Changes in Design (A3)

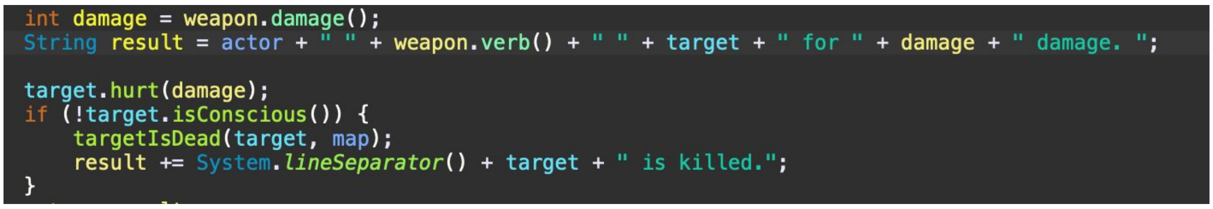
***This documentation only shows the changes in design and the reasons behind. For complete updated documentations, please see "Design Rationale".*

IdentityCapability

We found out there are many occasions where we need to take the type of an actor into account before returning a relevant action or behaviour. Previously, we used the keyword "instanceof" to overcome the issue but we quickly realized that it was not the best practice recommended. Therefore, we decided to create an enum class that has all types of actors called IdentityCapability which has ZOMBIE, HUMAN, FARMER, PLAYER and MAMBOMARIE. Each constant denotes a type of character in the game. Now that each character has an IdentityCapability in their own capabilities, it eliminates the usage of "instanceof" and increases the readability of the code.

AttackAction

In the previous two assignments, we have split the load of AttackAction into two subclasses, AttackZombieAction and AttackHumanAction, as we reviewed our code, we found out by doing so, it created duplicated code and made the reader harder to trace the action. Thus, we have removed both AttackZombieAction and AttackHumanAction and put both functionalities back to the AttackAction class so all all attack action is done within the class. By doing so we have achieved two major things: Firstly, encapsulation, all the information involving the action of attacking another actor, be it zombie attacking humans or the other way round, is hidden within the class. Secondly, we can keep the code DRY as the code duplication is reduced. For instance:



```
int damage = weapon.damage();
String result = actor + " " + weapon.verb() + " " + target + " for " + damage + " damage. ";

target.hurt(damage);
if (!target.isConscious()) {
    targetIsDead(target, map);
    result += System.lineSeparator() + target + " is killed.";
}
```

This code has appeared twice in AttackZombieAction and once in AttackHumanAction, now that both actions are gathered in AttackAction, we only need to implement this snippet of code once.

Since AttackZombieAction and AttackHumanAction is put together in AttackAction, we need to modify the class that originally has dependencies and associations with it. For example: AttackBehaviour, the getAction() of AttackBehaviour no longer has to check the type of actor before returning a relevant attack action, as returning AttackAction alone is sufficient.

Limb

The name of Limb class is changed to UpgradableWeapon as it fits better into the system that acts as a base class for all the weapons that can be upgraded into a superior weapon. It will have an abstract craftWeapon method that will return an upgraded weapon to that weapon instance just like the Limb did. In case we want to add other kinds of weapons that can be crafted into a better weapon in the future, UpgradableWeapon will make it clearer for the readers and programmers to understand the base class for this functionality already exists. On the other hand, by naming it Limb like we did before has limited what this class is capable of.