Work Breakdown Agreement for FIT2099 Assignment 1

Team 3 Lab 10

1. Mikael Andrew Susanto (ID: 32332270)
2. Looi Teck San (ID: 32439180)
3. Tan Kah Hong (ID: 31110126)

We will separate the work into three parts:

1. All documents for REQ 1 - 2, sequence diagram for REQ 7
2. All documents for REQ 3 - 4, class diagram for REQ 7
3. All documents for REQ 5 - 6, design rationale for REQ 7

Mikael will be responsible for Part 1 and its design rationale, Reviewer: All of the members, Completion: Wednesday, 5 April 2022

Teck San will be responsible for Part 2 and its design rationale, Reviewer: All of the members, Completion: Wednesday, 5 April 2022

Kah Hong will be responsible for Part 3 and its design rationale, Reviewer: All of the members, Completion: Wednesday, 5 April 2022

Signed by
I accept this WBA. - Looi Teck San
I accept this WBA. - Tan Kah Hong
I accept this WBA. - Mikael Andrew Susanto

Rationale

Requirement 1

A new class has been created. Because the Tree class represents all of them and may be named its Parent class, Sprout, Sapling, and Mature will inherit from it. We'll also be following along the Object-Oriented Program design.

New class Sprout -> Inherit from Tree class:

It's associated to Goomba because it has a 10% chance of spawning Goomba on its position if no actor stands on the Sprout, which is why it depends on the abstract Actor class. Depending on the probability and whether or not the actor stands on it, each Sprout can spawn 0 or more Goomba.

New class Sapling -> Inherit from Tree class:

It's associated to Coin since it has a 10% chance of dropping a ($20) coin on its position every turn. Sapling can only grow from Sprout; hence it is likewise depended on Sprout (10 turns).

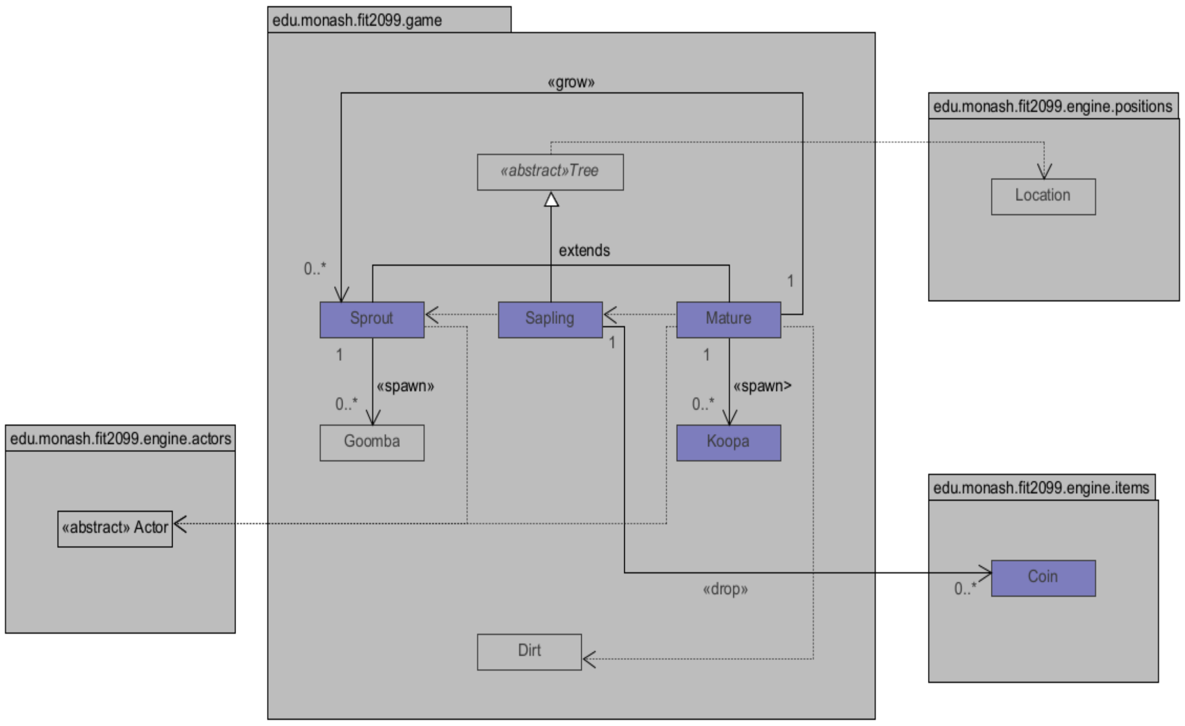
New class Mature -> Inherit from Tree class:

It's associated to Sprout since it can grow a new Sprout in its surroundings every 5 turns if it's a Dirt, which is why it's also depending on Dirt. It is also associated to Koopa because it has a 15% chance of spawning Koopa every turn if no actor stands on it. As a result, it is likewise dependent on the abstract Actor class.

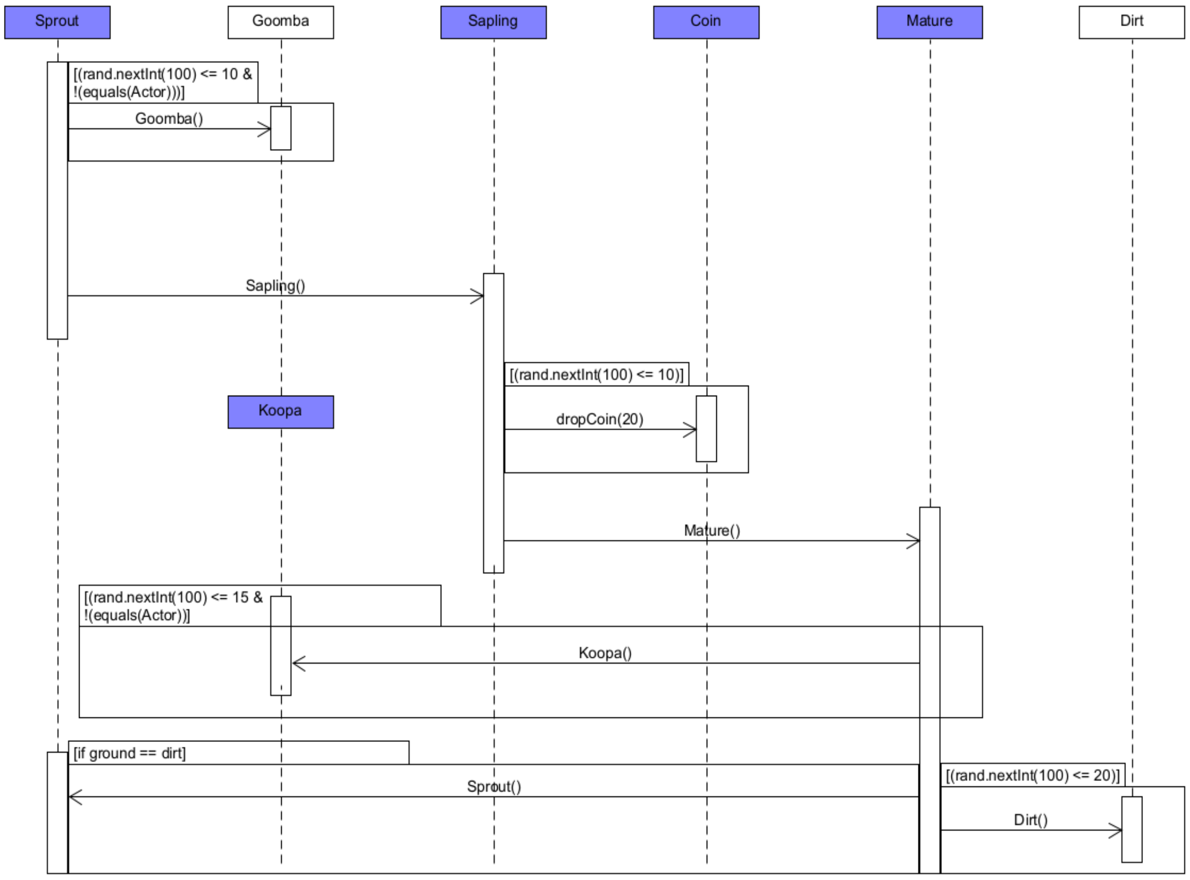
Abstract class Tree:

