Design Rationale

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 is 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.

Design Rationale

UML Documentation Classes

Within the UML diagram, it consists of 4 main classes that will influence the gameplay of the Star Wars assignment, “Force”, “Ben Kenobi”, “Character”, “Droids” and “Lightsabres”. These classes are chosen due to the reasons below.

Class: Force

In this game’s objective, it revolves around character position and advantageous manoeuvre, relying on the “Force” as the main source of power and advantage gain. Therefore, the Force class is affiliated with all the other classes, which indicate whether they are force sensitive (can use the force) or not. The main force attribute is public as all objects are associated with the force and a non-force user can suddenly become a force user. The attributes weakForce and strongForce will be set private as they indicate a specific character’s power.

Class: Ben Kenobi

Although there is probably one instance of this class, it is important to know that Ben will play a bigger role in the game. The code provided in the documents, prove to be able to move and interact with other objects. Ben also adds Force to a character, such as Luke.

Class: Character

This class allows enemies, heroes and Tuskans to be created. All characters having attributes such as health, currentForce and droidPartsCollected, cause them to interact with the objects in the game. The Behaviour is listed as a private attribute, due to the reasons that a character is unlikely to switch sides in much of the game. However, if necessary, a set method will be needed to change the behaviour of a character.

Class: Lightsabres

This Lightsabre class contains only one attribute, “Weapon” as only characters that are force sensitive will be able to use the lightsabre as a weapon. As there will be many lightsabres, using a class for this weapon will create instances efficiently and effectively.
Of all the classes, only the characters and Ben Kenobi are inherited with the Force class, because again, the Force dictates what a character can and cannot do. The Characters are associated with the Lightsabres due to the ability to pick them up and with the droids, as each droid requires an owner.