**REQ4**

Class Diagram:

Diagram

Description automatically generated

Requirement:

* Both magical items, Super Mushroom and Power Star would buff actor once consumed
* Super Mushroom increases actor’s max HP by 50, mario’s display character turns to M and actor can jump freely with 100% success rate and receive no fall damage
  + Super Mushroom buff wears off if actor gets damaged but max HP stays
* Power Star grants actors an invincible effect where it heals the actor by 200 HP and allows the actor to not need to jump to high grounds and can walk normally. Power Star also grants actor immunity and actors are also able to instantly kill enemies.
  + High grounds being stepped on would be converted to Dirt and a coin with $5 would be dropped
  + PowerStar would fade after 10 turns if not consumed, and if consumed, Power Star’s invincible effect would only last for 10 turns.

Design Rationale:

For Requirement 4, we would be required to create and implement the only two Magical Itemswe would have so far, which are **Super Mushroom** and **Power Star**, thus, everything covered in this requirement would be related allowing players to consume these magical items as well as buffing the players according to which magical item consumed and also the after effects after consuming the magical items. Since both Super Mushroom and Power Star are able to be consumed by the player to have some buffs to the player, we created an abstract class **ConsumableItem class** that extends **Item** class and this abstract class would have an abstract method called consumeItem, which defines how the actor would be buffed upon consuming the magical item. **SuperMushroom** and **PowerStar** class would both extend ConsumableItem and need to override consumeItem to add in how the actor would be buffed.

We would now first talk about how we would implement the buffs for both SuperMushroom and PowerStar. Upon consumption for **SuperMushroom**, the actor’s maximum hit points would be increased by 50 and the actor would add a capability Status.SUPER. By having this capability Status.SUPER, we can make the display character of mario to turn from m to M. This can be easily done in player’s getDisplayChar() where we just check if the player has capability Status.SUPER, if yes, return an uppercase version of player’s initial display char(which is m), if not, just return the initial display char. In addition to having this capability, we can also implement it so that the player can jump freely onto **HighGround** objects with a 100% success rate and no fall damage, where inside our defined **JumpAction** in REQ2, it would check if actor has capability Status.SUPER, if yes, we would just directly move the actor the respective location and not hurt the actor, so that the actor will always jump successfully and receive no fall damage when actor has capability Status.SUPER. Other than that, we also need to make sure that mario loses the Super Mushroom buff if it ever gets damaged by an enemy, hence we would need to check in AttackAction where if target has capability Status.SUPER, remove that capability so mario loses the buff and gets converted back to its original display char. By having these implementations, the buffs of Super Mushroom for actors would be done as per the requirements. Also, I have noticed that the base code has Status.TALL which tells that the current instance has grown, and this Status.TALL is used in the player's class when getting its display character. I have decided to change Status.TALL to Status.SUPER as I feel like it's more meaningful as we would only have an uppercase letter M for the player's display character once we consumed a Super Mushroom.

For **PowerStar**, the actor who has consumed it would be healed by 200 hit points and be added a capability Status.INVINCIBLE by holding the PowerStar after consuming it. In other words, we implemented it in the way that after consuming Power Star, it would remain in the actor's inventory but it doesn’t give the ConsumeItemAction anymore, and PowerStar would be added the INVINCIBLE capability. By having this capability Status.INVINCIBLE, we can allow actors to not need to jump to higher level grounds anymore and just walk normally, while any higher level ground that is stepped on would be converted to dirt and an a coin of $5 value would be spawned at that ground’s location. We could of course add if and else statements in JumpAction to move the actor as well as set the ground as dirt and create a new coin if the actor has capability Status.INVINCIBLE, however, we took consideration regarding the **Single Responsibility Principle (SRP)** and we feel like if we implement it this way, JumpAction would be having more than one responsibility, which is essentially allowing actors to be able to jump on **High Ground** objects and also walk over High Ground objects, thus we feel like this implementation would violate **SRP**. Hence, we decided to create a new action for actors to walk over High Ground objects, which is called **InvincibleWalkAction**. In High Ground, we would override allowableActions and we would check if the actor has capability Status.INVINCIBLE, if yes, it would return the InvincibleWalkAction, if not, it would return the JumpAction. Under InvincibleWalkAction’s execute method, we would call our existing MoveActorAction method to move the actor to the chosen location. We would also set the ground of that location and set it to a new Dirt object. A new coin object of value $5 would also be created and added to that location. The reason why we called our existing MoveActorAction method is we wanted to follow the **DRY principle**, where we don’t repeat the code of moving the actor from one place to another, as it is already defined in the MoveActorAction and there is no point writing it twice.