We will abstract the Tree class so that it may represent all trees. It will represent all of the trees and will be dependent on the Location class because we need to know where each tree is in order to spawn foes and drop a coin on its location.

Class Diagram:



Sequence Diagram:


Sprout
Goomba
Sapling
Coin
Mature
Dirt
[(rand.nextInt(100) <= 10 & !(equals(Actor)))]
Goomba()
Sapling()
Koopa
[(rand.nextInt(100) <= 10)]
dropCoin(20)
Mature()
[(rand.nextInt(100) <= 15 & !(equals(Actor))]
Koopa()
[if ground == dirt]
Sprout()
[(rand.nextInt(100) <= 20)]
Dirt()

Requirement 2

- We generated a new class named JumpBehaviour that inherits from the abstract Actor class and implements the Behaviour interface class, allowing the player to jump.

- Because it is the objects that the Player can jump on (each object has a different success rate), JumpBehaviour is dependent on the Sprout, Sapling, Mature, and Wall classes. It is also dependent on the Location because we need to know if the objects are adjacent to the Player.

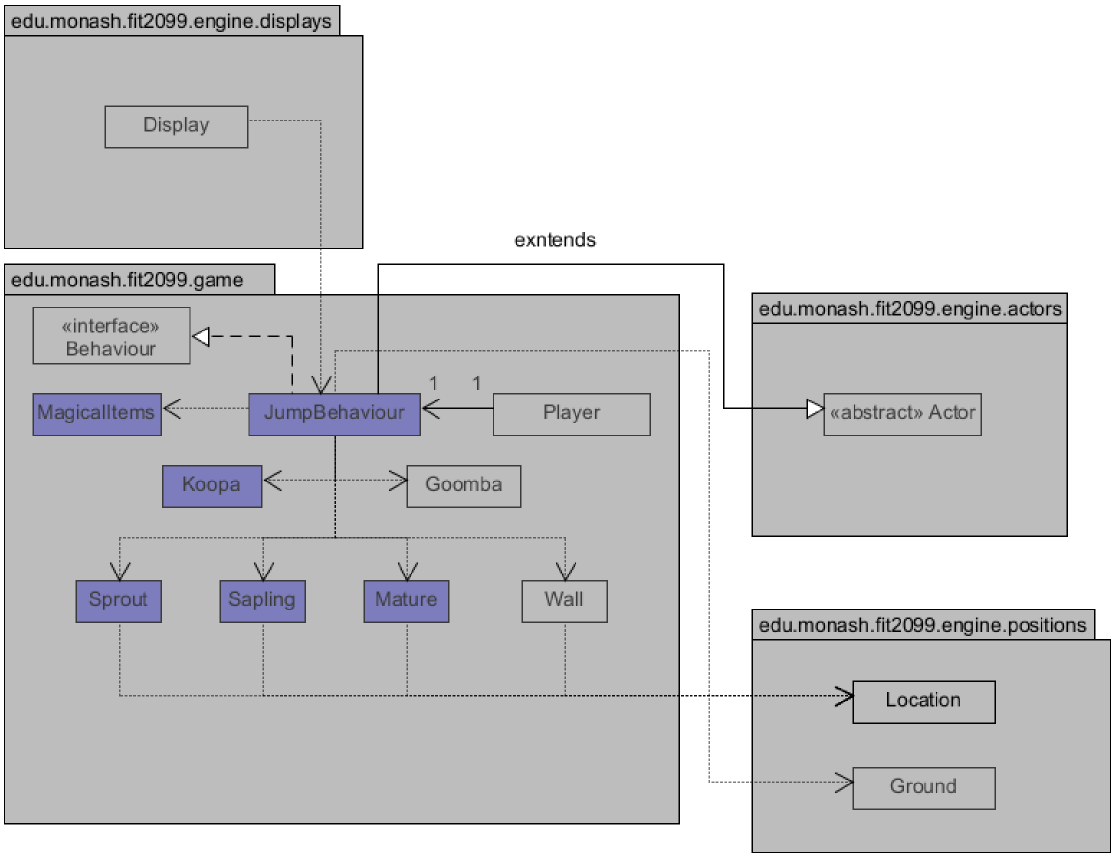
- Because the Actor can travel down immediately if the ground is not the objects from previously, it is also dependent on Ground to check if it's a normal ground or not.

- It will also be dependent on the MagicalItems class, because when the Player consumes the Super Mushroom, he will have a 100% success rate and will not suffer any fall damage.

