

## Design rationale

### Player Class

This class represents players in the game. Player class extends Actor class. It is associated with the Vendor class to allow players to purchase magical items. Player class uses PickItemsAction ( which extends the Action class) to pick up magical items, which are Super Mushroom and Power Star; and use PickCoinAction to pick up coins.

### PickItemAction class

This class extends the Action class to allow players to pick up Super Mushroom and Power Star

Description of method:

- updateStatus method to update the player's status according to effects of magical items collected

Why i choose to do it that way:

Since PickItemAction is an action, functions in the action class are required and hence need to extend the action class.

Advantage:

Using this design, the Single Responsibility principle can be implemented as this class will only be used when used to pick up magical items ( which this class will then modify the status of players depending on which item was picked up), hence higher cohesion.

Disadvantage:

N/A

### PickCoinAction

This class extends the Action class to allow player to pick up coins and then add the collected coins into the users wallet

Description of method:

- addToWallet method adds the amount of coins collected by player into the player's wallet if player reaches location that contains coins

Why i choose to do it that way:

Since PickCoinAction is an action, functions in the action class are required and hence need to extend the action class.

Advantage:

Using this design, the Single Responsibility principle can be implemented as this class will only be used when used to pick up coins (which this class will then be used to increase the player's wallet balance) hence higher cohesion.

Disadvantage:

N/A

### **PowerStar class**

This class is a subclass of items. Players that consume it will be healed by 200 hit points (hp) and become invincible. The invincible effect replaces fading duration (aka, fading turn's ticker stops), and it lasts for another 10 turns. It fades and disappears from the game within 10 turns.

Description of method:

- `updateStatus` method sets the player's status such that player does not need to jump to higher level ground, add 5 coins into player's wallet for every destroyed ground, make player immune to damage and enable player to attack enemy successfully

Description of attributes:

- `HEALED_HIT_POINTS` is a public static integer attribute with value of 200 that indicates the hit points players can get healed by after consuming the Power Star

Why i choose to do it that way:

Since Power Star is an item, functions in the `Item` class are required and hence need to extend the `Item` class. I created this class in order to store information about this item, such as the effect it can bring to players and its ability to be traded for coins

Advantage:

This class is created using the Separation of Concern principle where the program is separated into sections with its own responsibilities, by having well- defined concerns and as little overlapping as possible. Which in this class, the class is only responsible for storing information about the Power Star

Disadvantage:

N/A

### **SuperMushroom class**

This class is a subclass of items. The effect will last until it receives any damage (e.g., hit by the enemy). Once the effect wears off, the display character returns to normal (lowercase), but the maximum HP stays.

Description of method:

- `updateStatus` method set the player's status such that:
  - the display character evolves to the uppercase letter (e.g., from m to M).
  - it can jump freely with a 100% success rate and no fall damage.
  - increase max HP by 50

Why i choose to do it that way:

Since Super Mushroom is an item, functions in the `Item` class are required and hence need to extend the `Item` class. I created this class in order to store information about this item, such as the effect it can bring to players.

Advantage:

This class is created using the Separation of Concern principle where the program is separated into sections with its own responsibilities, by having well- defined concerns and as little overlapping as possible. Which in this class, the class is only responsible for storing information about the Super Mushroom

Disadvantage:

N/A

### **Coin class**

This class is a subclass of Items. It is currency the player uses to trade for magical items. A coin has an integer value that determines the actual value of the money. Coins will spawn randomly from the Sapling (τ). The collected coins can be traded with Toad (Vendor class) for Wrench, Power Star and Super Mushroom.

Description of method:

- addToWallet method adds the amount of coins collected by player into the player's wallet if player reaches location that contains coins

Why i choose to do it that way:

Since Coin is an item, functions in the Item class are required and hence need to extend the Item class. I created this class in order to store information about this item, such as the ability for it to be traded for magical items and increase the player's wallet balance upon collection.

Advantage:

This class is created using the Separation of Concern principle where the program is separated into sections with its own responsibilities, by having well- defined concerns and as little overlapping as possible. In this class, the class is only responsible for storing information about the Coin and updating the player's wallet balance upon collection.

Disadvantage:

N/A

### **Wallet class**

This class is used to store and keep track of the amount of coins players have.

Description of method:

- getWalletBalance method returns the current amount of coins in player's wallet

Why i choose to do it that way:

I created this class to keep track of the amount of coins players have and implement a getWalletBalance method to check if players balance is sufficient to purchase magical items.

Advantage:

This class is created using the Single Responsibility Principle where it does not need to take extra responsibility. In case of need to change responsibility, all pieces needed will be there. This makes the system easier to maintain and extend. Therefore, this class is only responsible to constantly being update about player's coin amount and return the amount of coins players have if needed

Disadvantage:

N/A

**SwapAction class**

This class extends the action class and it is used when players trade Power Star for coins.

Description of method:

- convertToCoins method takes the Power Star picked up by player and add coins into the player's wallet

Why i choose to do it that way:

Since SwapAction is an action, functions in the action class are required and hence need to extend the action class.

Advantage:

Using this design, the Dependency Inversion Principle can be applied, that is, if there are changes in items involved for exchange, methods in this class do not need to be modified. As long as the interface remains the same, the code in this class that depends on it remains independent.

Disadvantage:

N/A