Design Report for Assignment 3

REQ1

A smaller map called Lava map is created and is added to the gameMap and World through Application. Multiple blazing fire grounds called lava have been placed in this secondary map which hurt the player per turn.

WarpPipe is a concrete class that extends from Items abstract class in the engine. WarpPipe is a non portable item that is placed on the ground that allows the player to teleport to the other map.

Multiple warp pipes are scattered throughout the main map. but only one warp pipe is placed in lava map as per the requirements. The warp pipes in the main map spawn a piranha plant on their location in the second iteration of the game. In the playTurn method inside Player, we check if the item on the ground (Warp pipe) has a teleport status and perform the teleport action from the TeleportAction class. We have made a TeleportAction class as it is a type of action and having this class allows us to manage this feature better.

REQ2

Bowser and Piranha Plant are enemies that inherit from the Enemy abstract class. Princess Peach is an ally that extends from Actor class. Application uses both Bowser and Princess Peach and places them on the map. This establishes a dependency between Application and the other classes mentioned. Each of these classes inherit from Actor either directly or indirectly. The reason for this is because these characters are actors even if they are not playable.

We have made an AbstractKoopa class that extends from the abstract Enemy class and FlyingKoopa in turn extends from AbstractKoopa. This is because we have multiple Koopa types (Normal and Flyable) and this also allows us to extend the functionality of the respective Koopa for further implementation and extension without violating SOLID principles.

REQ3

This requirement takes advantage of using the capabilities class to easily check to see what the player can and cannot do with the bottle. I did this to avoid having to have many check statements to see what attributes the fountain has. I can assign the Status POWERFOUNTAIN or HEALTHFOUNTAIN to each fountain based on the character passed in the constructor on intitiation. This allows me to add another fountain by just adding another capability and therefor does not violate any SOLID principles.

I created an Enum for the bottle to keep track of what water is stored inside each bottle. By doing this, I can simply change the enum of the object Bottle in the player's inventory with a single line. I chose to do this because it means I can do this based on the attributes the fountain has instead of having to use a bunch of if statements.

REQ4

FireFlower is a concrete class that is extended from the MagicalItems abstract class. This decision was made as the user can consume this item and gain special abilities. The sprout and sapling classes spawns the fire flower in their growing phases hence they have a dependency with the FireFlower class.

The ConsumeItemAction class which extends the Action class takes in a parameter of the fire flower and does the consume action and assigns a FIRE_ATTACK capability to Player. Using capabilities helps us avoid using the instanceof feature which usually is a sign of poor design. The FireAttackAction class checks if the player has the capability and performs the fire attack on enemies.

REQ5

Each of the classes with a speaking capability implement a Speak interface which override a speak method in it.

By using an interface we have the flexibility of implementing any functionality in the the method while instantly knowing which classes have the speaking capability.

Furthermore, the reason why we didn't make Speak a class, is because it does not have enough functionality to implement, and hence does not require a dedicated class.