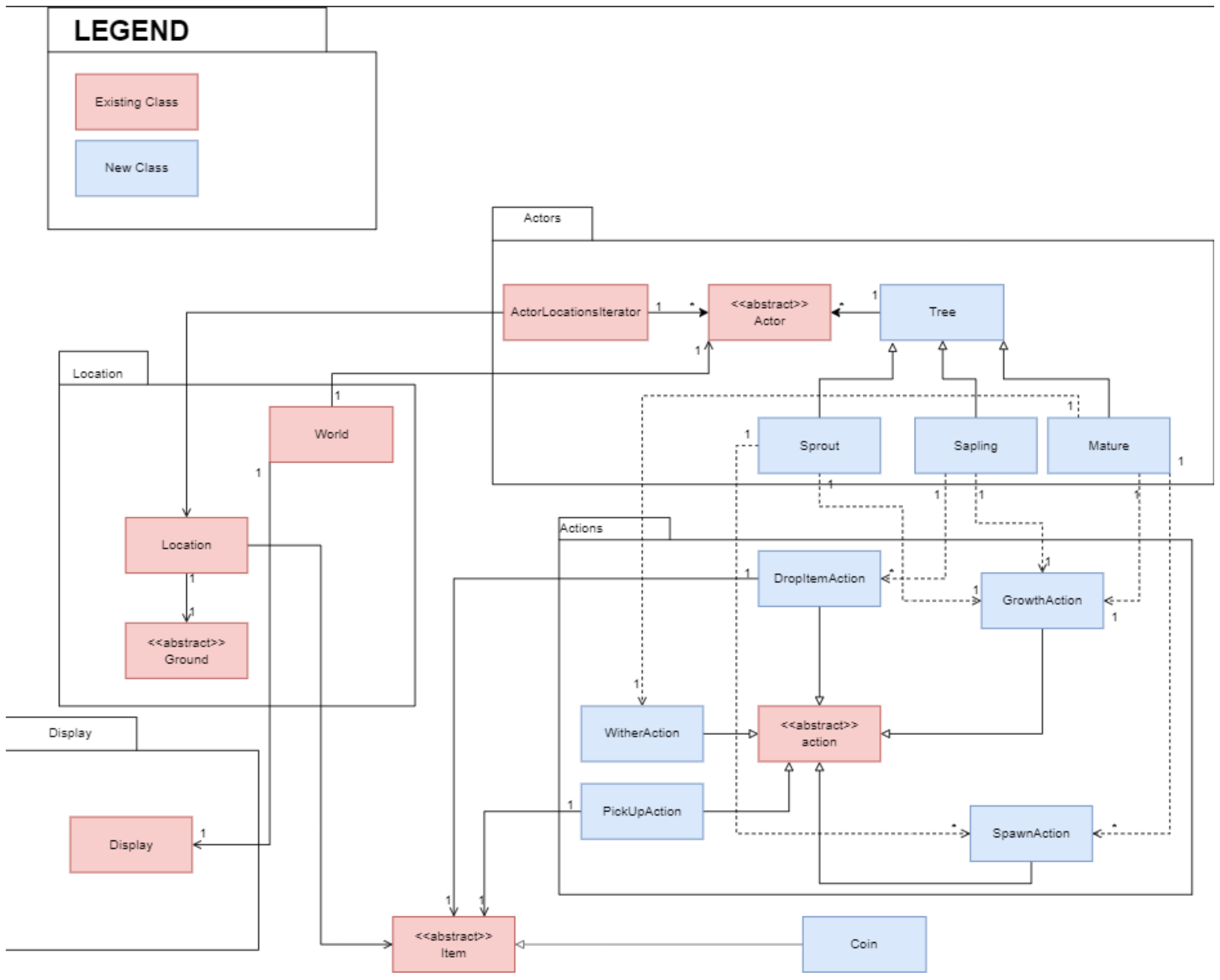
Requirements Documentation:

