Work Breakdown Agreement (Assignment 1)

This Work Breakdown Agreement is for Team 4 in lab 14. Team members include Devireddy Prasanth Kumar Reddy, Rishi Bidani and Jack Robert Draganich. The table below will outline which team member is responsible for which deliverable, the team member who will test and review each deliverable, and the dates by which the deliverables are to be ready. All deliverables are expected to be ready by 9/4/22 to provide the team with enough time to ensure that everything is in order. All team members should help out one another when the need arises.

Team Member	Deliverable	Member in charge of review	Due date
Prasanth	UML class diagrams and Design rationale for REQ1-REQ3.	Rishi, Jack	8/4/22
Rishi	UML class diagrams and Design rationale for REQ4 & REQ5. Sequence Diagram for Tree functionality (REQ1)	Prasanth, Jack	8/4/22
Jack	UML class diagrams and Design rationale for REQ6 & REQ7. Sequence diagram for the Trading functionality (REQ5)	Rishi, Prasanth	9/4/22

Prasanth: I accept this WBA
Rishi : I accept this WBA
Jack: I accept this WBA

Design Rationale

The Application class (which is the main class for the Mario World game) is where most of the interaction of the classes will take place

In the UML class diagrams, the existing classes or modified classes are represented with blue colour, the new classes in the system are represented with green, and the packages are in yellow colour.

REQ1:
Tree is an abstract class which contains methods that Sprout, Sapling, and Mature classes inherit. In other words, Sprout, Sapling and Mature extends the Tree abstract class.All these classes will be in the Tree package. The Application class would have methods to spawn koopa or goomba at the respective coordinates by taking instances of the tree and identifying the right type.Tree has access to coordinates using the location class (application class has access to location through this)

REQ2:
Jump is an interface that is used by the Player and Enemy class. These classes then interact with the Application class to get the location of objects to jump over. Player will have an inventory (a hashmap of magical items, coins and other accessories) and if player accesses the super mushroom, they can jump freely

REQ3:
Enemy will be an abstract class that is extended by Goomba and Koopa. The Enemy class is extended by the Actor class.The Application class will have an arraylist of all enemies and depending on the type of enemy, the player will be able to perform the necessary actions

