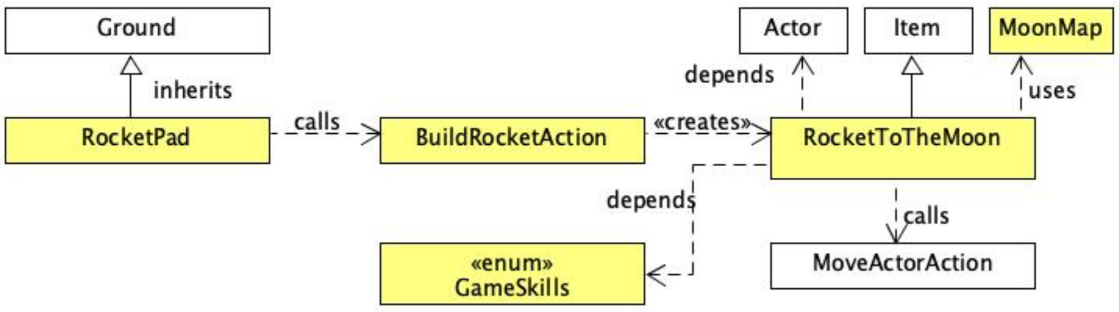
Design document
Going to the Moon
The newly added classes are MoonMap, EarthMap, RocketToTheMoon, RocketToEarth.



Based on the **class diagram** above, RocketPad inherits from Ground. In the RocketPad class's allowableActions method, it calls the BuildRocketAction class if the rocket body and rocket engine are placed on the rocket pad. The BuildRocketAction class creates a rocket by calling the RocketToTheMoon class.

The RocketToTheMoon class inherits from Item and is a furniture on the map. In order for the player to fly to the Moon map, the RocketToTheMoon class checks if the player standing on it has the spacesuit and oxygen tank in its inventory. The spacesuit has GameSkills.SPACETRAVELLER and the oxygen tank has GameSkills.OXYGENTANK. If the player has these two items in their inventory, the getAllowableActions method adds the MoveActorAction into actions. The MoveActorAction uses the MoonMap class's getMap method to get the Moon map for the player to move to and also uses the MoonMap's static variables ROCKET_X and ROCKET_Y to give the impression that the player is on the rocket's location in the Moon.

To allow the player to fly back to Earth, a RocketToEarth class has been instantiated on the Moon map. The MoveActorAction uses the EarthMap class' getMap method to get the Earth map for the player to move to and also uses the EarthMap static variables ROCKET_X and ROCKET_Y to give the impression that the player is on the rocket's location on Earth.
Class MoonMap
```
public class MoonMap
```

The MoonMap class represents the Moon map in the game. It stores the List of Strings of the Moon map and creates a GameMap of the Moon. It also has public final variables that represent the x and y coordinates of the rocket on the Moon. It has a static getMap method to get the current instance of the GameMap of the Moon.

Class EarthMap
```
public class EarthMap
```

The EarthMap class represents the Earth map in the game. It stores the List of Strings of the Earth map and creates a GameMap of the Earth. It also has public final variables that represent the x and y coordinates of the rocket on Earth. It has a static getMap method to get the current instance of the GameMap of the Earth.

Class RocketToTheMoon
```
public class RocketToTheMoon extends Item
```

The RocketToTheMoon class inherits from Item. It represents a furniture item on the map. It is responsible for transporting the player from its current location to the Moon.

How it works:
In the overridden getAllowableActions method, it checks if the player has the spacesuit and the oxygen tank in its inventory by calling its own class's checkSpacesuit and checkOxygenTank

methods. If both are true, the MoveActorAction is added into Actions for the player to move to the Moon's map. To give the impression that the player is on the rocket's location in the Moon, it uses the MoonMap's static variables ROCKET_X and ROCKET_Y when using the MoveActorAction.

