Design Rationale

Jiten Verma and Ayesha Ali

Going to town

Feature Number	Feature description
1	Place a vehicle somewhere on your existing map. The vehicle should provide the player with the option to move to a town map. The design of the town can be customised
2	Somewhere on the town map, place a sniper rifle and a shotgun.
3	When you leave a map, any creatures in the old map should continue to move and act.

Feature 1

A new class called Vehicle which extends item would be created, which would require the map to be transported to. The constructor class would add an allowable action for MoveActorAction to allow us to transport the actor onto the town map.
In application, the item of Vehicle will be created.
Modify the Application class to create a new map and add the townMap into the World.

The choice of creating a new Vehicle class was to ensure that the vehicle item could be created again if necessary without having to copy and paste code. This would save us time from having to refactor it in the future.

Feature 2

On the map created in Application, place sniper rifle and shotgun in a randomly selected location.

Feature 3

The run method in the World class ticks all maps stored already every turn.

New weapons: shotgun and sniper rifle

Feature Number	Feature Description

1	Create a shotgun weapon
2	Shotgun spread: The shotgun has a short range, but sends a 90◦ cone of pellets out that can hit more than one target – that is, it does area effect damage. Rather than being fired at a target, the shotgun is fired in a direction. Its range is three squares. So, if the shotgun is fired north, it can hit anything in the three squares north of the shooter, northeast of the shooter, northwest of the shooter, or anything in between.
3	It has a 75% chance of hitting any Actor within its area of effect and decides the amount of damage for a shotgun.
4	Create sniper rifle weapon
5	When the player fires your sniper rifle, they should be presented with a submenu allowing them to choose a target. Once they have selected a target, they have the option of shooting straight away or spending a round aiming. Spending time aiming improves the result as follows: • No aim: 75% chance to hit, standard damage • One round aiming: 90% chance to hit, double damage • Two rounds aiming: 100% chance to hit, instakill
6	If the shooter takes any action other than aiming or firing during this process, their concentration is broken and they lose their target. They will also lose their target if they take any damage.
7	Create ammunition boxes

Feature 1

Create a new class named Shotgun which extends weapon. Player would be able to pick up and use this weapon.

Feature 2

Create a new class named FireShotgunAction which would enable a player to fire with the shotgun. If the player chooses to fire the shotgun, the possible locations to fire would be determined and printed in the form of a submenu.
Creating a new class for this action allows us abide by the Single Responsibility Principle in order to ensure that a single class is responsible for everything regarding shooting with a shotgun. The current AttackAction randomly selects a weapon to use when attacking, hence will not be suitable for the use of shotgun as a weapon.

Feature 3

If an actor is within the range of the shotgun, then there will be a randomly generated number between 0-99. If the number is below 75 then the target actor will be damaged.

Feature 4

Create a new class named SniperRifle which extends weapon. Player would be able to pick up and use this weapon.

Feature 5

Create a new action called UseSniperRifleAction which would enable player to use the sniper rifle. A new attribute named concentration with type integer to store the number of turns a player has spent aiming would be initialised to 0 in this class. Furthermore, an attribute named target would be created to store the player's target. These two attributes would be passed in by the constructor of UseSniperRifleAction to set the target and initialise to 0.
A getNextAction method would be added to UseSniperRifleAction class. This would increment concentration and then return the action itself. This would allow us to create this multiturn action.
An attribute named concentration would also be stored in the Player class and the execute method in UseSniperRifleAction would return the value of concentration to update it.

The execute method in FireSniperRifleAction creates a submenu for the player to select who to target if one has not been selected already.
Next, another submenu would appear for the player to decide whether to shoot or aim.

Creating a new class for this action allows us abide by the Single Responsibility Principle in order to ensure that a single class is responsible for everything regarding shooting with a sniper rifle. The current AttackAction randomly selects a weapon to use when attacking, hence will not be suitable for the use of a sniper rifle as a weapon.

Feature 6

If the LastAction was not FireSniperRifleAction then it sets the value of concentration to 0.
An override hurt method in player is created which sets value of concentration to 0 as well.
If the value of concentration is 0 then a new instance of FireSniperRifleAction is added to the player menu.

Feature 7

A new class named AmmunitionBox which extends PortableItem. If an item is picked up, then the player will gain more ammunition.
An attribute called Ammo would be created in the Player class to keep track.

Mambo Marie

Feature Number	Feature Description
1	Mambo Marie is a Voodoo priestess and the source of the local zombie epidemic. If she is not currently on the map, she has a 5% chance per turn of appearing.
2	She starts at the edge of the map and wanders randomly
3	Every 10 turns, she will stop and spend a turn chanting. This will cause five new zombies to appear in random locations on the map. If she is not killed, she will vanish after 30 turns.
4	Mambo Marie will keep coming back until she is killed.

Feature 1

Make a new class called MamboMarie which extends ZombieActor. This class will have all the functionality related to MamboMarie. We can generate a random integer from 1 to 100 and if she is not on the map, we can place her object.

Feature 2

We can make a list with possible edge locations.If the above condition meets, we can place MamboMarie's object anywhere in the possible location. To make it wander randomly, we can associate the wanderBehavoir with MamboMarie.

Feature 3

Make a new attribute which is updated by overriding the tick location.If the number of turns reaches 10, we can choose 5 locations using Math.random and place Zombie objects in them.At 30 turns, if she is conscious then we can remove her object from the map.

Feature 4
Using the condition at feature 1 and given the fact she is conscious, we can place her object anywhere in the edge location of the map.

Ending the game

Feature Number	Feature Description
1	A “quit game” option in the menu
2	A “player loses” ending for when the player is killed, or all the other humans in the compound are killed
3	A “player wins” ending for when the zombies and Mambo Marie have been wiped out and the compound is safe

Feature 1

Create a new class called ExitAction with a hotkey 6.It is added to list Player's actions and whenever it clicks on ‘6’, the game is finished.

Feature 2

In Player's playTurn method, we can the maximum values of x and y values of map with the getXRange and getYRange in GameMap class.Using that we can check if there are instances of Human and Player in compound map.If not then we show “Player lost” and the game ends.

Feature 3

In Player’s playTurn method, we can the maximum values of x and y values of map with the getXRange and getYRange in GameMap class.Using that we can check if there are instances of Zombie and Mambo Marie in compound map.If not then we show “Player won” and the game ends.