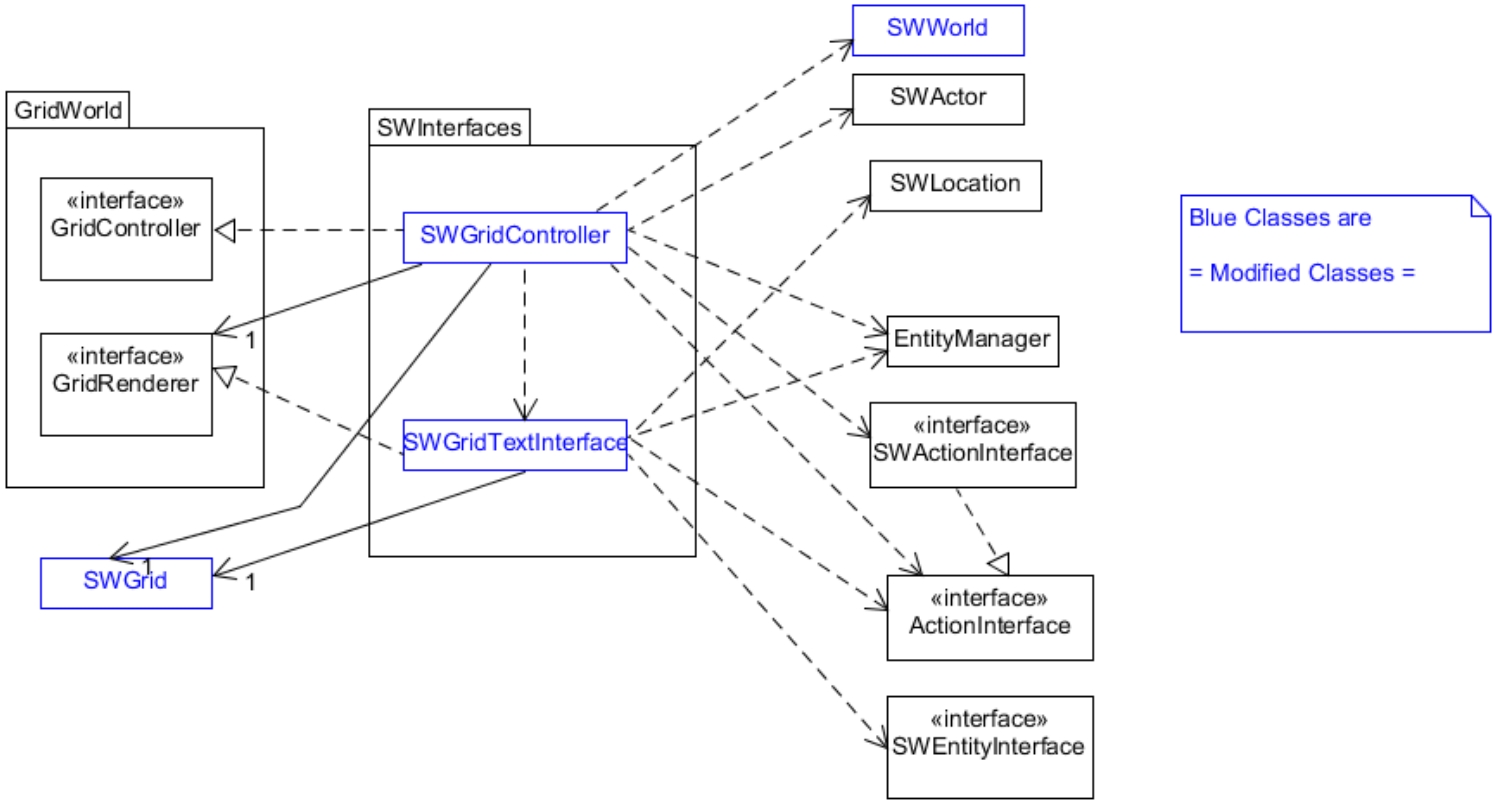
Multiple_Worlds –Design



UML Class Diagram For Interface Changes for Multiple Worlds

To allow for interstellar travel, multiple “worlds” had to be created.

