The philosophy behind our design is primarily to implement the code so that it is conscience, encapsulated and easy to update. Within the project, we have utilised the classes to ensure encapsulation and added new classes such as Control, Train, Humanoids, Fix and Droids to allow the tasks required to be completed. Although some aspects were not able to be implemented through refactoring, areas of improvement are shown upon the deadline of the project.

SWActor Class:

In this class methods relating to health, force training, and force ability were added. This is since all actors’ entities require a base health, and using this class, we can use getters and setters to retrieve or add health quickly and efficiently.

Fix Class:

This class is new and added in the starwars.action package to allow the droids and Luke to fix immobile droids. This is associated with the Affordance methods, which causes all entity’s associated with the Fix class, are able to repair immobile droids.

SWWorld Class:

This class includes all the entities initialised including, Aunt Beru, Uncle Owen, Canteen along Ben’s path, R2D2, C3PO , more Tusken Raiders and immobile Droids. As all the entities are generated in this class, allows simpleness and conscience as we further update our code for Assignment 3.

Droid Class:

Initially our design includes attributes such as health and droid parts, however as we started implementing, we came to a conclusion that it was better leaving the health in the SWActor Class and only focus on validation for 3 different situations. The droid class also has attributes such that, it use the take and fix affordance classes.

1. If the droid name was R2D2, it would have specific patrol routes as well as being able to repair
2. If the droid name was C3PO, it would be stationary and have a 10% chance of displaying a message calling for help.
3. If the droid health is less than 0, it will be called a droid part.

In fact, Droids can become owners of other droids.

Humanoid Class:

The humanoid class was created to accommodate for Uncle Owen and Aunt Beru, instead of creating classes for each individual character since their abilities are the same. Their purpose is to provide testing of Luke’s mind control once he is fully trained by Ben. Implementation of the force was incomplete, we still need to add manipulation of the move command when mind control is selected. This section can continue to be updated as we progress through the third assignment.

Train Class:

The train class allows Luke to receive training from Ben, this action is only allowed to occur if both characters occupy the same position. Once trained, Luke can force players to move in a certain direction, this part is undergoing maintenance to properly display this option on the menu once fully trained status is achieved. Training accounts for a move, meaning it’s methods are in the entity interface

1. Luke starts on 0 training points and receives 25 training points upon training
2. Once his training points reaches 100, his level of control is raised from 20 to 80, enabling him to use the force to manipulate the moves of other objects

Control Class:

Once fully trained, the player can select to use his move by controlling the subsequent direction of characters on opposing teams. This is done by utilising the move class and setting the direction of intended travel to one chosen by the user. In the case of Tusken Raiders, they only move on half of the available turns, this feature will force them to have a move in the chosen direction