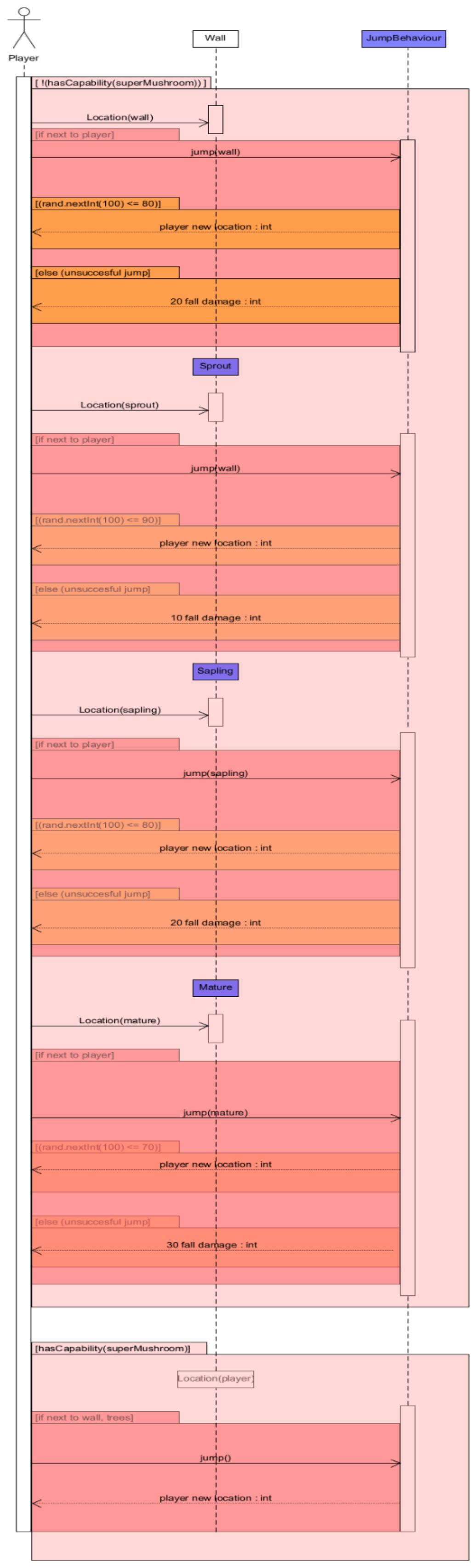
- The enemy's class will be depending on the JumpBehaviour class because if they are standing on high ground (Wall, Sprout, Sapling, Mature), they can fall straight down if the ground adjacent to them is normal ground.

- Finally, we need to change the Display menu since we need to add an action if the Player is next to a high ground so that the Player can jump to it.

Class Diagram:



Sequence Diagram:



Requirement 3

The new class I added in are EnemiesBehaviour, Koopa and SuperMushroom class.

New class Koopa class

- Create Koopa and it will handle all functionality relevant to Koopa, such as turn into shell, create SuperMushroom instance, behaviour of Koopa and as well as the damage and hit rate of Koopa. The new class will create Koopa which is the enemy required by the task
- dependency AttackBehaviour so I can let Koopa to attack the player automatically
- Inherit Actor class to have the method to implement Koopa and Goomba hp, Koopa and Goomba display character whether they are conscious, being hurt by player and etc
- Dependency on SuperMushroom to create SuperMushroom instance once the shell is broken
- Dependency on GameMap is to remove Koopa once the condition is met such as if the Koopa shell is broken

Modified Goomba class

- Changes: Add hurt() method to attack the player
- Inherit Actor class to have the method to implement Goomb and Goomba hp, Goomb and Goomba display character whether they are conscious, being hurt by player and as well as the damage and hit rate of Goomba
- dependency AttackBehaviour so I can let Goomba to attack the player automatically
- Dependency on GameMap is to remove Goomba once the condition is met such as if the Goomba is not conscious.

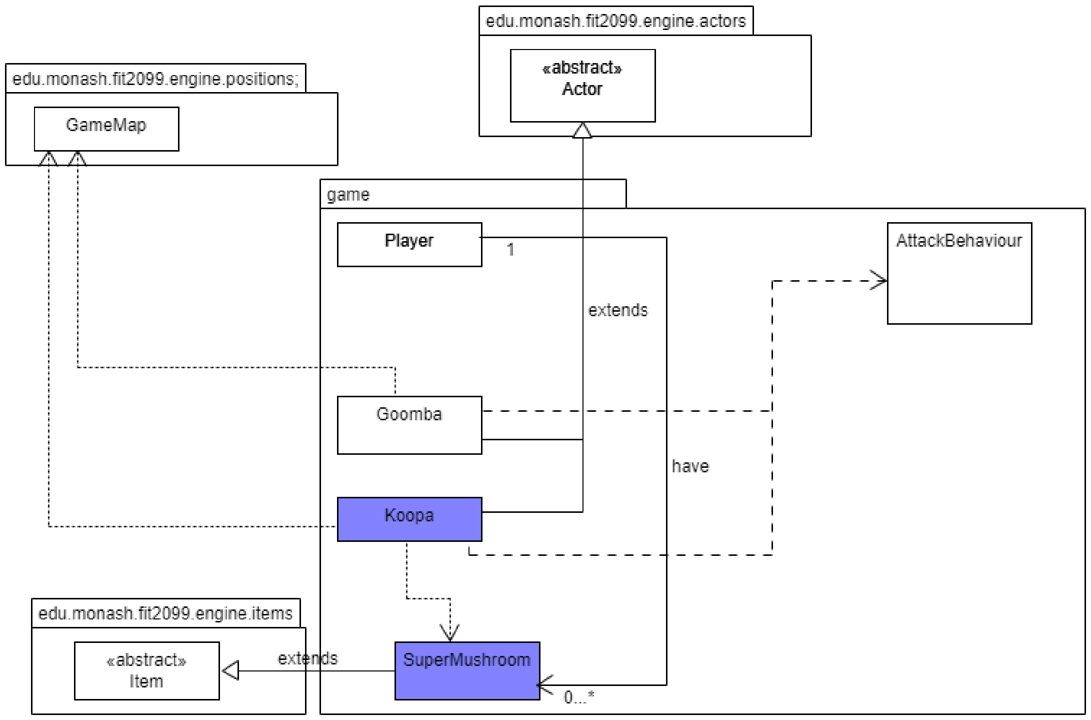
New class SuperMushroom class

- Represent the SuperMushroom instance of mushroom
- Player associate SuperMushroom class so that I can have a list of SuperMushroom instance in Player class, since a player can have 0 to many supermushroom in the inventory, so the list of supermushroom will be representing the player’s inventory.
- inherit abstract class item to have the method to add into player inventory.