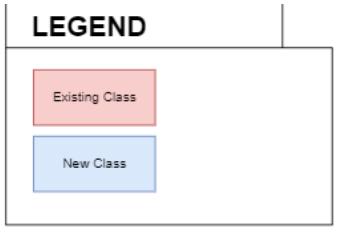
Requirement 1:

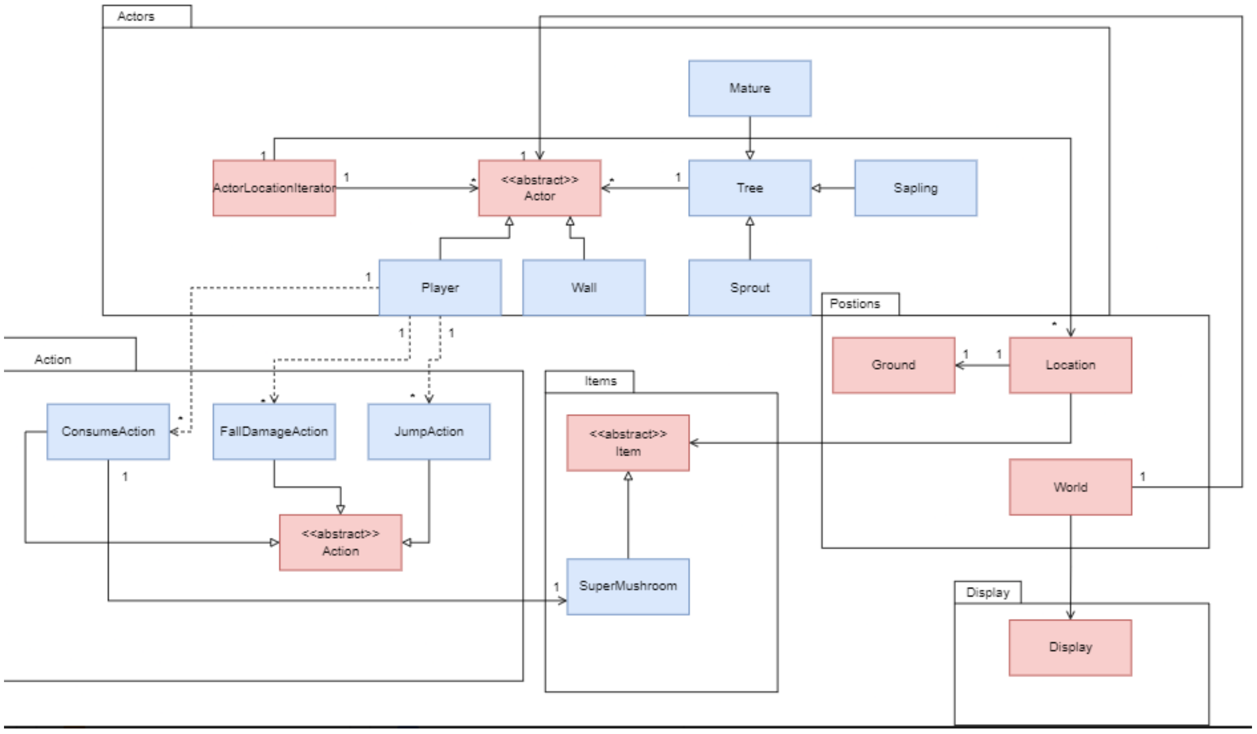


Design Rationale:

The interaction between sprout, sapling and mature to enrich the world requires the implementation of growth and wither allows for a spawn cycle to enrich the world, the spawn cycle enables for random generation of sprouts throughout a duration of approximately 40 iterations. This spawn system adheres to single-responsibility principle as its only task is to spawn and only spawn goomba.The introduction of coin class which is obtained from a drop from the sapling which ultimately offers the player the chance to increase their balance.