Since the Item spaceSuit has GameSkills.SPACETRAVELLER and the OxygenTank item has GameSkills.OXYGENTANK, the checkSpacesuit checks if the player's inventory has an Item with GameSkills.SPACETRAVELLER and checkOxygenTank methods checks if the player's inventory has an Item with GameSkills.OXYGENTANK.

<u>ClassRocketToEarth</u>

```
public class RocketToEarth extends Item
```

The RocketToEarth class inherits from Item. It represents a furniture item on the map. It is responsible for transporting the player from its current location to Earth.

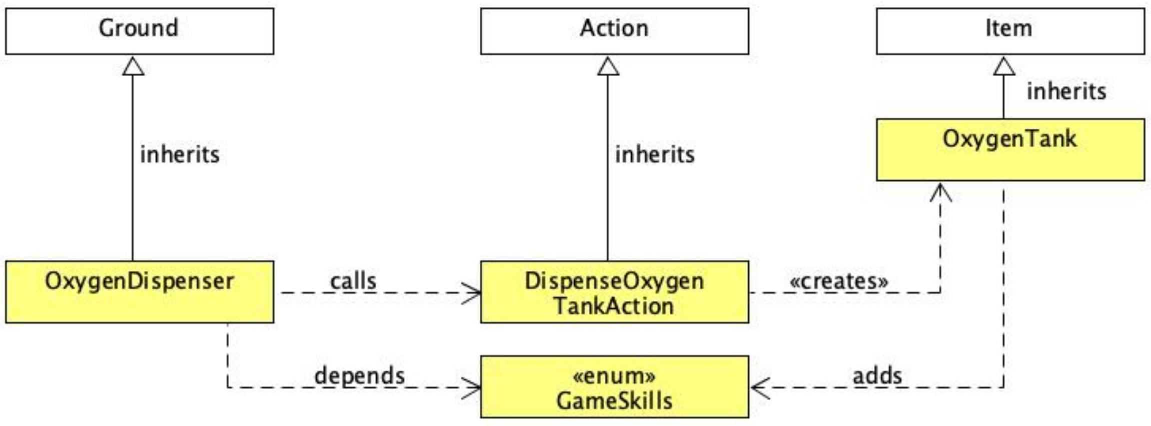
How it works:
In the overridden getAllowableActions method, the MoveActorAction is added into Actions for the player to move to the Earth's map. To give the impression that the player is on the rocket's location in the Moon, it uses the EarthMap's static variables ROCKET_X and ROCKET_Y when using the MoveActorAction.

Spacesuit and Oxygen
The newly added classes are OxygenDispenser, OxygenTank, DispenseOxygenTankAction.

This section will be broken down into 3 parts.

Part 1: Adding a spacesuit item to the Earth lair map
Class Application
Item spaceSuit has been added in the EarthMap class's GameMap. The spaceSuit has skills GameSkills.SPACETRAVELLER.

Part 2: Adding an oxygen dispenser which has a button that, when pressed, causes the dispenser to produce an oxygen tank in its location on the **next turn**. The button does not work while the dispenser is producing the tank, or while there is an oxygen tank in the location.



Based on the **class diagram** above, the OxygenDispenser class inherits from Ground. It depends on the enum GameSkills. If there is no OxygenTank object which has GameSkills.OXYGENTANK on the oxygen dispenser, it adds a newly instantiated DispenseOxygenTankAction class into a List of Actions that can be performed by the Actor. The DispenseOxygenTankAction creates a newly instantiated OxygenTank and adds it to the OxygenDispenser's location. The OxygenTank class inherits from Item and adds the skill GameSkills.OXYGENTANK.

How did we perceive this question and what is our proposed solution?
From our understanding from the question, the turns taken to dispense the oxygen tank are as follows:
Turn 1: The player is adjacent to the oxygen dispenser and presses the oxygen dispenser's button to dispense the oxygen tank.
Turn 2: The oxygen tank is dispensed on the location of the oxygen dispenser.
Turn 3: The player moves to the location of the oxygen dispenser and picks up the oxygen tank.