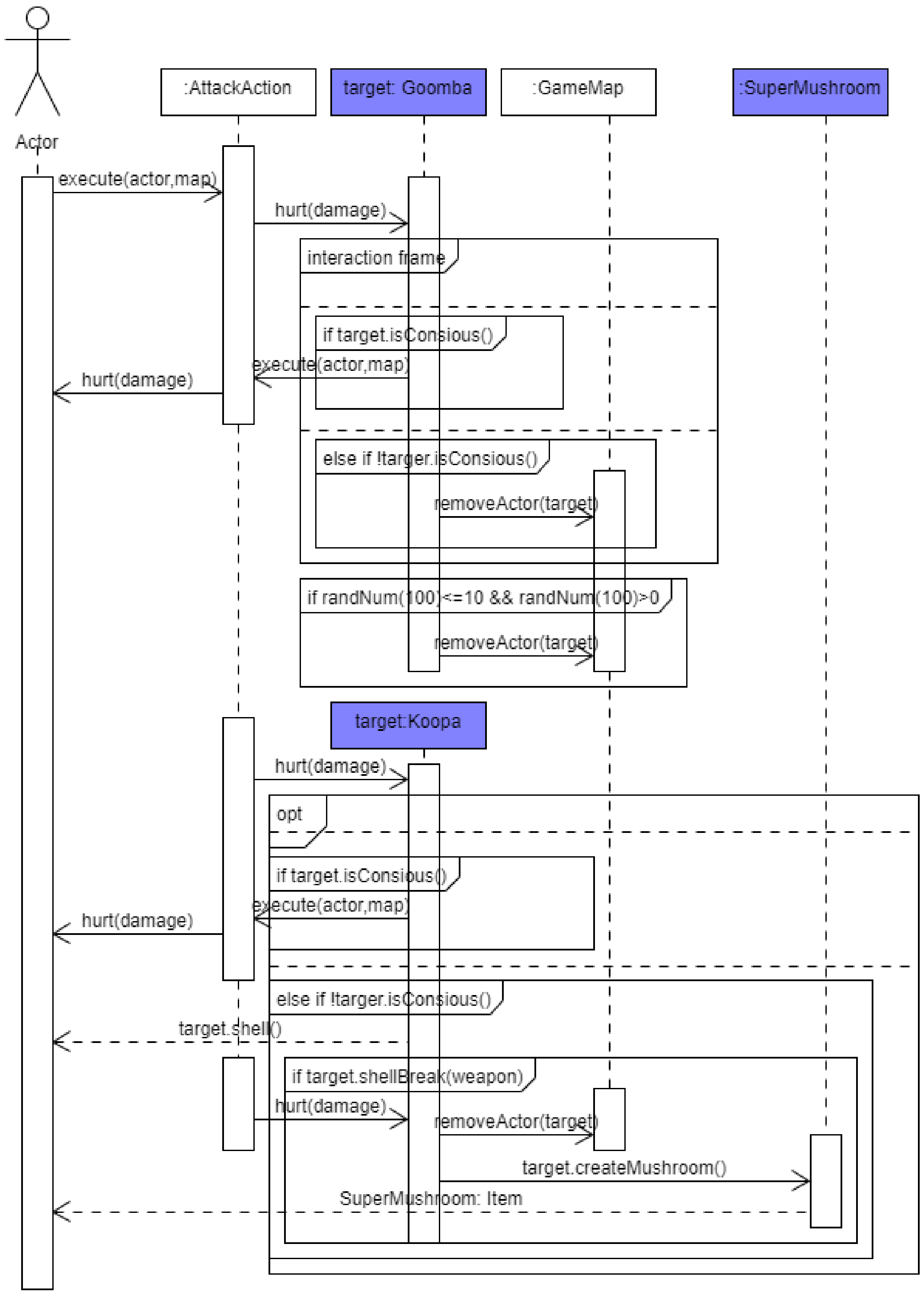
Requirement 3

(ps: also pushed onto git for these pictures, can look for the pictures in the folder if the pictures here are not that clear to read)

Class diagram



Sequence diagram



Requirement 4

New class SuperMushroom class

- create the SuperMushroom instance
- Player class have an associate on SuperMushroom so that the player can have 1 to many superMushroom in inventory
- inherit abstract class item to have the method to add into player inventory.

New class Coin[covered in another requirement]

New class WalkBehaviour

- To update the player walk behaviour after using PowerStar where the player can walk through any type of ground
- Implements interface class Behaviour because the player can have different behaviour so later the player can have a list or hashmap of behaviour

New Class JumpBehaviour

- To update the player jump behaviour after using SuperMushroom the player have 100% success jump rate
- Implements interface class Behaviour because the player can have different behaviour so later the player can have a list or hashmap of behaviour

New Class PowerStar Class

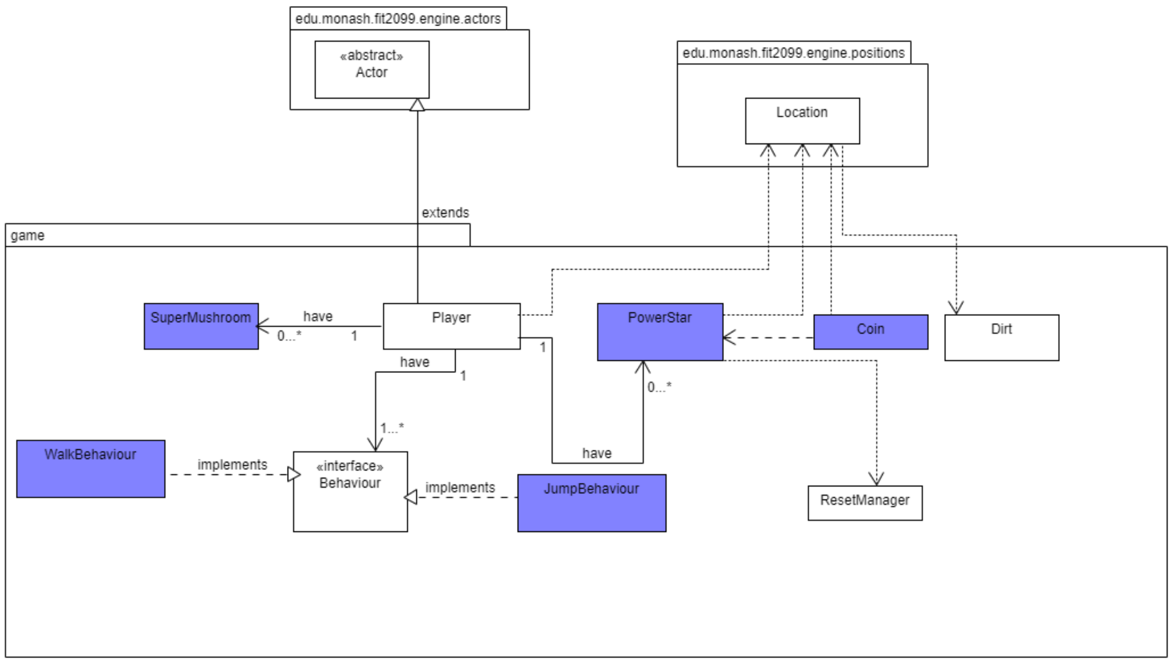
- create the PowerStar instance
- Player class have an associate on PowerStar so that the player can have 1 to many powerStar in inventory
- inherit abstract class item to have the method to add into player inventory.

