**Zombie Attacks**

Created new classes ZombieAttackBehaviour and ZombieAttackAction. 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 and return a DoNothingAction. Added new methods to prevent repetition to AttackBehaviour and AttackAction to follow by DRY principle.

**ZombieAttackBehaviour**

* Inherits AttackBehaviour
* Uses ZombieAttackAction

**ZombieAttackAction**

* Inherits AttackAction
* Has constant miss chances, bite damage and health restored
* Has a chance to either use a normal attack or a bite attack

**Beating up the Zombies**

Created new classes ZombieArm and ZombieLeg.

**ZombieArm**

* Inherits WeaponItem

**ZombieLeg**

* Inherits WeaponItem

**Zombie**

* Added 4 new attributes to Zombie class (armsNumber, legsNumber, LIMB\_LOSS\_CHANCE and isSecondTurn)
* Add new loseLimb method to either reduce number of arms or number of legs
* Change AttackAction to check if target is a Zombie and calls loseLimb
* Change ZombieAttackBehaviour constructor to accept number of arms as a parameter
* Change ZombieAttackAction constructor to accept number of arms as a parameter
* Change ZombieAttackAction to increase miss rate and use of bite attacks
* 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 by using getCraftItem() method)
* 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 PortableItem
* Has conversionCounter
* conversionCounter is a random integer between 5-10
* Override tick() method from parent class Item to decrease conversionCounter every game tick
* Has comparison in tick() to check if conversionCounter == 0
* Has method to remove ZombieCorpse from the location and create a Zombie actor at the location
* Uses both tick() from when the ZombieCorpse is on the ground or in an Actor’s inventory
* Calls the removal method when conversionCounter == 0

**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(HarvestAction, SowAction, FertilizeAction) in its getAction() method
* The probability of using sowAction is generated by creating a double(that is equal or more than 0 but less than 1) using Math.random()

**SowAction**

* Inherits Action
* Has an attribute target, which is the Location to be sowed
* Has method to change Dirt to Crop

**FertilizeAction**

* Inherits Action
* Has attribute target which is the Crop to fertilize
* Has method to reduce turns of Crop to be ripe

**HarvestAction**

* Inherits Action
* Has attributes which take in the crop to be harvested and location of the crop
* 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 an attribute ripeAge integer constant
* With every game tick, ripeAge decrements by 1
* Overrides tick() method to add HarvestAction when it is ripe
* Has method isRipe() to return whether the crop is ripe

**Food**

* Inherits Item
* Has 1 attribute, NUTIRENTS 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

**Going to Town**

**SolidTerrain**

* Abstract class inherits Ground
* Overrides canActorEnter to prevent Actors from entering

**HorizontalWall, Path, Floor, Vertical Wall**

* Inherites SolidTerrain

**Vehicle**

* Inherits Item

**Application**

* Added new town GameMap with added terrain
* Added a new Vehicle Item with a MoveActorAction on both GameMaps

**Shotgun and Sniper Rifle**

**GunItem**

* Abstract class inherits WeaponItem
* Has methods to get respective Ammo type, Action type and general method to reduce the respective ammo rounds

**Ammo**

* Abstract class inherits PortableItem
* Has methods to get rounds and reduce rounds

**ShotgunAmmo**

* Inherits Ammo

**SniperAmmo**

* Inherits Ammo

**Shotgun**

* Inherits GunItem
* Overrides getAmmo to return ShotgunAmmo
* Overrides getAction to return ShotgunAction

**ShotgunAction**

* Inherits Action
* Overrides execute method to return a submenu with ShotgunShootActions in each Exit direction from the Player’s location

**ShotgunShootAction**

* Inherits Action
* Overrides execute method to loop over the required area using Exits
* Adds the affected locations into an ArrayList
* Sets any Ground that blocks thrown objects to a Path instance
* Creates a new ShotgunAttackAction for every Actor in the affected area

**ShotgunAttackAction**

* Inherits AttackAction
* Overrides execute method to damage target using protected methods

**SniperAction**

* Inherits Action
* Overrides execute to loop through the whole GameMap and adds a SniperShootAction for each valid target as a submenu
* If has previous aims, return the previous Actions for aiming and shooting
* If not aiming or shooting at the same target, clear aims

**SniperShootAction**

* Inherits AttackAction
* Overrides execute method to damage or kill target based on number of aims

**Player**

* Changed playTurn method to check inventory for GunItems which are loaded
* Checks for number of aims to return the previous SniperAction if required
* Adds the respective Actions from each GunItem into the menu