**Design Rationale**

**Group : Team 2099**

**Members : Siang Jo Yee, Lua Shi Liang**

Rising from the dead

**Corpse** [New Class]

* An **Item** added to the **GameMap** when an **Actor** (**Human**, **Farmer** and **Zombie**) is killed.

New Changes and their Responsibilities:

* This class inherits from an abstract class called **Item**.
* It has two private attributes (**int** turn and **Actor** initialActor).
  + turn - acts as a counter
  + initialActor - the dead **Actor** who owns the **Corpse**
* It has a constructor that takes in 2 parameters (**String** name and **Actor** initialActor). It initialises initialActor and also makes use of the superclass constructor to initialise name, displayChar and portable.
* It overrides tick() method to fulfil the required functionality for the **Corpse** to “rise from the dead”
  + Parameter: **Location** currentLocation - Current location where the corpse is placed
  + This method is executed every turn in the game.
  + turn increments by 1.
  + If initialActor does not hasCapability(*ZombieCapability.UNDEAD*), in which it is not a **Zombie**, generate a random integer between 5 to 10 inclusive.
  + Then, if the turn equals the random integer and there is no **Actor** on the currentLocation, A new **Zombie** object is instantiated.
  + The **Zombie** is added onto currentLocation and the **Corpse** is removed from currentLocation using methods in **Location** called addActor() and removeItem() respectively.
  + A string is printed to notify the **Player** that a dead **Human** becomes a **Zombie**.
  + This will look as though the **Corpse** becomes a **Zombie** 5 to 10 turns later.

Design Choices / Reasons:

* The design in which the Corpse inherits from Item takes into account the Do Not Repeat Yourself principle because Corpse also uses the features in Item and inheritance helps to avoid repetition of codes.
* **Zombie** objectis instantiated in the method as a local variable instead of as an attribute to follow Declare things in the tightest possible scope principle to lessen the risk that something can depend on it, thus reduce the risk of possible failure in the future.

**AttackAction** [Existing Class]

New Changes and their Responsibilities:

* execute() method is modified
  + If an **Actor** is killed after being attacked [existing code], a new **Corpse** object is instantiated [modified] instead of a **PortableItem** [existing code], then added to the **Location** of the dead **Actor** which is removed from the **Location** [existing code].
  + This is because there is a new **Corpse** class created in which it is able to increment the counter every turn in the game to control the turn when it becomes a **Zombie** to fulfil the required functionality.

Design Choices / Reasons:

* **Corpse** is instantiated in the method as a local variable instead of as an attribute because Declare things in the tightest possible scope principle is taken into account in this design.