Modified player class

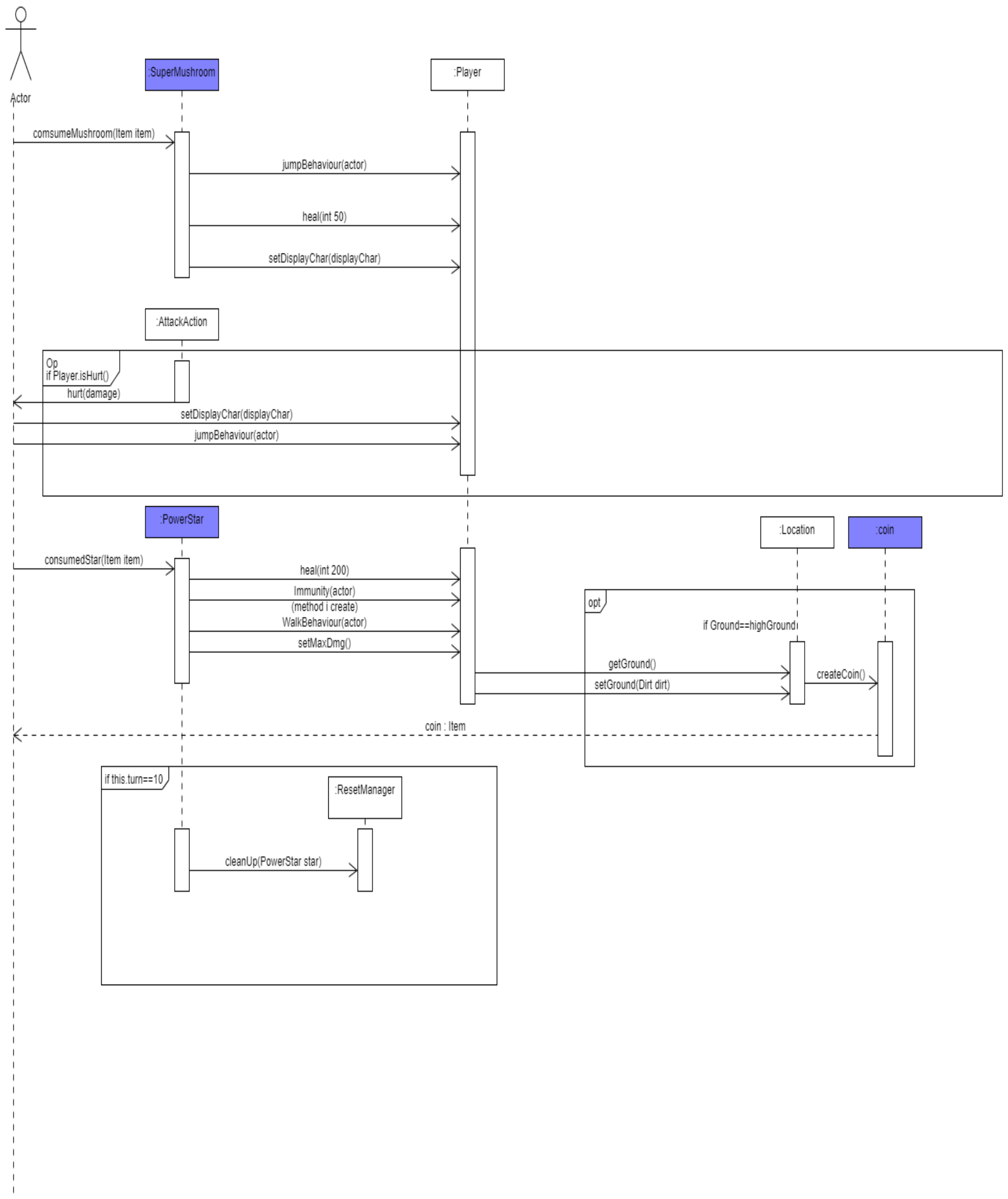
- To change the player behaviour, hp, set player display character
- Use the setDisplayChar(char displayChar) inherited from the super class to change the player name for example from m to M
- Add new method Immunity() which will set the player to immune from the attack of enemies.
- Add new method setMaxDmg() which will set the player Damage to max and kill enemy with only one hit

Requirement 4

Class diagram



Sequence diagram



Requirement 5

New Class: Toad

- Represents the Toad, will handle all functionality relevant to Toad, such as trading and speaking.
- Inherits Actor, hence obtains all the properties and functionality of an Actor, such as spawning in location, detecting proximity to Player, etc.

New Class: TradeAction

- Handles all trading-related actions.
- Inherits Action to access all functionality related to Actions, such as displaying it in the console menu.
- With this implementation, TradeAction holds a list of SellableItems. It is assumed that the Toad-shop has unlimited stock (Player can purchase the same item multiple times). If this is not the case, a different implementation will be favoured: using the inventory of Toad to represent the shop's items.

New Static Class: Wallet

- Handles coin storing, picking up, and deduction (for trade)
- Chosen to be Static because Wallet never needs to be instantiated
- Alternative: have Wallet as a permanent item in Player's inventory. This was not chosen because Wallet doesn't share many characteristics with other items.

New Class: Coin

- Represents the coin item
- Inherits Item to access functionality common to other items which is needed for Coin, like spawning in a position, and being able to be picked up.
- Sapling has dependency on Coin to potentially spawn it in its location.

New Interface: SellableItems

- Handles the features unique to items that are sold in the shop.
- Items that implement this interface have an extra attribute price. This way, the price property of the object is stored inside the item's Class. (Class responsible for own properties)
- If another item is to be included in the shop, all it needs to do is to implement this interface and add a price property. Scalable.

New Class: Wrench

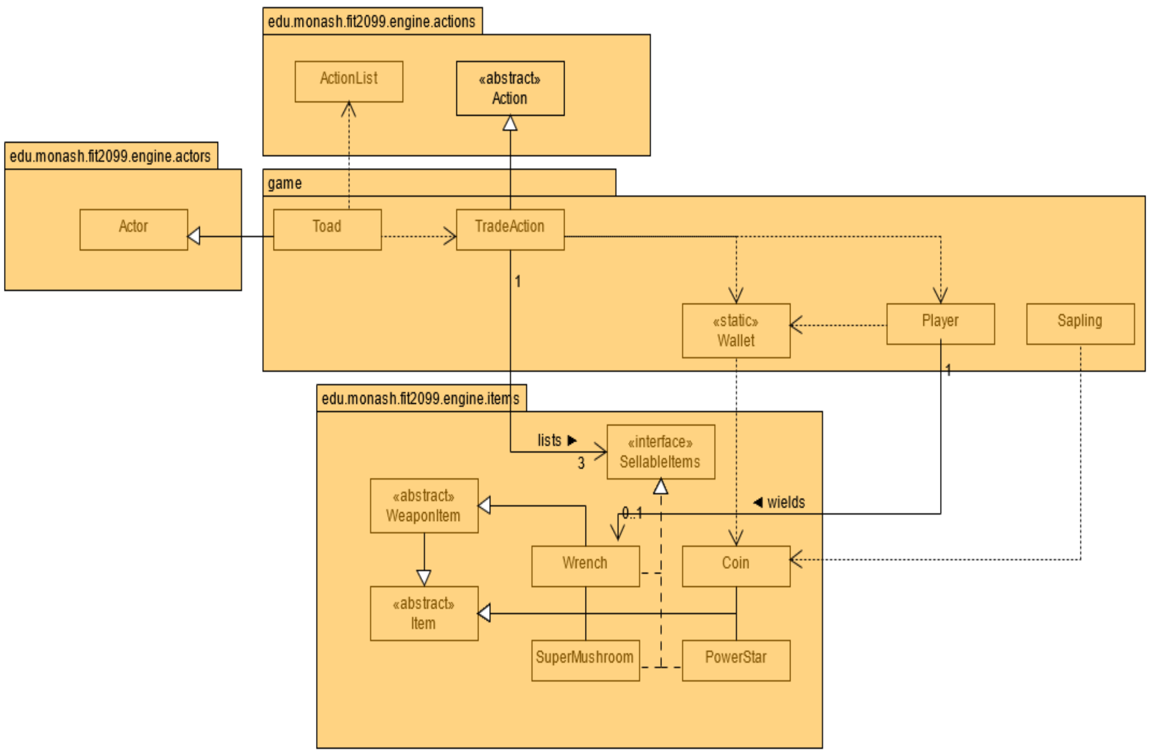
- Represents the Wrench item
- Inherits WeaponItem to access functionality common to other WeaponItems, such as chance to hit, damage.
- Implements SellableItems.