Requirement 2:

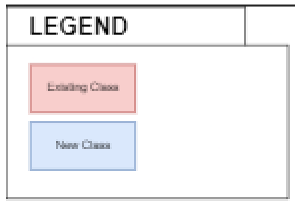


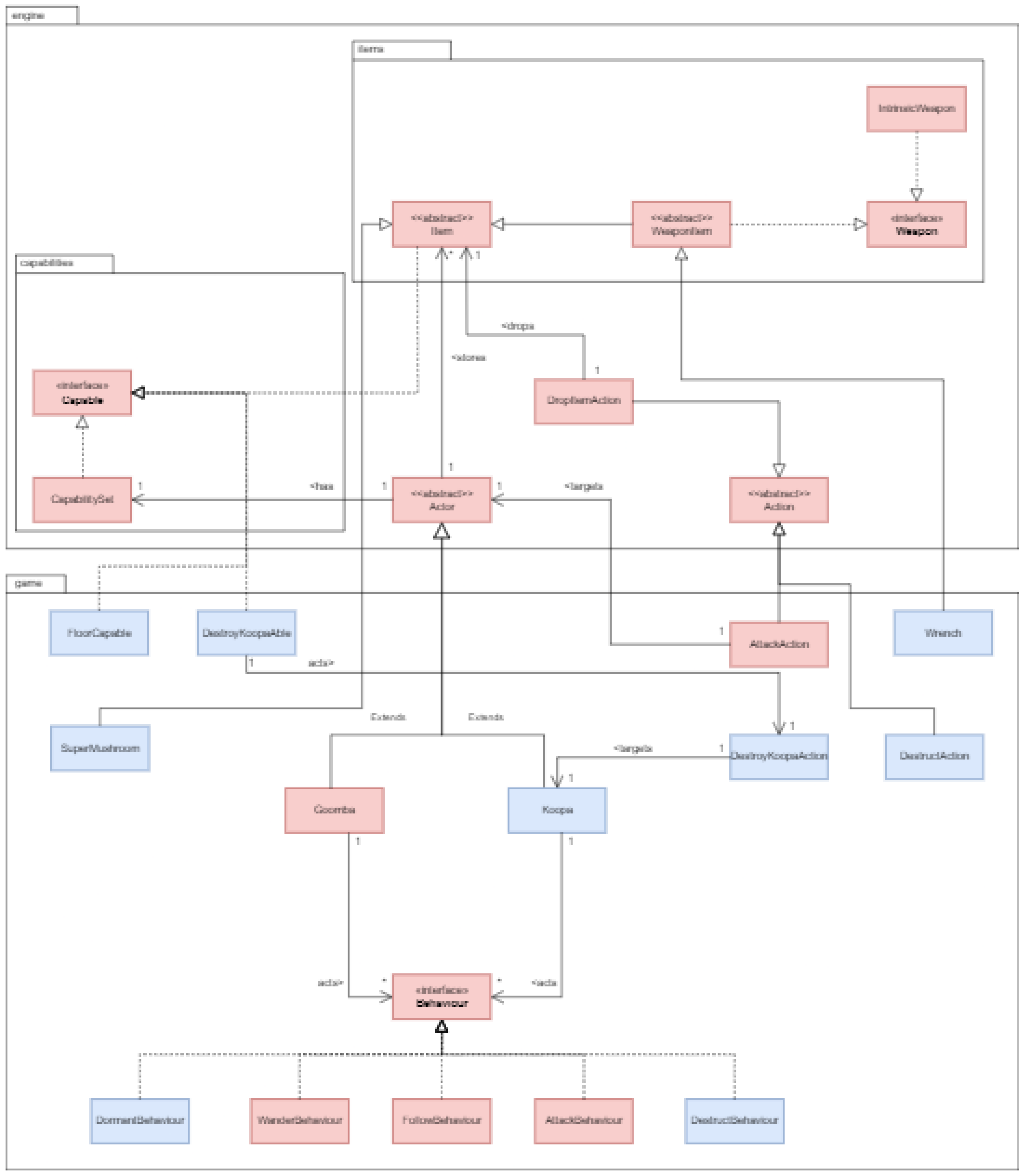


Design Rationale:

The implementation of jump option provides the players a chance to jump onto a higher ground, the display enabled for a message to be imprinted for users to see whether or not they are successful with their respective jumps. This is an example of open-closed principle as we add further extension e.g high jump but the overall concept of jumping should not be modified.The falldamage action will determine fall damage based of what tree/wall that you failed to jump, further implementation of consumeAction facilitates for the usage of super mushroom will guaranteed chance of a successful jump.