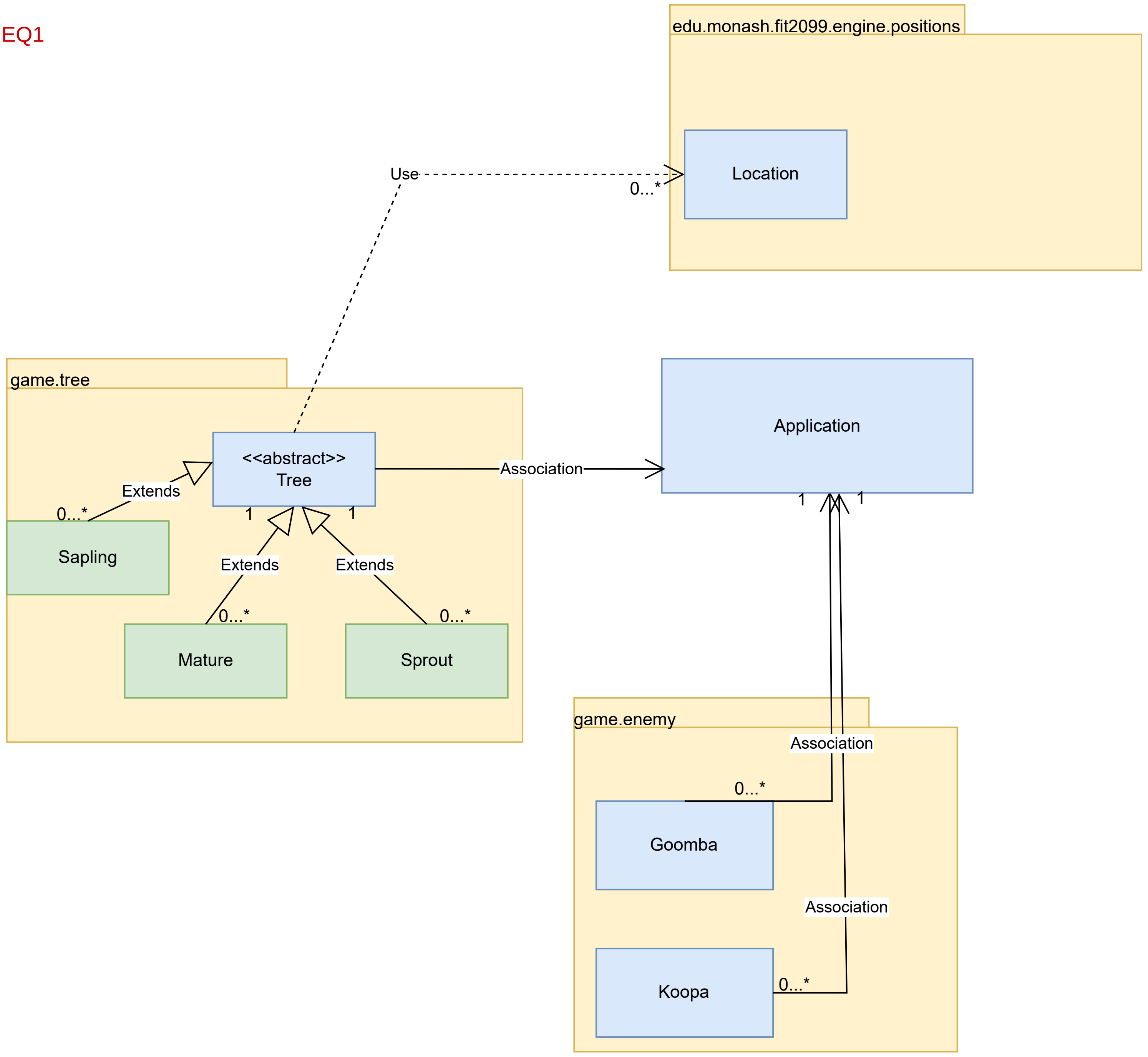
REQ4:
MagicalItems is an abstract class that is extended by PowerStar and SuperMushroom. The magical items are stored in a list in the application class. When an instance of player is created, the power star and super mushroom are placed on the same ground. The player has an inventory to store such items and the application class helps modify this inventory.

REQ5:
Toad class would have a method called trading and a parameter of what to buy. This would call Application class to update the inventory of Player .Inventory (which is a hashmap) would also keep track of how many coins the player has.