New Classes: SuperMushroom and PowerStar

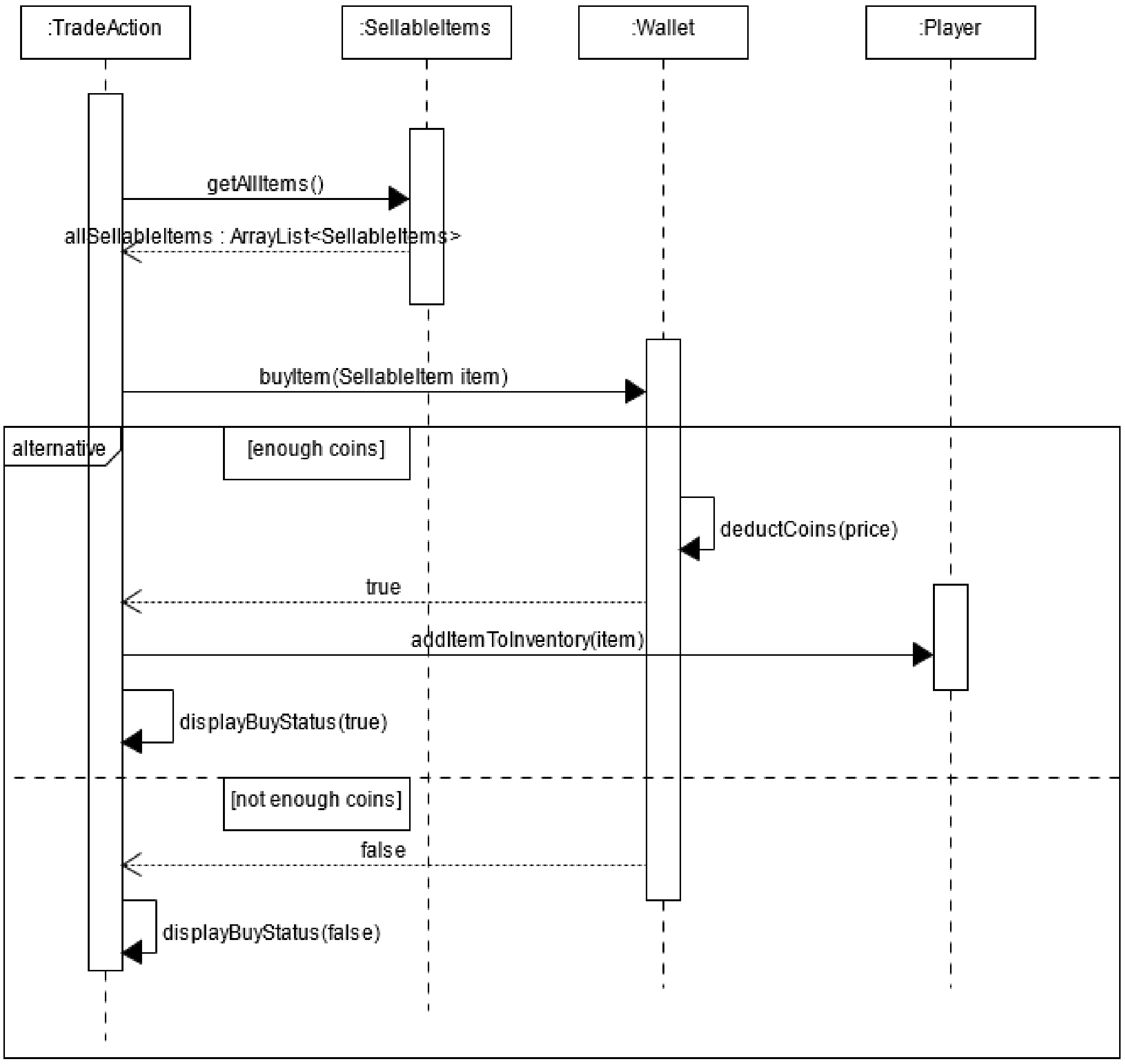
[Covered in REQ 4]

- Implements SellableItems.

Class Diagram:



Sequence Diagram:



Requirement 6

New Class: SpeakAction

- Handles the speaking feature of Toad.
- Inherits Action, to access the functionality of Actions such as printing to the console.
- Checks CapabilitySet to get whether PowerStar effect is active.
- Checks inventory of Player to get whether Player has a wrench.
- Has an array of Strings for all possible dialogue of Toad. New dialogue can be easily added.

New Class: Toad

- Represents the Toad, will handle all functionality relevant to Toad, such as trading and speaking.
- Inherits Actor, hence obtains all the properties and functionality of an Actor, such as spawning in location, detecting proximity to Player, etc.