This implementation was handled with changes to four existing classes:

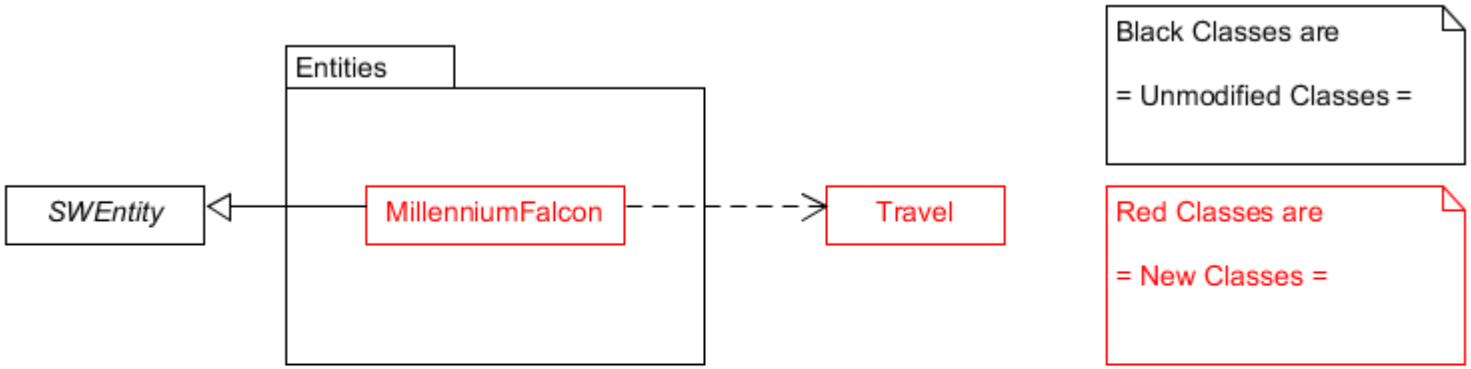
1. SWWorld
2. SWGrid
3. SWGridTextInterface
4. SWGridController

- SWGrid’s constructor was changed to be able to create grids of different sizes.

- SWWorld was modified to store multiple SWGrids that represent the multiple different worlds. Initialisation of worlds was also slightly changed so that different worlds (Tatooine, RebelHQ, DeathStar) are initialized separately on different SWGrids.

- SWGridTextInterface and SWGridController were changed to handle the current SWGrid (current world) that the player is on instead of one static world. The current world is obtained by calling a method from SWWorld

