# Updates

None yet.

# Reservoir Rationale

### Main UMLs

*What classes exist in your extended system?*

There will be not further additions to the extended system. The current reservoir will have an additional method takeDamage() in which the conditions of assignment 3 will be met.

*What is role and responsibility of each new class?*

N/A

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

The new additions to the current reservoir class will interact with locations simulator.matter.Entity and starwars.SWEntity for use of their convenient methods as described below.

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

|  |  |
| --- | --- |
| - all reservoir objects start with 40 hitpoints | Since reservoir is already an entity, we can easily give it hitpoints using setHitpoints() |
| - If the hitpoints of a reservoir fall below 20, its short description must change to “a damaged water reservoir”,  - its long description must change to “a damaged water reservoir, leaking slowly”,  - its symbol must change to V.  - If the hitpoints of a reservoir fall to zero or below, its short description must change to “the wreckage of a water reservoir”  - its long description must change to “the wreckage of a water reservoir, surrounded by slightly damp soil”,  - its symbol must change to X. | These conditions can be implemented in a takeDamage(int) method. As well as have the reservoir take damage, we can check the condition of the hitpoints and if necessary change;  Descriptions, for which we can draw on simulator.matter.Entity.setShort/LongDescription(str)  And for map symbols;  starwars.SWEntity.setSymbol(str) |