Proposed solution:
In turn 1, when there is no oxygen tank on the location of the oxygen dispenser (which is an object of the OxygenDispenser class), the allowableActions method will add the DispenseOxygenTankAction into a List of Actions that can be performed by the player. If the player chooses the option to press the button to dispense an oxygen tank, in turn 2, the execute method in the DispenseOxygenTankAction class adds an oxygen tank (which is an object of the OxygenTank class) on the map on the location of the oxygen dispenser. In turn 3, the player moves to the location of the oxygen dispenser and picks up the oxygen tank. If the dispenser is producing the tank or when there is an oxygen tank on the oxygen dispenser's location, the allowableActions method in the OxygenDispenser class returns null so the player is not allowed to press the button.

Class OxygenTank
`public class OxygenTank **extends** Item`

The OxygenTank class inherits from Item. The class represents an oxygen tank item in the game. The oxygen tank has GameSkills.OXYGENTANK.

Class OxygenDispenser
`public class OxygenDispenser **extends** Ground`

The OxygenDispenser class inherits from Ground. It represents the oxygen dispenser location in the game. It has a checkOxygenTank method that checks if the oxygen tank is on its location. In its overridden allowableActions method, if there is no oxygen tank on its location, a newly instantiated DispenseOxygenTankAction is added into a List of Actions that can be performed when the player is adjacent to the oxygen dispenser.

Class DispenseOxygenTankAction
`public class DispenseOxygenTankAction **extends** Action`

The DispenseOxygenTankAction class inherits from Action. This class is responsible for the Action of dispensing the Item oxygen tank. In its overridden execute method, it creates a new OxygenTank item and places it on the location of the oxygen dispenser.

Part 3: Every turn spent in the moonbase costs the player one point of oxygen. A new oxygen tank holds 10 points worth of oxygen. If the player runs out of oxygen on the Moon, a safety system automatically transports them back to the rocket's location on Earth.

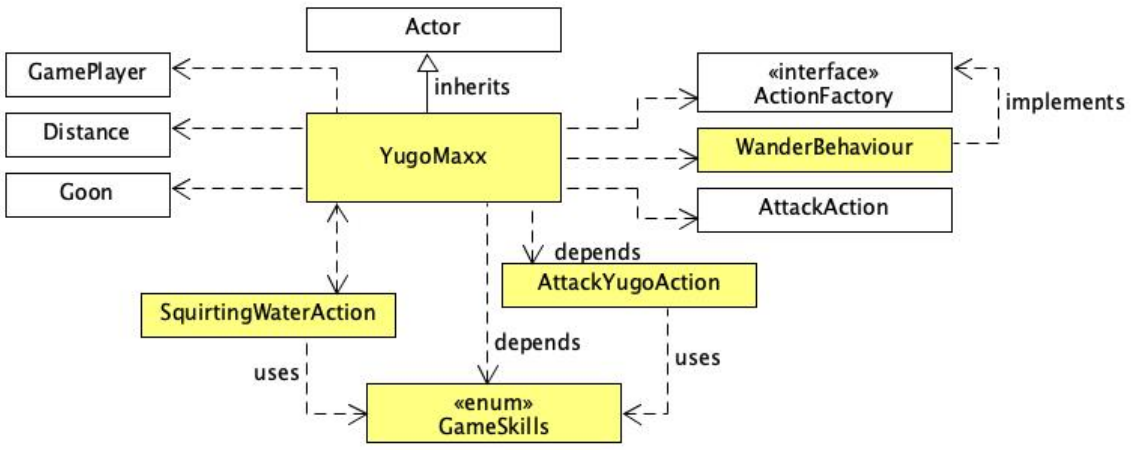
The newly added methods in the GamePlayer class are addOxygenPoints, onTheMoon, removeOxygenTank.

