Has method to remove ZombieCorpse from the location and create a Zombie actor at the location
* Once conversionCounter == conversionTurn, call conversion method

**Farmers and food**

Changed playTurn() method in Human class to pick up Food on the ground and eat if damaged.

**Farmer**

* Inherits Human
* Has FarmerBehaviour as an attribute

**FarmerBehaviour**

* Implements Behaviour interface
* Has 3 actions in descending priority in its getAction() method

**SowAction**

* Inherits Action
* Has attribute sowProbability which is set by constants
* Has method to change Dirt to Crop

**FertilizeAction**

* Inherits Action
* Has attribute turnsReduced which is set by constants
* Has method to reduce turns of Crop to be ripe

**HarvestAction**

* Inherits Action
* Has method to check if crop is ripe. If ripe, converts Crop back to Dirt and adds Food on the ground

**Crop**

* Inherits Ground
* Has 2 attributes, ripeCounter and ripeAge integer constant
* With every game tick, ripeCounter increases by 1
* Has method isRipe() to return whether the crop is ripe

**Food**

* Inherits Item
* Has 1 attribute, nutrients integer constant
* Has EatAction in getAllowableActions()

**EatAction**

* Inherits Action
* Overrides execute() method from Action parent class to remove Food from inventory and add health to the actor performing the action

**PlayerHarvestAction**

* Inherits HarvestAction
* Performs the same actions as HarvestAction except it makes it usable by Player actors, eg. adding Food to player’s inventory instead of dropping it on the ground