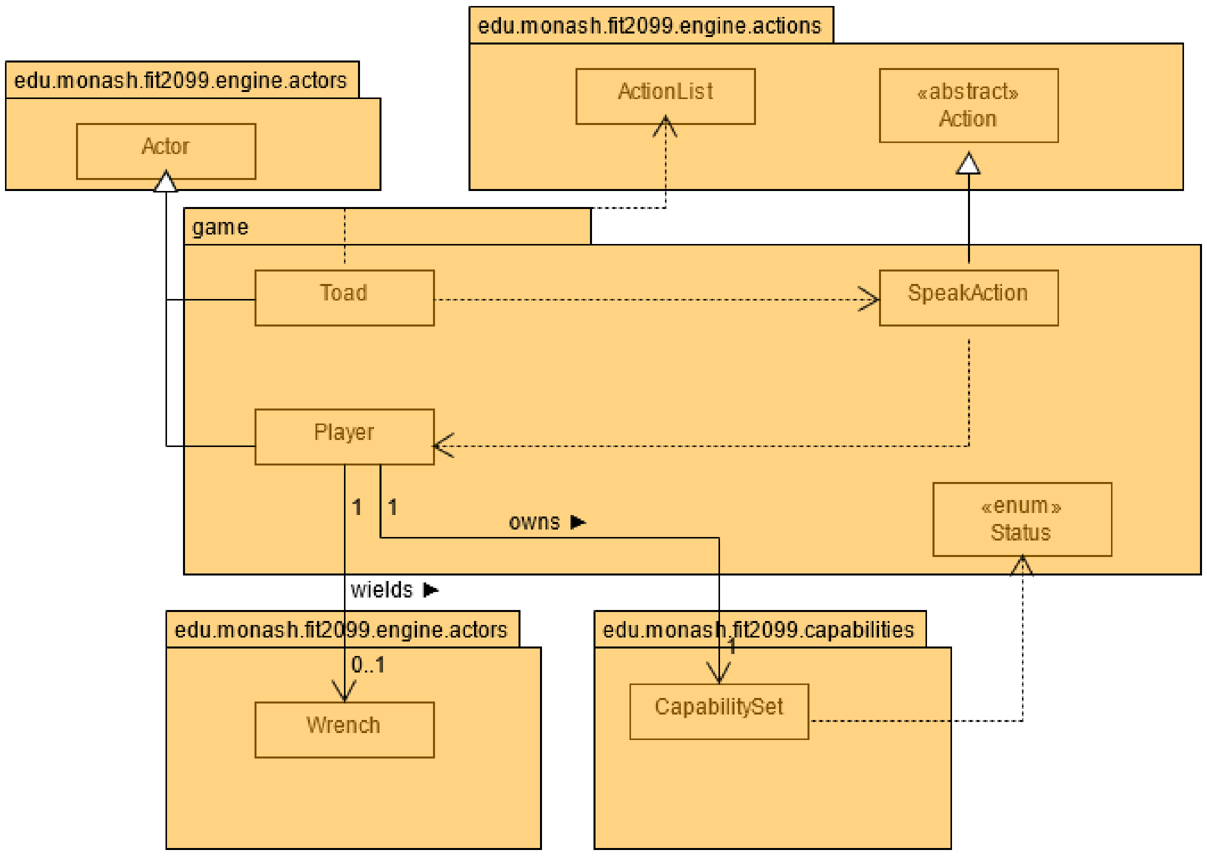
New Class: Wrench

- Represents the Wrench item
- Inherits WeaponItem to access functionality common to other WeaponItems, such as chance to hit, damage.
- SpeakAction can easily check whether wrench is in Player's inventory.

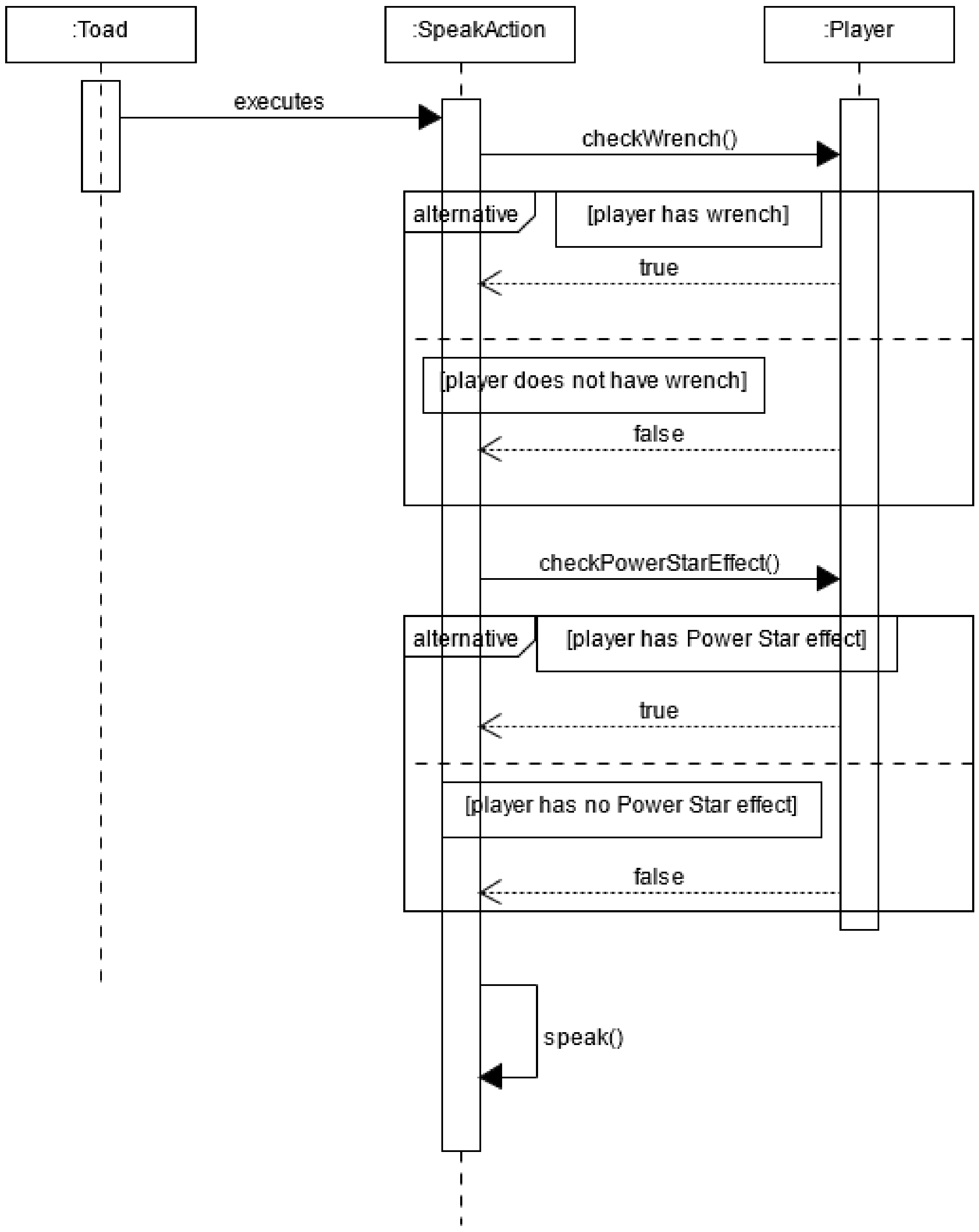
Changed Enum: Status

- Extra status added representing the PowerStar effect.
- Makes use of the existing system to scalably add new effects.

Class Diagram:


edu.monash.fit2099.engine.actors
Actor
edu.monash.fit2099.engine.actions
ActionList
«abstract»
Action
game
Toad
SpeakAction
Player
1
1
owns ▶
«enum»
Status
wields ▶
edu.monash.fit2099.engine.actors
0..1
Wrench
edu.monash.fit2099.capabilities
1
CapabilitySet

