

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.