Requirement 3:





Design Rationale:

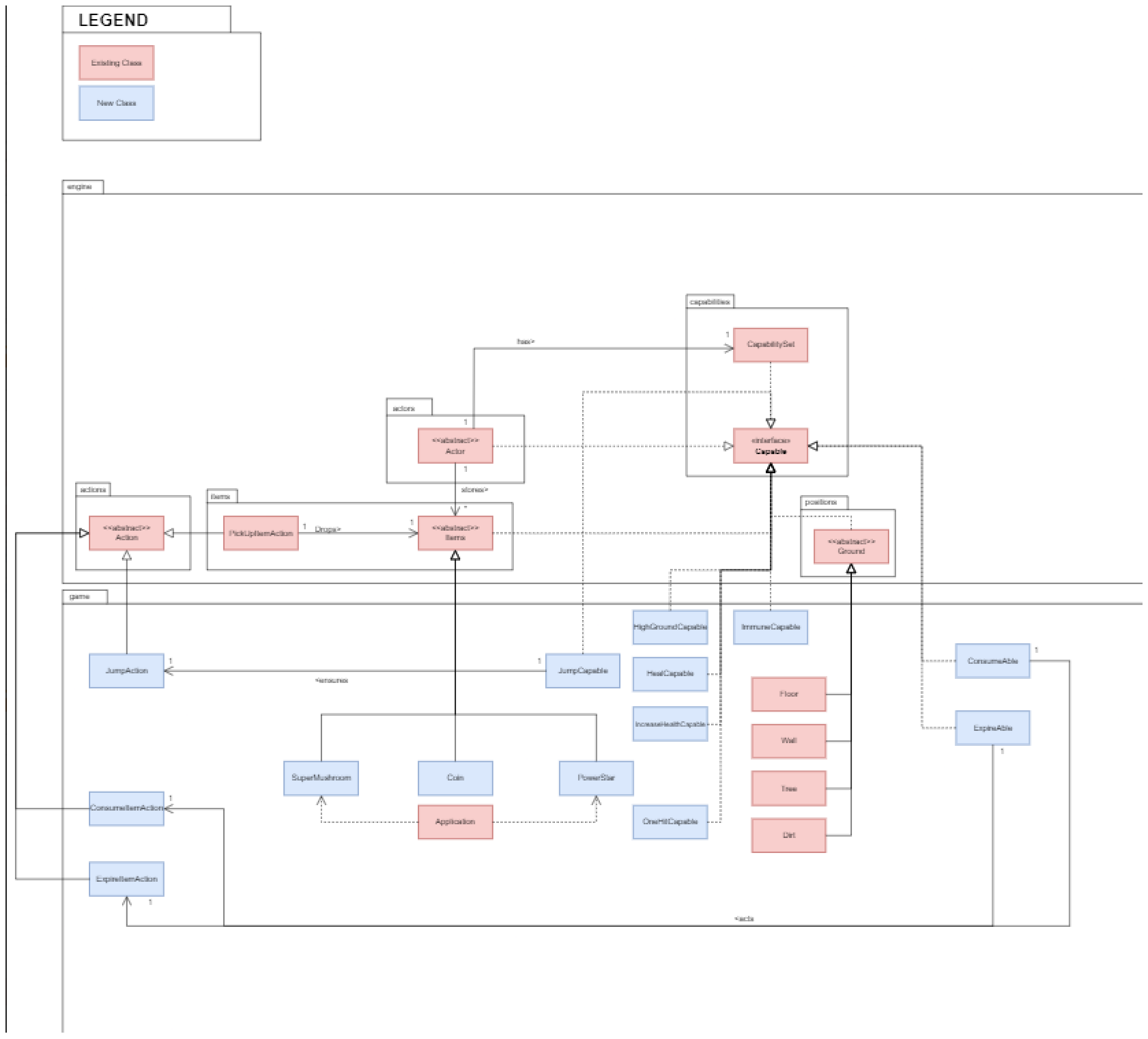
Goomba:

