# Design Changes

## Zombie Attacks

1. Instead of integrating the bite attack into **AttackAction**, a **BiteAttack** class was created instead. The bite attack requires a different method to retrieve the **Weapon** to execute a bite attack and it also has a different probability to miss compared to a normal attack. This disrupts the flow of the code through many conditional checks to check if the **actor** is a **Zombie** and whether **weapon** is for a bite attack would make the code messy and hard to follow. It would also be difficult to integrate other types of attacks (if needed) if this design is followed again.

At first, it was decided to integrate the bite attack into **AttackAction** to avoid duplicated code that would be in the **execute(..)** method of **AttackAction** and **BiteAttack**. This would exist if **BiteAttack** was created to inherit **AttackAction** without any refactoring. In the new design, this is avoided by refactoring common blocks of code in the **AttackAction** **execute(...)** method into package-private methods. **BiteAttack** inherits these methods and uses them in its own execute method which avoids duplicated code and follows the DRY principle.

1. Since the bite attack is now in the **BiteAttack** class, the bite attack health restore is now in **BiteAttack.**
2. **DropAdjacentItemAction** was changed to inherit **DropAdjacentItemAction** instead of **Action** as it will contain the same attribute and most methods. The **execute(...)** method is different and is overridden to drop the item in an adjacent cell, rather than at the **actor’s** location. Since of the change in inheritance, **DropAdjacentItemAction** will now only have a minimum and maximum of 1 **ZombieLimb**.
3. **AttackAction** is no longer designed to be responsible for dropping zombie limbs when they are knocked off. This is because when a **Zombie** takes damage, it may have its limbs knocked off which modifies attributes in the **Zombie** class. To follow the “Classes should be responsible for their own properties”, the loss of limbs should be handled in the **Zombie** class for encapsulation and avoids down casting.
4. It was originally thought that a **Zombie** speaking wouldn’t turn up a turn, however this is incorrect, and it should take up a turn. So, a **SpeechBehaviour** and **SpeakAction** class have been added to fit the requirement. The **SpeechBehaviour** is the second behaviour in the **Zombie** behaviour list. This is because a **Zombie** is required to pick a weapon at the start of its turn if it is available which is checked in the **ScanvengeBehaviour**. Then there is a 10% chance that the **Zombie** will speak before checking for anything other possible actions.