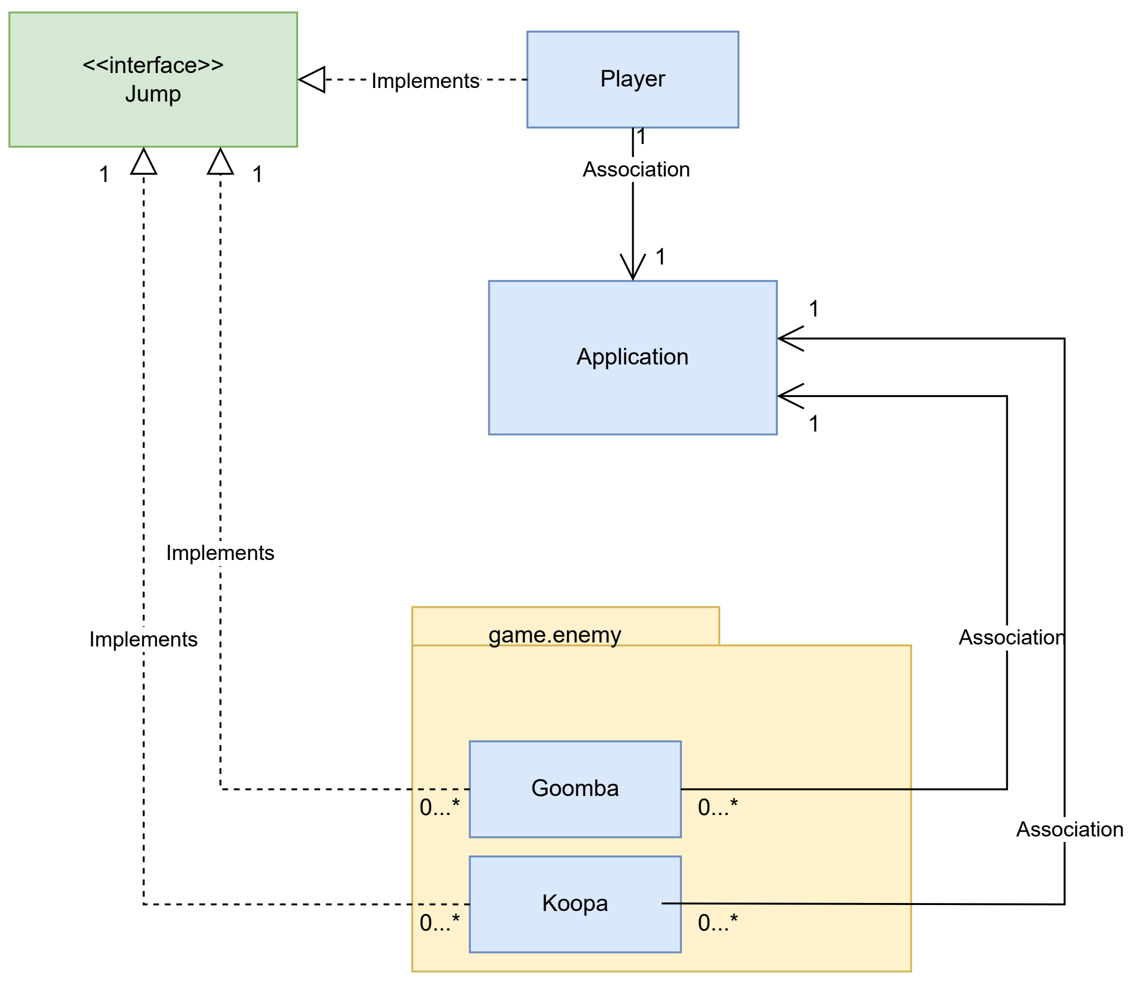
REQ6:
Monologue is also another method inside of Toad similarly to the trading method. This is because we just need Toad's and Player's location to check whether they can interact and then just run a couple of checking statements to determine what dialogue Toad can produce.

REQ7:
Reset game can be a method inside of the application function as it needs access to many different variables and classes. When the user hits 'r', the method can simply be called and the game can be reset.

