**Force**

We have implemented *Force* as an integer value in *SWActor*. This is because there may be other characters that can use the *Force*, not limited to just *Player* alone. One such example is *BenKenobi*.

**Leave**

We replaced the **getActorLocation()** method in the original design with some equivalent code implemented inside the **act()** method, which invokes the **whereIs()** method to determine where the entity target should be placed.

**Train**

We have decided to implement *Train* as a *SWAction*. By default, the *Luke* instance created in *SWWorld* will be initialized with *Train* added as an *Affordance* (this is because *SWAction* and *SWAffordance* both implements *SWActionInterface*, so they have a relation). However, the *Train* option will only be available in the text interface menu if *BenKenobi* happens to be in the same location as *Player*. *Train* will also decide if, after undergoing training, *Player* has sufficient *Force* ability to wield a *LightSaber*.

**LightSaber**

We removed the default **WEAPON** capability of a *LightSaber* that is initialized by its constructor. In its place, we have added a method, **canUseAsWeapon()**, that checks if the *SWActor* holding the *LightSaber* has sufficient force to wield it as a weapon. If they do, add a **WEAPON** capability to the *LightSaber*. This method will also be called in the constructor, as *BenKenobi* also possesses a *LightSaber*.

**Droid**

We chose to use *SWActor* rather than *SWEntity* as the superclass of the *Droid*. This is because a *Droid* has *hitpoints* attribute as do other *SWActors*, and it needs to move while time passes. Besides that, the inherited methods **canDo()**, **act()**, and **getDescription()** are sufficient to handle the *Droid* logic.