# Changes to Assignment 2 Implementation

* Added **CropCapability** enum to be used in **Crop** and **Farmer** to identify if **Crop** is ripe or unripe. So, there is no need to make assumptions if the crop is ripe or unripe based on the size of **allowableAction** on **Crop**.
* Instead of **CraftWeaponAction** being responsible of creating the upgraded weapon from **ZombieLimb**, the subclasses of **ZombieLimb** are responsible. This is done to follow the Open/Closed Principle. Now **CraftWeaponAction** is open for extension by introducing new upgradable items without modifying it. To achieve this, an **upgrade()** method is added to the **ItemInterface** which returns an upgraded form of the item if it has one, else, it’ll return null.
* Originally, the name of the **ZombieLimb** is used to identify whether the limb is an arm or leg. This is a Connascence of Name (CoN) that an IDE cannot pick up and can lead to a bug that is hard to identify if the name of the **ZombieLimb** is changed. To avoid this, **ZombieLimb** is changed to an abstract class and has two subclasses – **ZombieArm** and **ZombieLeg**. This still has a CoN but it works to our benefit as an IDE can easily identify the bug.
* Removed use of **instanceof** as it is a code smell which restricts polymorphism and reduces code extendibility. It is replaced with the use of **Capabilities** to follow The Open/Closed Principle.
  + Added **EatCapability** enum to identify whether an **Item** can be eaten, rather than checking if the **Item** is an instance of **Food**.
  + Added **GroundCapability** enum to identify whether a **Ground** is sowable, rather than checking if the **Ground** is an instance of **Dirt**.
  + Added **ActorCapability** enum to identify whether an **Actor** drops the crop or pockets it after harvest.
* Refactored some code out of **Human** and **Farmer** so they only have one responsibility, and that is to store the behaviours they exhibit. The responsibilities refactored out are:
  + Eat food from inventory
  + Pick up food on the ground at their location
  + Fertilize unripe crop (Farmer only)
  + Sow crop (Farmer only)
  + Harvest crop (Farmer only)

A class that implements **Behaviour** is created for each responsibility and is appropriately added to their **Behaviour** attribute which is now an array **Behaviour[]**, to hold several behaviours. This now follows the Single Responsibility Principle which makes **Human** and **Farmer** easier to maintain and extend.

* Separated movement related **Behaviour** (ie. **HuntBehaviour**, **WanderBehaviour**) into a different array which will be iterated through after the non-movement **Behaviour** array. This is done to make the code more readable and the **Behaviour** object doesn’t have to be checked if it is a movement behaviour at the start of each iteration.
* Duplicated code that determines a valid location for an **Item** to be dropped has been refactored out of **HarvestAction** and **DropAdjacentItemAction** and made into its own class. This was done to follow the DRY principle.
* The **Food** class was changed to an abstract class so more types of food can be added into the system easily. A **Spinach** class is added and is now created in **HarvestAction** instead of **Food**.
* Encapsulated tightly coupled classes into game.eat package and game.farming package. game.eat contains classes that work together for the process of eating food. game.farm contains classes related to farming, included fertilizing, harvesting and sowing.