Goomba is a subclass of the Actor class as it shares several attributes. Any actor is spawned by the ActorLocationsIterator, since a Goomba will spawn from a Tree there is an implied dependency of Goomba to Tree. Goomba will implement several behaviours. First is the WanderBehaviour, which makes the actor wander to a random location. Second is the FollowBehaviour, when the player enters the Goombas proximity, it will move towards the player. Third is AttackBehaviour, when the player is within the range of the Goombas attack, then the Goomba will automatically try to attack the player. The attack can be implemented with AttackAction and its base damage will be given by the InstrinsicWeapon. Fourth, the DestructBehaviour is for giving it a 10% chance of being removed from the map every turn. The DestructBehaviour will use the DestructAction to remove the Goomba. The above has been implemented as behaviours since only NPCs will have the DestructBehaviour and AttackBehaviour, but not the player. In addition, adding these as behaviours adheres to the open close principle, making it easier to implement for future Actors.

Koopa:

Koopa is also a subclass of the Actor class, however when it is killed it is put into a dormant state. Koopa also has a WanderBehaviour, FollowBehaviour and AttackBehaviour like Goomba. However, the Koopa does not have a DestructBehaviour and instead has a DormantBehaviour. When in DormantBehaviour, it will stop the Wander, Attack and Follow Behaviour. A Koopa can be destroyed in its Dormant State by an Actor that uses a Wrench. The Wrench adds the capability of DestroyKoopaAble and that adds the Action DestroyKoopaAction. Doing this will remove the Koopa from the map and drop a SuperMushroom.

Requirement 4:



Design Rationale:

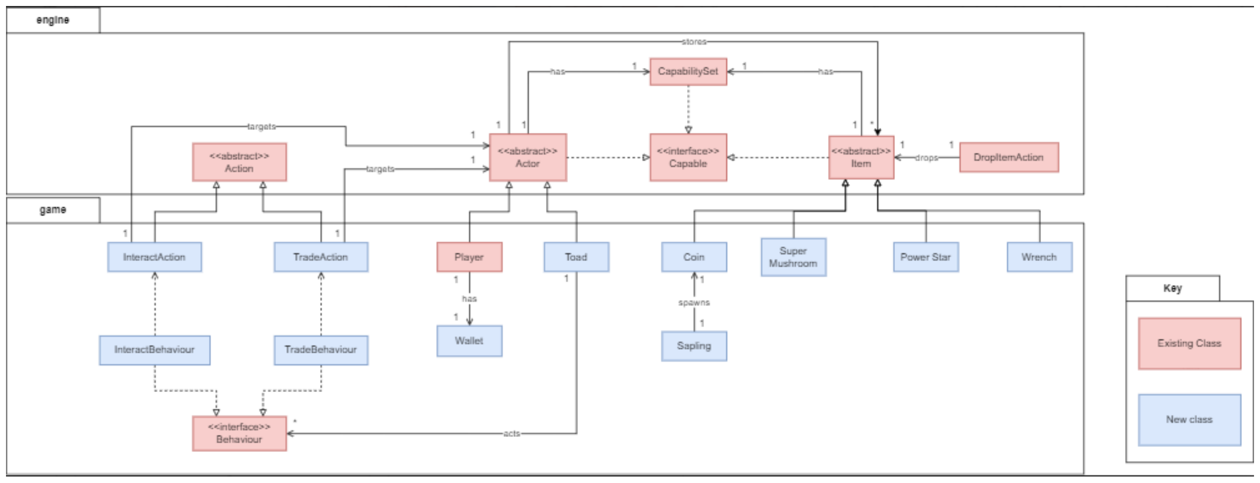
Super Mushroom:

An actor can pick up a SuperMushroom by using PickUpItemAction. If an Actor picks up an Item that implements ConsumeAble, then the Actor can use ConsumeItemAction to eat the Item. Eating a SuperMushroom will increase the Actor's HP for a specific amount, therefore is IncreaseHealthCapable. The SuperMushroom gives the enum status TALL and JumpCapable allowing the Actor to jump over high grounds.