REQ1

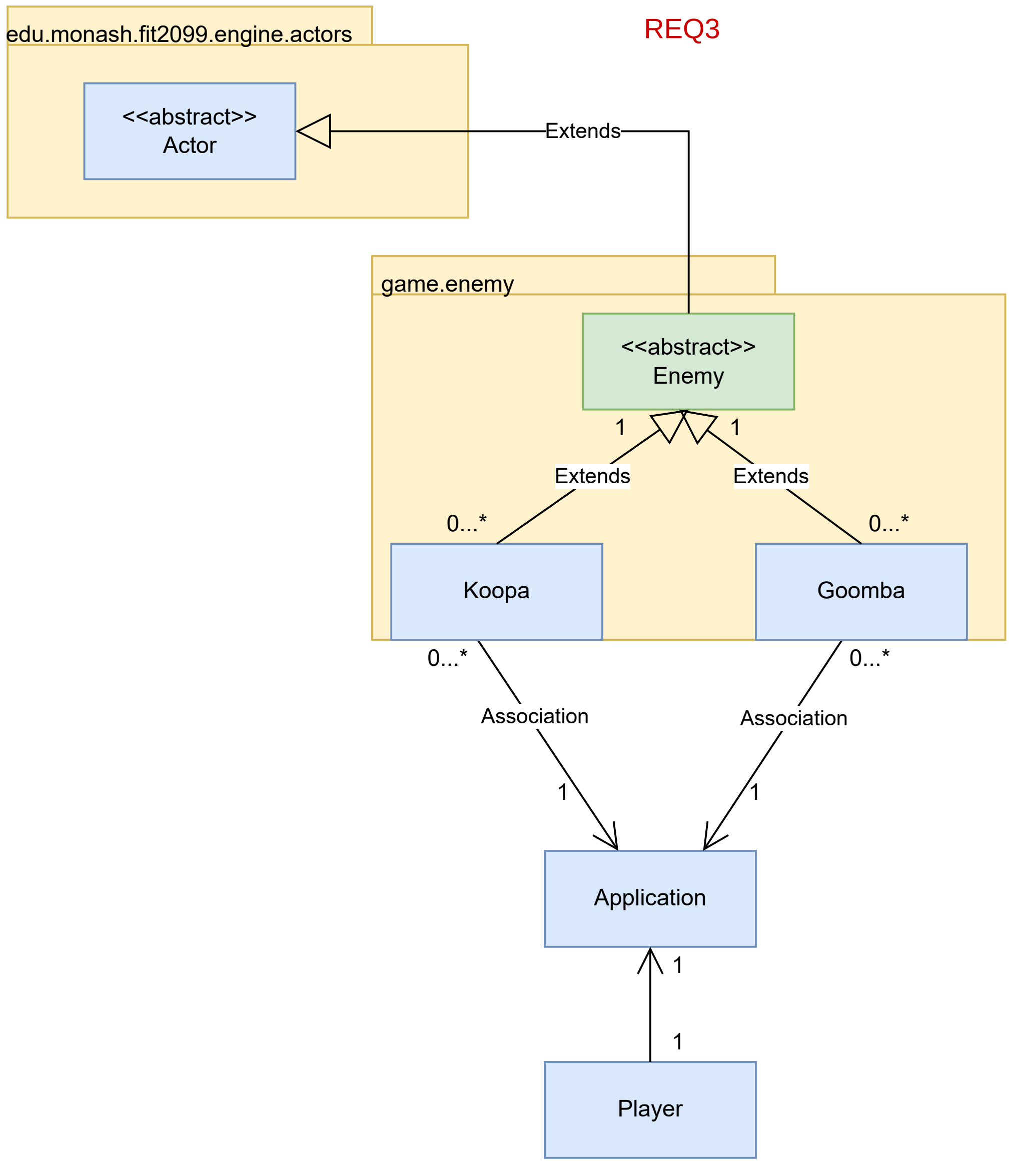

edu.monash.fit2099.engine.positions
Location
Use
0...*
game.tree
<<abstract>>
Tree
Extends
0...*
Sapling
1
1
Extends
Extends
0...*
0...*
Mature
Sprout
Association
Application
1
1
game.enemy
Association
0...*
Goomba
Association
Koopa
0...*