Proposed solution:
The player is a GamePlayer object in the game. We have added a Counter attribute, oxygenPoints that represents the player's oxygen points. In the player's playTurn method, it checks if the player has picked up an oxygen tank by calling the addOxygenPoints method. This method checks if the player has an Item with GameSkills.OXYGENTANK and not GameSkills.USEDOXYGENTANK. If it's true, this means that the player has picked up a new oxygen tank and 10 oxygen points will be added to the oxygenPoints attribute. Then, the GameSkills.USEDOXYGENTANK will be added to the oxygen tank to prevent the method from adding 10 more oxygen points to the oxygenPoints attribute in the future.

A method called onTheMoon has been added to the GamePlayer's class to check if the player is on the Moon by comparing the player's current map with Moon.getMap(). If it's true, the decrementOxygenPoints method will be executed whereby it decrements the player's oxygenPoints attribute by one on each turn.

To implement the safety system, when the player is on the Moon and their oxygen points are less than or equals to 0, the playTurn method will add the RocketToEarth class' getAllowableActions into Actions so that the player will be transported back to the rocket's location on Earth. Then the removeOxygenTank method is called to remove the used oxygen tanks from the player's inventory.

Final boss fight
The newly added classes are YugoMaxx, SquirtingWaterAction, AttackYugoAction, Water, FillEmptyPistolAction.



Based on the **class diagram** above, YugoMaxx inherits from Actor. The YugoMaxx class has a List attribute of type ActionFactory that stores the behaviour of YugoMaxx. In its playTurn method, it uses the Distance class' isAdjacent method to check if it's adjacent to the player. If it is not adjacent to the player, it uses the WanderBehaviour class to wander around the map. Else, it uses the AttackAction class to attack the player. Since it hits as hard as Goon, it uses Goon's final static GOON_DAMAGE variable for its own damage.

In YugoMaxx's getAllowableActions method, it adds the AttackYugoAction into a List of Actions that can be performed. It also checks if the player has an Item with GameSkills.PISTOLISFULL. If the player has a filled water pistol and YugoMaxx's exoskeleton is present, it adds SquirtingWaterAction into a List of Actions that can be performed by the player. The AttackYugoAction allows the player to attack Yugo Maxx. The AttackYugoAction checks if Yugo Maxx's exoskeleton is present by calling YugoMaxx class' hasExoskeleton method. If it's true, the player's attack will have 0 damage as its exoskeleton makes it invulnerable to damage.

The SquirtingWaterAction uses the GameSkills enum by replacing the water pistol's skill, GameSkills.PISTOLISFULL to GameSkills.PISTOLISEMPTY. Its execute method has a 70% chance of successfully removing Yugo Maxx's exoskeleton. If it is successful, it uses YugoMaxx class' removeExoskeleton method to remove Yugo Maxx's exoskeleton.

YugoMaxx is instantiated on the Moon map more than 10 squares away from the rocket. The water pistol item is also instantiated on the Moon map and it starts out empty so it has GameSkills.PISTOLISEMPTY. The water pistol is instantiated more than 5 squares away from the rocket.

Class YugoMaxx
```
public class YugoMaxx extends Actor
```

The YugoMaxx class inherits from Actor. This class represents the mini boss on the Moon which is YugoMaxx. YugoMaxx has a boolean attribute called exoskeleton that indicates whether the exoskeleton is present. In its overridden getAllowableActions method, it adds the AttackYugoAction into a List of Actions that can be performed. It also checks if the player has an Item with GameSkills.PISTOLISFULL. If the player has a filled water pistol and YugoMaxx's exoskeleton is present, it adds SquirtingWaterAction into a List of Actions that can be performed by the player.