Millennium_Falcon_Entity-Design



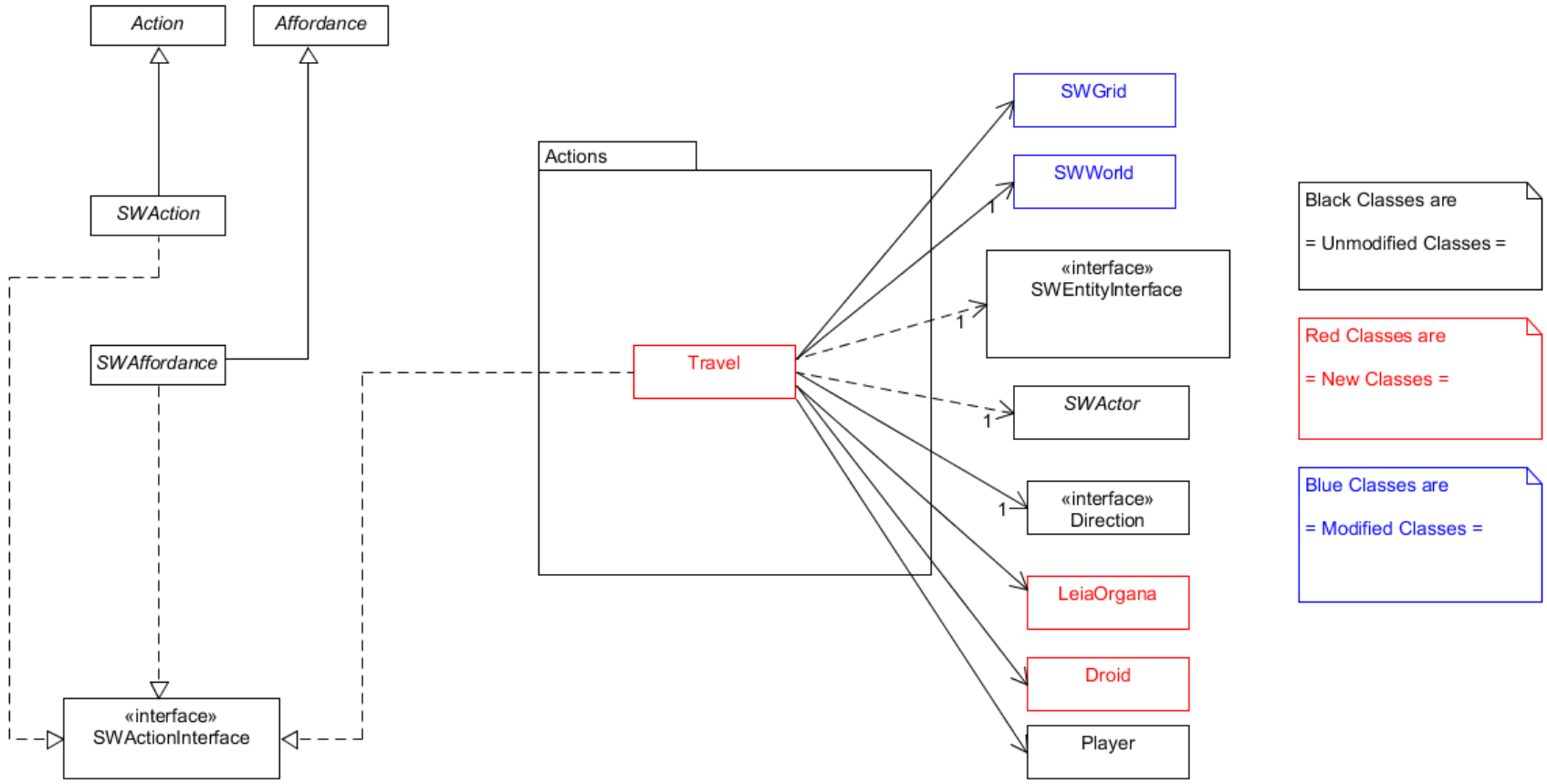
UML Class Diagram For Interstellar Travel - Millennium Falcon Entity

A new MillenniumFalcon entity is included in the game to allow for interstellar travel. Each different “world” contains a MilleniumFalcon.

The MilleniumFalcon has the travel affordance which allows the player to perform the action of travelling to different worlds.

We decided to implement a singular MillenniumFalcon Entity instead of multiple to prevent having to create multiple new classes and when new worlds are introduced and also to avoid having to alter multiple classes.

Travel_Action-Design



UML Class Diagram For Travel Action

A new Travel affordance is created to allow for interstellar travel. This affordance is added to the MillenniumFalcon Entity.

As it deals with travelling between worlds, the Travel affordance has interactions with many different classes. These are explained as follows:

- SWGrid & SWWorld are needed in order to identify which SWGrid, and this, which location to move the player and all following party members to.
- SWEntityInterface is needed to locate the SWActor and all following party members.
- The Direction Interface is needed to identify entities that are on neighbouring locations to the player.
- LeiaOrgana, Droid and Player classes are needed to determine if an Entities are instances of these classes to know if they should be transported to a new location.

We have decided to implement only a single Travel action instead of multiple Travel actions for similar reasons to implementing only a single MillenniumFalcon Entity, which is to avoid the need for creating new Travel classes when new worlds are introduced and modifying multiple Travel classes.