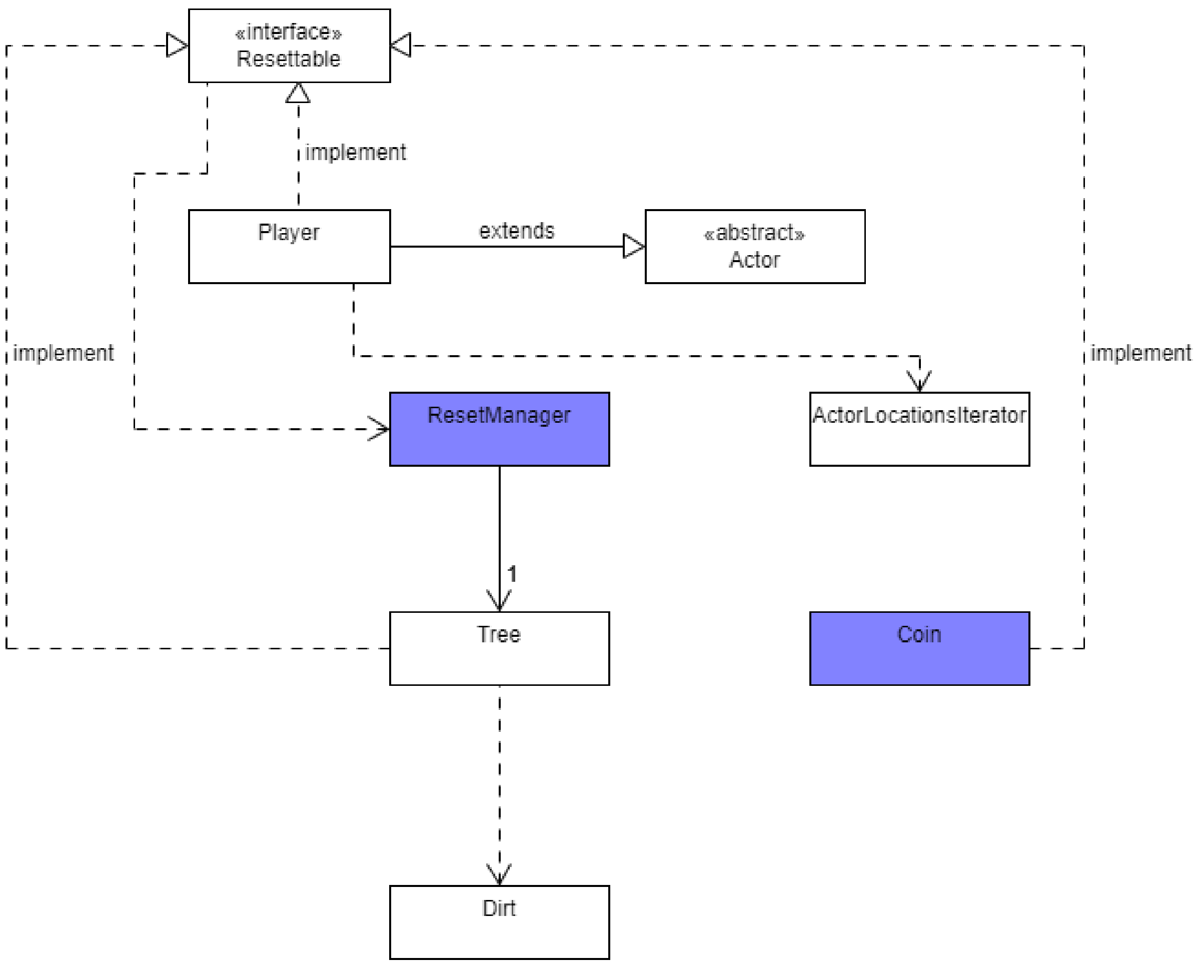
Sequence Diagram:



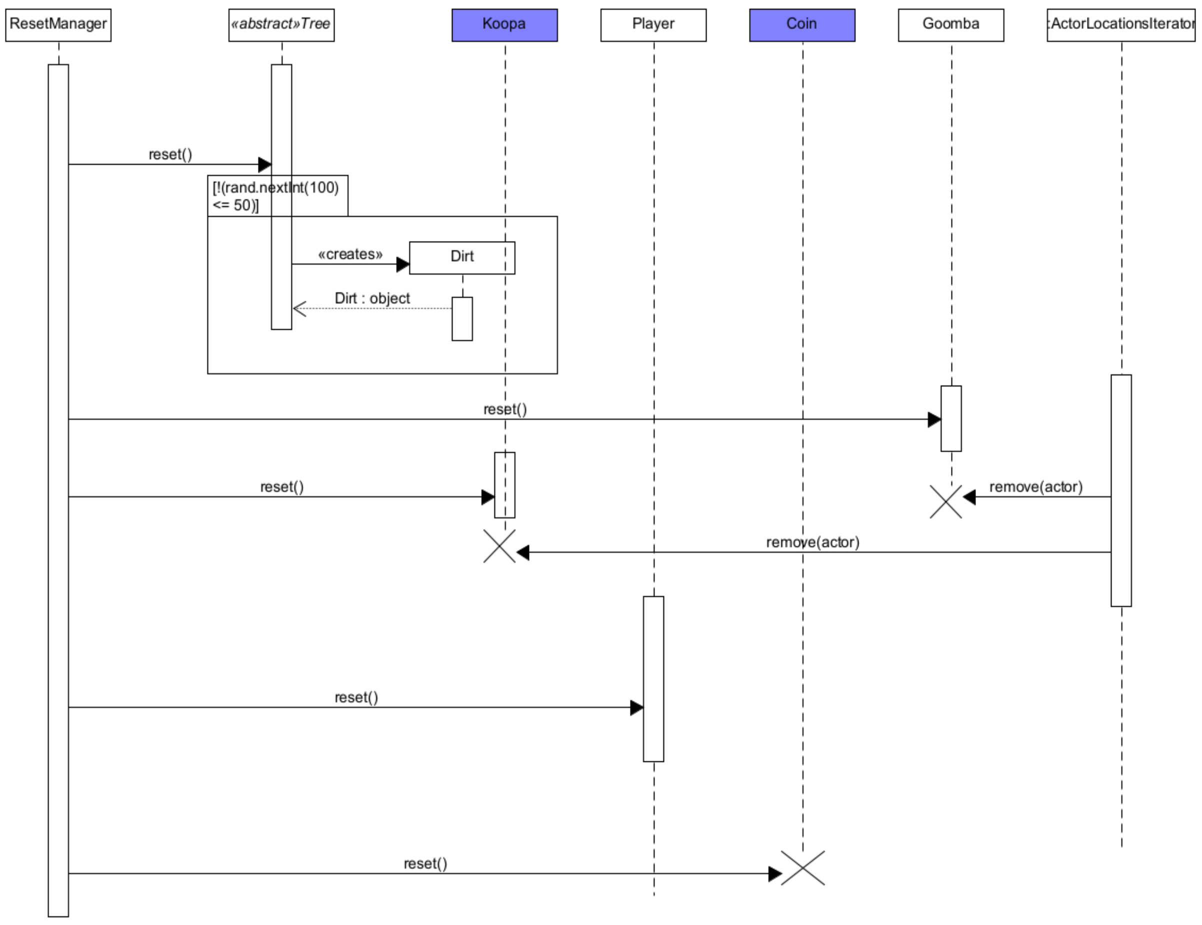
Requirement 7

- Tree, Enemy, Player, Coin implements Resettable
- By using the Resettable interface, any additional features that happen upon a reset can be easily added.
- All Resettables will implement a reset method that handles their own behaviour upon reset.

Class Diagram:



Sequence Diagram:


ResetManager
«abstract»Tree
Koopa
Player
Coin
Goomba
:ActorLocationsIterator
reset()
[!(rand.nextInt(100)
<= 50)]
«creates»
Dirt
Dirt : object
reset()
reset()
remove(actor)
remove(actor)
reset()
reset()