Class SquirtingWaterAction
```
public class SquirtingWaterAction extends Action
```

The SquirtingWaterAction class inherits from Action. This class is responsible for the Action that squirts water at YugoMaxx. It has an overridden execute method so that there's a 70% chance of destroying YugoMaxx's exoskeleton. Then it will empty the player's water pistol by replacing GameSkills.PISTOLISFULL with GameSkills.PISTOLISEMPTY. The water pistol will be emptied regardless of whether the player hits or misses.

Class AttackYugoAction
```
public class AttackYugoAction extends AttackAction
```

The AttackYugoAction class inherits from AttackAction. This class is responsible for the Action that attacks YugoMaxx. The AttackYugoAction class allows the player to attack Yugo Maxx. It checks if Yugo Maxx's exoskeleton is present by calling YugoMaxx class' hasExoskeleton method. If it's true, the player's attack will have 0 damage as its exoskeleton makes it invulnerable to damage.

Class Water
```
public class Water extends Ground
```

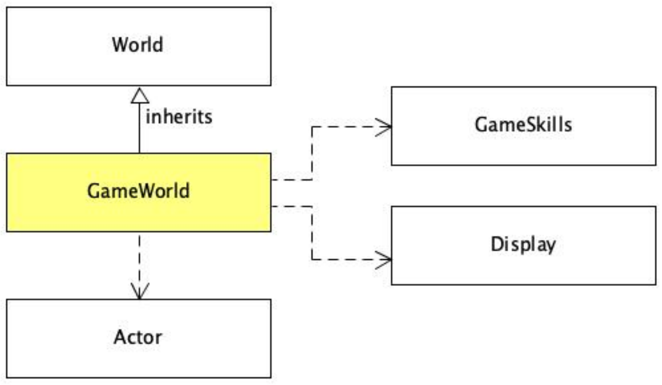
The Water class inherits from Ground. The class represents the water terrain. When there is a collection of water, it represents a pool of water on the map. It has an overridden canActorEnter method that returns false so that actors cannot walk on water. It has a containsPistol method that checks if the actor's inventory has an Item with GameSkills.PISTOLISEMPTY. In the overridden allowableActions method, the containsPistol method is called and if it's true, it will add the FillEmptyPistolAction into a List of Actions that the actor can perform.

Class FillEmptyPistolAction
```
public class FillEmptyPistolAction extends Action
```

The FillEmptyPistolAction class inherits from Action. This class is responsible for filling the empty water pistol item. It removes the skill, GameSkills.PISTOLISEMPTY on the water pistol and adds the GameSkills.PISTOLISFULL to indicate that the water pistol has been filled.

Ending The Game
The newly added classes are GameWorld and QuitGameAction.