REQ2

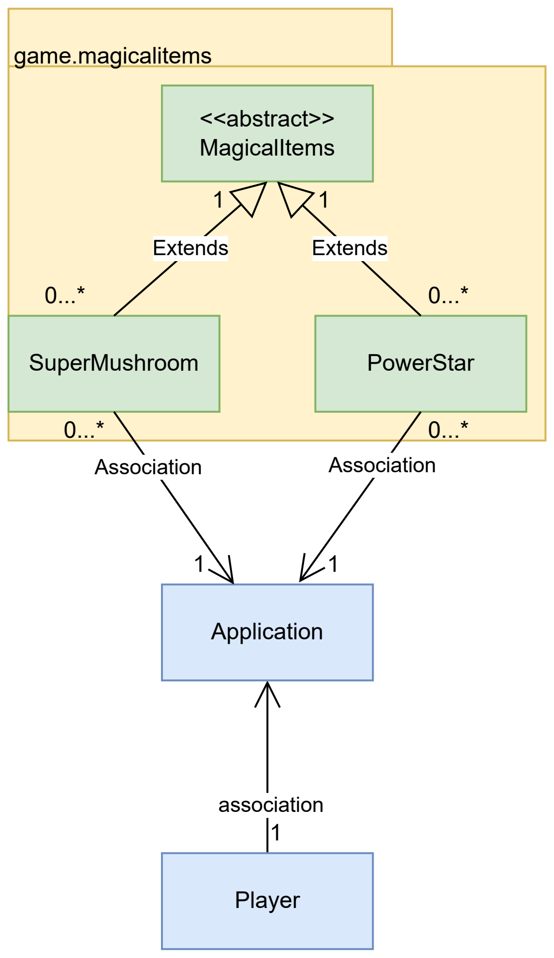


<<interface>>
Jump
Implements
Player
1
Association
1
Application
1
1
1
1
Implements
Implements
Association
game.enemy
Goomba
0...*
0...*
Association
Koopa
0...*
0...*

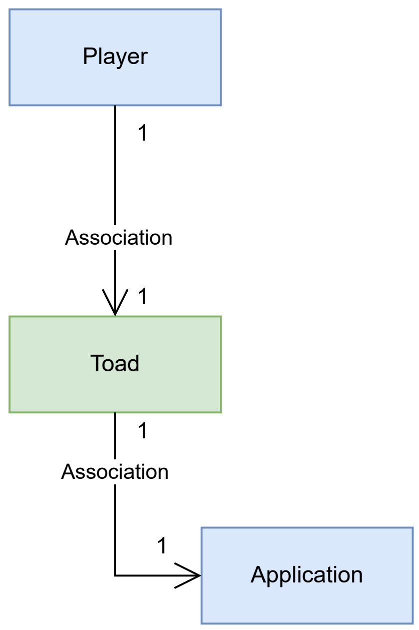
REQ3


edu.monash.fit2099.engine.actors
<<abstract>> Actor
Extends
game.enemy
<<abstract>> Enemy
1
1
Extends
Extends
0...*
0...*
Koopa
Goomba
0...*
0...*
Association
Association
1
1
Application
1
1
Player

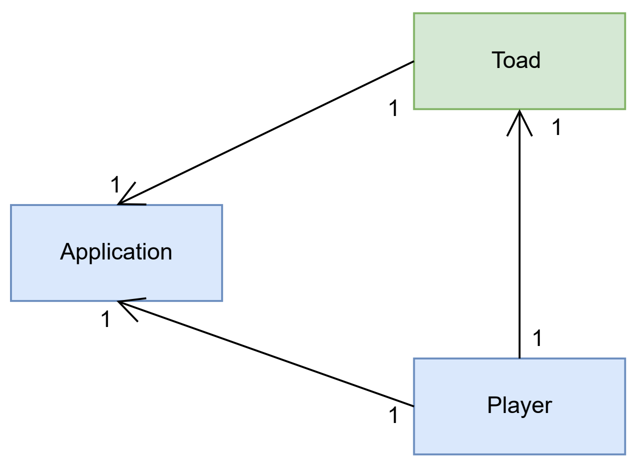
REQ4


game.magicalitems
<<abstract>>
MagicalItems
1
1
Extends
Extends
0...*
0...*
SuperMushroom
PowerStar
0...*
0...*
Association
Association
1
1
Application
association
1
Player

