FIT2099 Assignment 1 Star War: Responsibilities of Classes in UML

Loi Chai Lam (2813 6179)
Vinitha Raj Rajagopal Muthu (2824 9542)

1. Leave Affordance
   For Leave Affordance, I create a new class Leave under the package actions. The role of the Leave class is allowing an actor to put down the object they are currently carrying. After leaving, the actor should be holding nothing and the item should be in the location of the actor when it was done. I use the method in EntityManager (also in SWWorld), SWWorld.find(SWEntityInterface) to get the location of the actor, then I use the function in SWActor, SWActor.getItemCarried to get the item carried by the actor. After that, I will use the function in EntityManager, EntityManager.setLocation(SWEntityInterface,SWLocation) to set the item carried in the location of same actor. In SWActor, I use SWActor.setItemCarried = null, to point out that the actor is holding nothing.

2. Force ability
   For Force ability, I create a new class TheForce under starwars package. I create an object of TheForce in SWActor, to show that every actor has Force. In class TheForce, there are 2 attributes and 5 methods. These attributes are forceAbility (Boolean), forceNumber(int). The forceAbility for every actor is to verify whether the actor has the ability to use the Force, the forceNumber is the force level of that actor. The methods in class TheForce are getForceNumber(), which get the force number of the actor; setForceNumber(int), which set the force number of the actor; setForceAbility(boolean), which set the ability of the actor to use the force; addForceNumber(int), which increase the for number of that actor; checkForceAbility(), which check the ablity of the actor to use the force. This implementation shows that every actor has Force, but only some of them have the ability to use the Force.

3. Lightsabres
   For Lightsabres, in the class Capability, I create an enum, FORCEABLE to show that the Lightsabres is a weapon that need Force (so that if there is another weapon need Force, I just put it as FORCEABLE instead of WEAPON). In the class LightSaber, I will create an static final attribute, forceNeeded:int, to record the force needed for the actor to wield the Lightsabres. Also, I will change the capabily of Lightsabres to FORCEABLE instead of WEAPON. In the class Attack, I will modified the function act by adding the conditions which show that only the Actor who has force ability && the the item carried is a

FORCEABLE item && the force number of the actor is larger than the force needed in the FORCEABLE item can wield the Lightsabre by using the methods getItemCarried() in SWActor; hasForceAbility() in TheForce; hasCapability(Capability) in SWEntity and attribute forceNeeded in LightSaber.

4. Ben Kenobi
   For Ben Kenobi, I will create a new method train(SWActor) under the class SWLegend to show that only Legend can train actors. In method train(SWActor), I will check the location of the Legend and Actor using method find(SWEntityInterface) in SWWorld. If their location are same, I will add the actor's force number using the method addForceNumber(int) in class TheForce. In class BenKnobi, I will also create a method setStudent(SWActor), to set the student of Ben Kenobi. This will show that BenKenobi only can increase its student's force number. To pass the object Actor (student) into the class BenKenobi, I create an attribute student.

5. Droids
   A new method called isOwner is created under SWActor to specify if actor is owner. Droid class is introduced which inherits from SWActor and has extra methods to set, get owner and move in a random direction. SWActor is also connected to SWWorld so that the position of the droid and owner can be known so that the Droid can move accordingly using the move method in the Move class.