Power Star:

A PowerStar is an expirable Item therefore implements ExpireAble, which makes it disappear within 10 turns of it being spawned. Eating a PowerStar will grant the user the STAR status. This gives the user HighGroundCapable, which makes the Actor able to walk on high ground and turn them into coins. The Actor also gains ImmuneCapable, making any incoming damage zero. The Actor will also kill enemies instantly so it has a OneHitCapable. These powerups have been separated to follow the Interface Segregation Principle

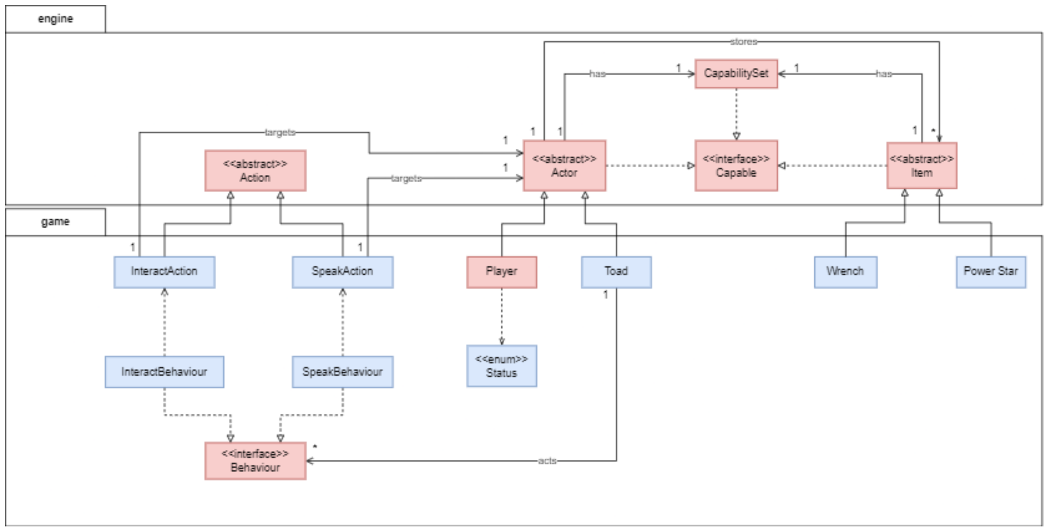
Requirement 5:



Design Rationale:

Toad will be an actor that can not move from the middle of the map, holding an unlimited amount of items to trade with. An interact behaviour and action must be implemented first so that the player is able to interact with Toad. A trade behaviour will need to be checked if the trade transaction is able to be executed. Depending on the value of the player's wallet, the specific item wanted could be transacted. If the behaviour says that the trade can be executed, then the trade action will be processed through. Or else, the transaction will be canceled and a sentence will be printed stating that the player does not have the required amount of coins. Coins may also be spawned from Saplings and destroyed high grounds so the dropitem class must be implemented for this action to occur. The interact behaviour and the trade behaviour also
both follow the Dependency Inversion Principle by implementing the behaviour interface as the addition of other behaviours will follow the same principles.