Class GameWorld
```
public class GameWorld extends World
```

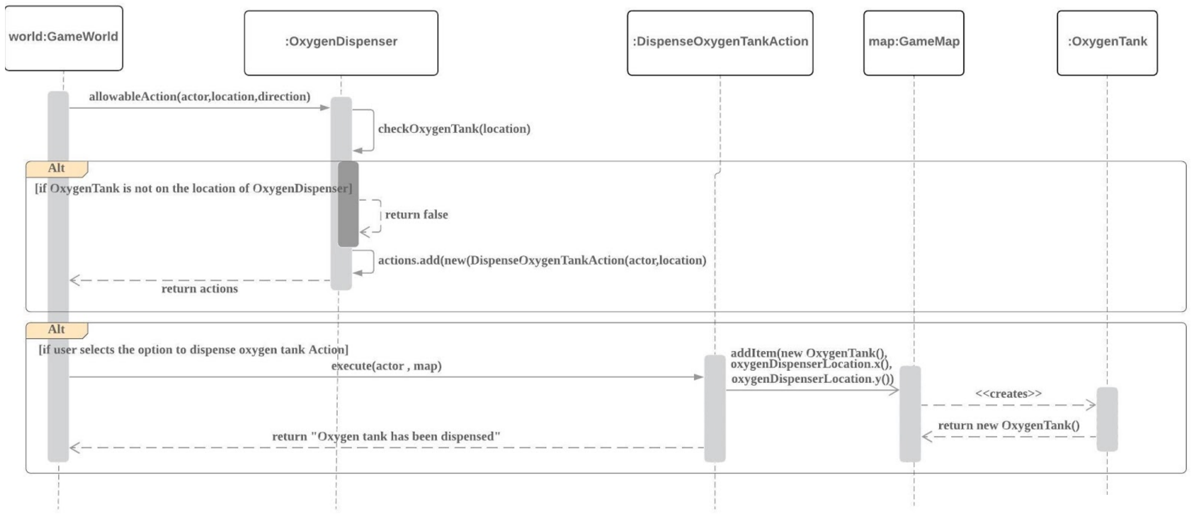


The GameWorld class inherits from World. The class is responsible for the game endings where the player wins or loses. In its overridden stillRunning method, it checks if the player has YugoMaxx's body (which has GameSkills.YUGOBODY) by calling the checkYugoMaxxBody method. If it's true, the stillRunning method returns false. The overridden endGameMessage returns "Player loses" if the player is knocked out, "Player wins" if checkYugoMaxxBody is true and "Game ended" if the player quits the game.

Class QuitGameAction
```
public class QuitGameAction extends Action
```

The QuitGameAction class inherits from Action. This class is responsible for the Action that allows the player to quit the game. The GamePlayer class adds a newly instantiated QuitGameAction into a list of Actions that can be performed by the player in its playTurn method.

Interaction diagram (To dispense an oxygen tank)


world:GameWorld
:OxygenDispenser
:DispenseOxygenTankAction
map:GameMap
:OxygenTank
allowableAction(actor,location,direction)
checkOxygenTank(location)
Alt
[if OxygenTank is not on the location of OxygenDispenser]
return false
actions.add(new(DispenseOxygenTankAction(actor,location)
return actions
Alt
[if user selects the option to dispense oxygen tank Action]
execute(actor , map)
addItem(new OxygenTank(), oxygenDispenserLocation.x(), oxygenDispenserLocation.y())
<<creates>>
return new OxygenTank()
return "Oxygen tank has been dispensed"

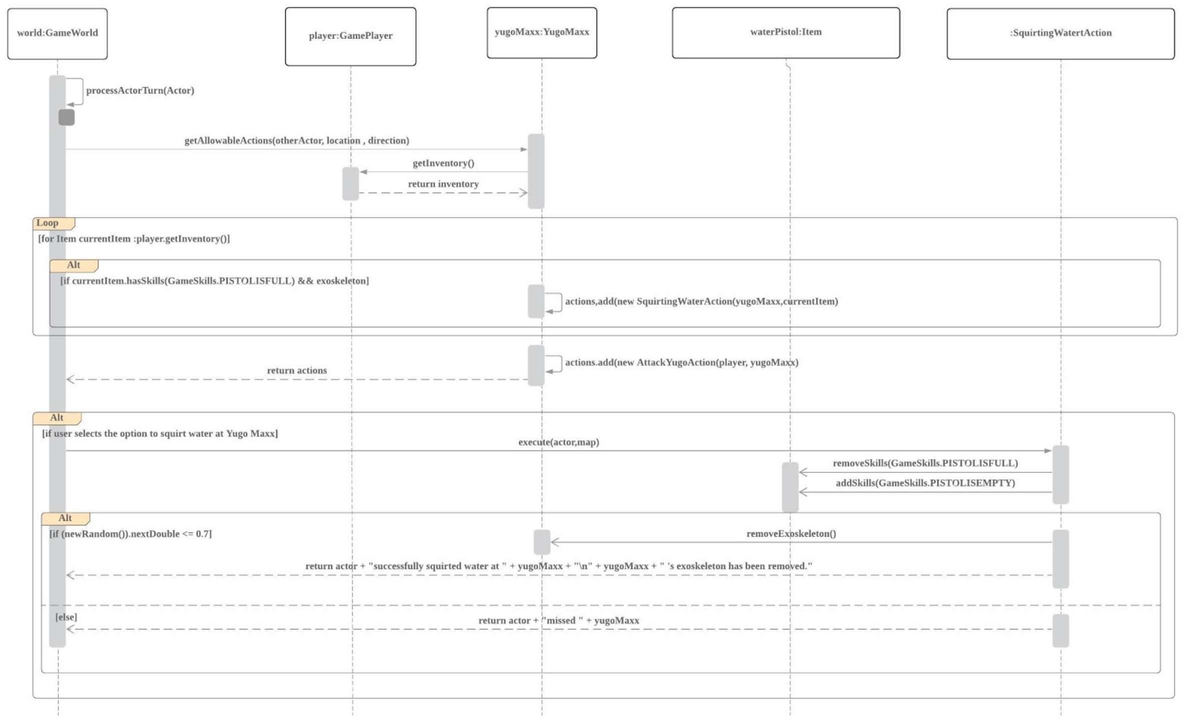
The interaction diagram above illustrates how the oxygen tank is dispensed.

