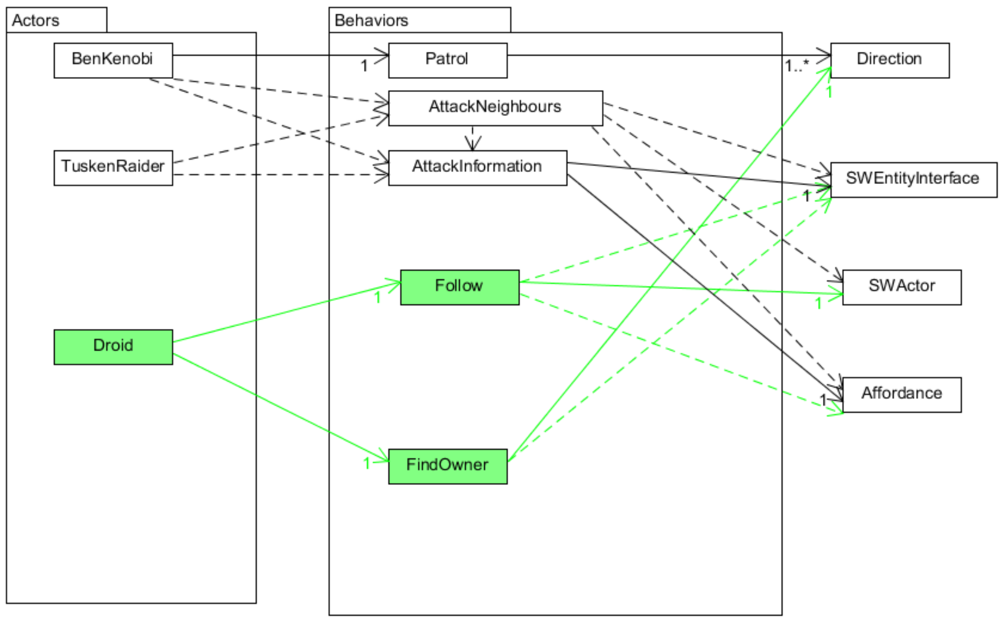
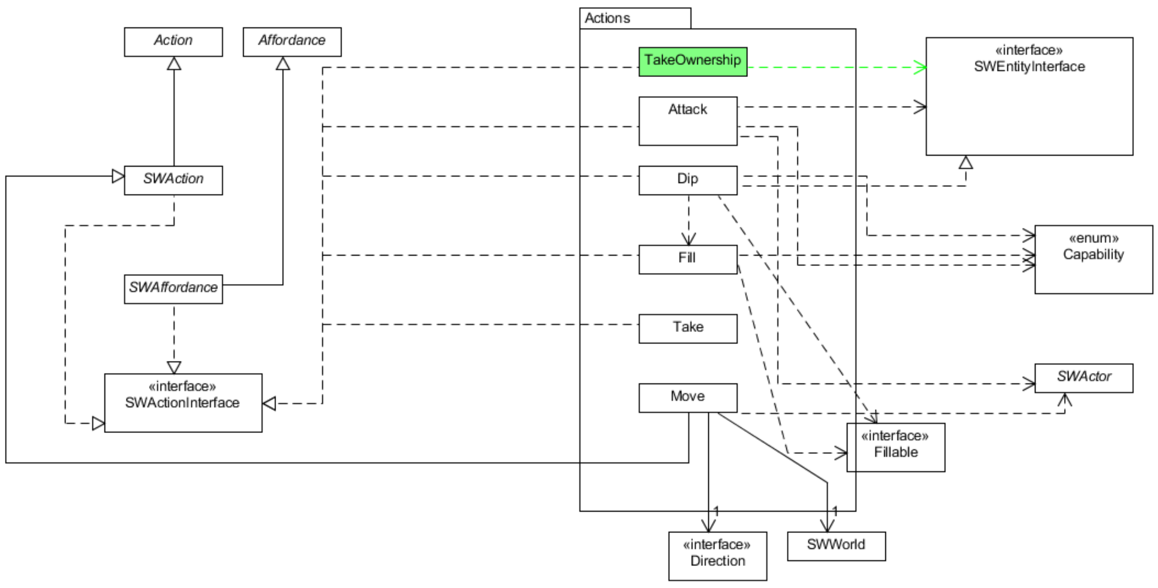
Droid Rationale

Main UMLs





What classes exist in your extended system?

The classes introduced in the extended system will be **Droid**, **Follow**, **TakeOwnership** and **FindOwner**.

What is role and responsibility of each new class?

The role of **Droid** is to encompass the detailed functionalities of the new SWActor.

The role of **Follow** is to allow a droid to be “in the same location as its owner. If the droid’s owner is in a neighbouring location, it will move to that location.”.

Following the occurrence of **Follow**,

- the **Droid** will be “in the same location as its owner”

The role of **TakeOwnership** is to have a **SWActor** take ownership of a **Droid**.

Following the occurrence of **TakeOwnership**,

- **Droid** will exhibit behaviours associated with having an owner
- The **SWActor** will own a **Droid**, perhaps altering an affordance, TBA

The role of **FindOwner** is to have the **Droid** “pick a direction at random and move in that direction. It will keep moving in that direction until it finds its owner, or can no longer move in that direction. If it can no longer move in its current direction, it will pick a new direction at random and move in that direction”.

Following the occurrence of **FindOwner**,

- “ the **Droid** will either have found its owner and then begin the **Follow** behaviour or
- The **Droid** will continue with the **FindOwner** behaviour”

How the new classes relate to and interact with the existing system.

Droid is simply an extension of SWActor, so as not to repeat code. **Droid** will interact with two new behaviours; **FindOwner** and **Follow**, otherwise remaining stationary. All of these functionalities should be covered/called in the **Droid** act() method.

How the (existing and new) classes will interact to deliver the required functionality.

- Droids can’t use the Force.	This is dependent on the design of Force Ability.
- A droid can have another actor as its owner - if it is in the same location as its owner, it will stay there - If the droid’s owner is in a neighbouring location, it will move to that location	Done using setLocation() on the item wherever the SWActor is. If the owner is moving, **Follow** will interact with **Move** by passing the new direction into the scheduler. This should ensure that if the Droid’s owner is in a neighbouring location, the Droid’s next movement will already be scheduled.
- if it cannot find its owner in its current or neighbouring locations, it will pick a direction at random and move in that direction - If it can no longer move in its current direction, it will pick a new direction at random and move in that direction	Not exactly sure when this would happen, so I might change my approach if a problem is introduced. Otherwise, this functionality will be covered in **FindOwner**, interacting with **Move** and probably **Follow**.
- If a droid has no owner, it will not move	Will be covered in the **Droid** class
- Droids lose health when they move in Badlands. - Droids don’t die, but they become immobile when their health runs out.	Will be covered in the **Droid** class (roughly(!) As below) If in location w symbol b {takeDamage} If(isDead()){ immobile }