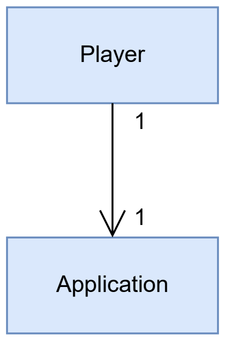
REQ5

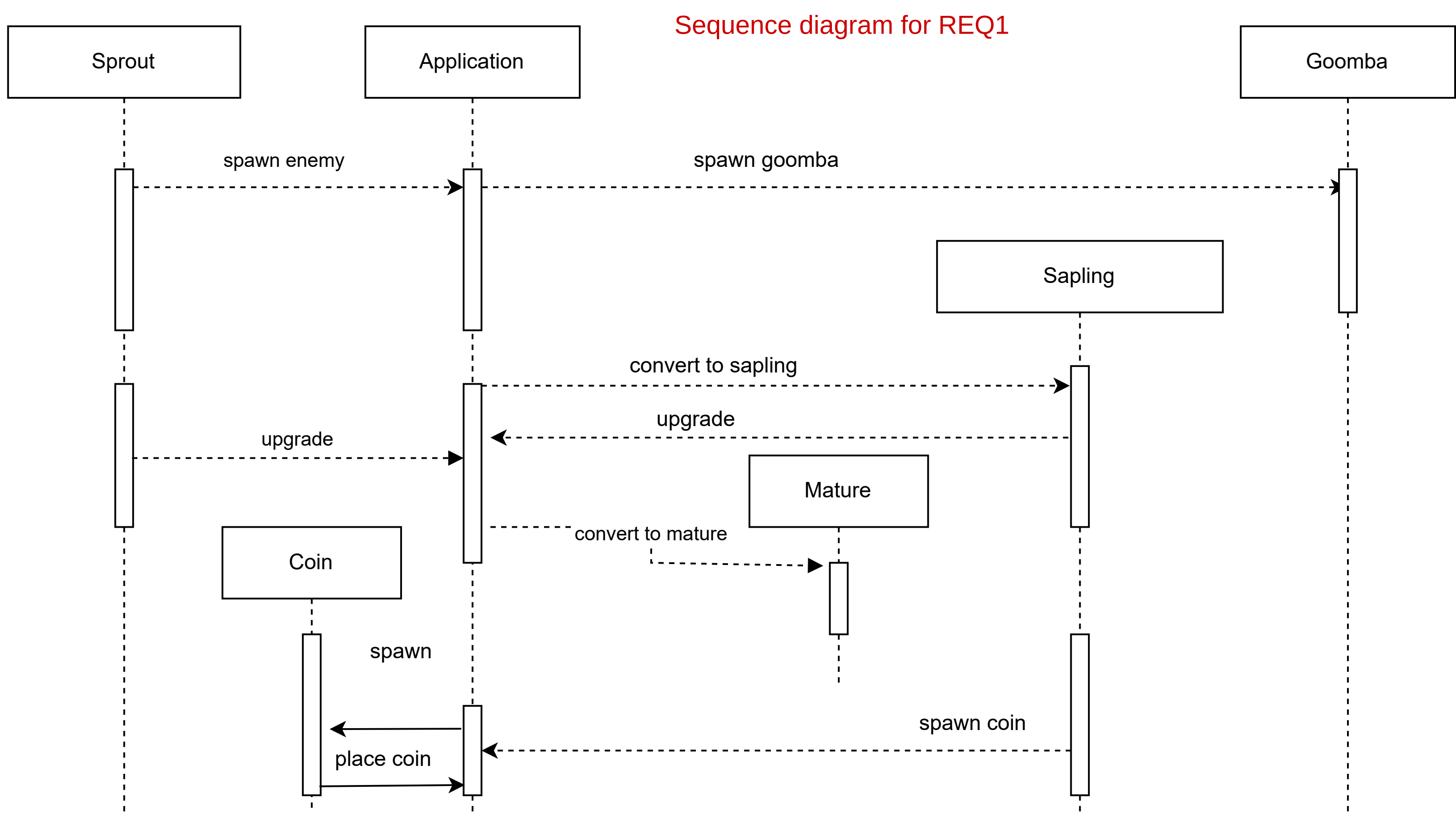


REQ6

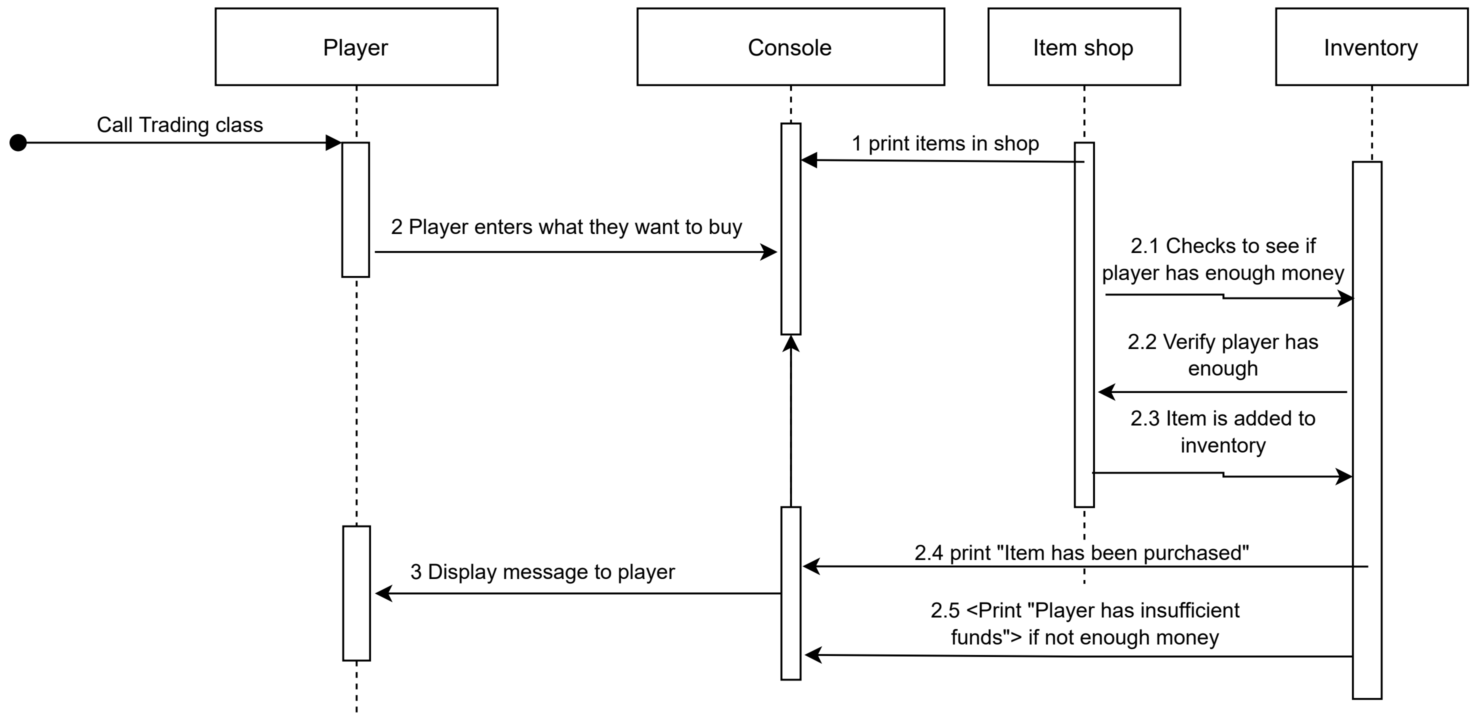


REQ7




Sequence diagram for REQ1
Sprout
Application
Goomba
spawn enemy
spawn goomba
Sapling
convert to sapling
upgrade
upgrade
Mature
convert to mature
Coin
spawn
spawn coin
place coin

Sequence diagram for REQ5



Player
Console
Item shop
Inventory
Call Trading class
1 print items in shop
2 Player enters what they want to buy
2.1 Checks to see if player has enough money
2.2 Verify player has enough
2.3 Item is added to inventory
2.4 print "Item has been purchased"
3 Display message to player
2.5 <Print "Player has insufficient funds"> if not enough money