Requirement 6:

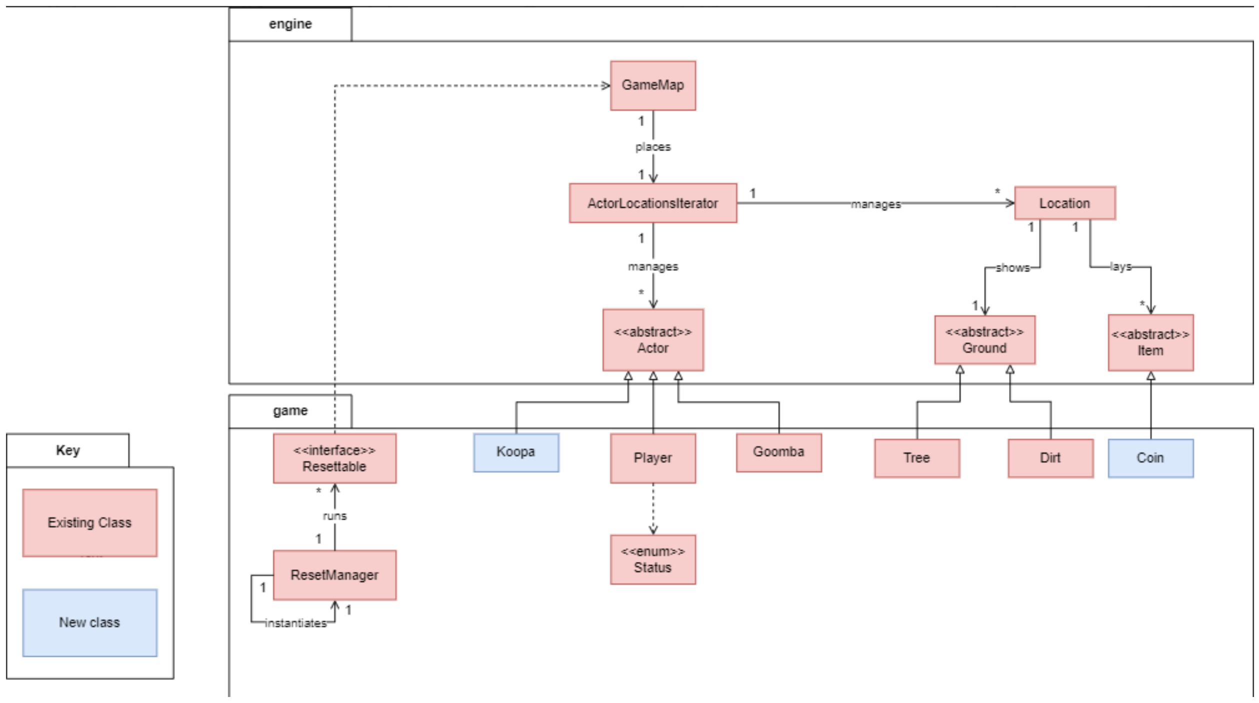


Key
Existing Class
New class

Design Rationale:

Similar to the behaviour in Req 5, Toad and the interact behaviour and action will be implemented. However, a new behaviour class will allow the player to have a possible speak action with Toad. The behaviour class will determine whether the speak action can be completed or not and which statements can be printed or not depending on the character's status and if he is holding a wrench. The addition of the trade behaviour also follow the Dependency Inversion Principle as it follows the same concepts within the behaviour interface.

Requirement 7:



Design Rationale:

The reset manager will be activated once the button r is pressed. To erase the coins and enemies, the locations of these items and enemies will be located, then they'll be erased from the location. These actions will be executed through existing function classes such as GameMap and Location class. The trees will follow the same concept, however, it'll have a 50% chance of not turning back into dirt. The player's status will also be reset back to none and the player's health will return to maximum. As it utilises the existing functions, The resettable interface follows the open-closed principle as it allows itself to be open for extension without modifications.

Work Breakdown Agreement

REQ 1 & 2:

Samuel Rath

REQ 3 & 4:

Daniel Ding

REQ 5, 6 & 7:

Felix Xu

Finish Date:

Saturday Night (around 7-8pm)

Meeting on:

Sunday Evening (around 8pm)

Reviewing:

Daniel Ding, Felix Xu,Samuel Rath

WBA Acceptance:

Daniel Ding: I accept this WBA

Felix Xu: I accept this WBA

Samuel Rath: I accept this WBA