If the player is adjacent to the OxygenDispenser object, it calls the OxygenDispenser class' allowableActions(actor,location,direction) method. Then, it checks if the OxygenTank Item that has the GameSkills.OXYGENTANK is on the location of OxygenDispenser.

If the location of OxygenDispenser does not contain an OxygenTank Item, a newly instantiated DispenseOxygenAction is added into a list of actions and the list of actions is returned so that the player can press the button to dispense an oxygen tank. However, if OxygenTank Item is found on the location of OxygenDispenser, an empty list of actions will be returned.

When the player chooses to dispense an oxygen tank , an oxygen tank will be added on the location of OxygenDispenser. The String “Oxygen Tank has been dispensed” will be returned.

Interaction diagram (To squirt water from a full water pistol onto Yugo Maxx)


world:GameWorld
player:GamePlayer
yugoMaxx:YugoMaxx
waterPistol:Item
:SquirtingWatertAction
processActorTurn(Actor)
getAllowableActions(otherActor, location , direction)
getInventory()
return inventory
Loop
[for Item currentItem :player.getInventory()]
Alt
[if currentItem.hasSkills(GameSkills.PISTOLISFULL) && exoskeleton]
actions,add(new SquirtingWaterAction(yugoMaxx,currentItem)
actions.add(new AttackYugoAction(player, yugoMaxx)
return actions
Alt
[if user selects the option to squirt water at Yugo Maxx]
execute(actor,map)
removeSkills(GameSkills.PISTOLISFULL)
addSkills(GameSkills.PISTOLISEMPTY)
Alt
[if (newRandom()).nextDouble <= 0.7]
removeExoskeleton()
return actor + "successfully squirted water at " + yugoMaxx + "\n" + yugoMaxx + " 's exoskeleton has been removed."
[else]
return actor + "missed " + yugoMaxx

The interaction diagram above illustrates how player:GamePlayer performs the squirting water action on yugoMaxx:YugoMaxx.

Firstly, yugoMaxx checks if the player's inventory has a filled water pistol which has GameSkills.PISTOLISFULL. If the player's water pistol is filled and yugoMaxx's exoskeleton is present, a newly instantiated SquirtingWaterAction is added into a List of Actions and returned. The AttackYugoAction is always added into the List of Actions. This allows the player to either squirt water at yugoMaxx or attack yugoMaxx.

If the player chooses to squirt water at yugoMaxx, there's a 70% that it will be a successful attack. If it was a successful attack, yugoMaxx's exoskeleton will be destroyed, and a message "Player successfully squirted water at Yugo Maxx. Yugo Maxx's exoskeleton has been removed" will be returned. Else, player misses yugoMaxx this round and a message "Player missed YugoMaxx" will be printed.