Moving on, enemies would also deal no damage to the invincible actor (grants immunity) and the actor would be able to instantly kill any enemies if the actor doesn't miss (including Koopas that are in dormant state). In terms of the immunity buff upon consuming Power Star, we would check if the target has capability Status.INVINCIBLE, if yes, the damage dealt would be set to 0. For being able to instantly kill enemies if the actor does not miss, we would just need to check if the actor misses in the AttackAction, if it does not miss and the actor has capability Status.INVINCIBLE, the target is removed immediately from the map. We would also check if the target has capability Status.DORMANT or Status.NOT\_DORMANT, this is to check if the instantly killed actor is a Koopa or not, if yes we would also spawn a Super Mushroom at that target’s location. We would also override in Koopa’s allowableAction to return super.allowableActions when the actor has capability Status.INVINCIBLE. Hence, by having all these implementations, the buffs of PowerStar upon consumption would be done as per the requirements.

Moving on, we would talk about how we would implement the action to consume these magical items. We have two scenarios for the two magical items, the first scenario is when the magical items are on the ground, and we consume it when we stand on top of them, the second scenario is when we purchase magical items from Toad and the items are added into the mario’s inventory. However, regardless of which scenario we are facing, we could create an action called **ConsumeItemAction** which basically allows us to consume any magical items. Remember the consumeItem method that was required to be implemented by every subclass of ConsumableItem? This consumeItem method would be used in ConsumeItemAction’s execute method whereby we would call the consumeItem method of the magical item onto the actor to provide the buffs to the actor. For SuperMushroom, regardless if it is consumed when it is on the ground/in the actor's inventory, it would be removed from ground/inventory upon consumption. However for Power Star, if it is consumed when it is on the ground, Power Star would be added to the actor’s inventory as Power Star grants the actor the INVINCIBLE effect upon having it in inventory. If Power Star is consumed when it is in actor’s inventory, it would remain in the actor’s inventory. In both scenarios, Power Star would automatically remove itself when the invincible effect wears off. The reason why we implemented this ConsumeItemAction in this way is because if in the future we have many more magical items, we could still use this ConsumeItemAction to consume those magical items as those newly added magical items would just need to implement the consumeItem method which defines how they would buff/debuff the player once consumed. This follows the **Open-Closed Principle (OCP)** where we are not modifying our current system and we are just extending our system by adding in new magical items in the future. This also relates to the **Dependency Inversion Principle (DIP)** as our system would not depend on the concrete classes of magical items such as Super Mushroom and Power Star, instead it depends on abstraction where it just depends on the abstract class ConsumableItem. In a way, I would see that OCP and DIP are related as when our system doesn't depend on low-level modules and depends on abstraction, such systems that depend on abstraction are generally very easy to extend and would not modify the current system.

Last but not least, since PowerStar will fade and it would also have limited invincible effect, we could calculate these turns in PowerStar’s tick method. As mentioned earlier, consumable items have two scenarios, whether it can be on the ground or in the inventory of the player, hence we would need to override both tick method’s (one for when item on ground and another for when item in inventory). PowerStart would by default have 10 turns left to signify the turns left before it fades, each tick would then decrease then turns left by 1. Since this implementation only uses one turns left counter, if the actor ever consumes the Power Star, the turns left counter would be resetted such that the turns left for the invincible effect to wear off is 10. If the turns left counter ever reaches 0, the Power Star would immediately be removed regardless if it has been consumed, depending if it is on the ground or in the actor's inventory. Since we have decided to let PowerStar have the responsibility of keeping track of the turns left instead of letting Player handle it, this follows the **SRP** as the invincible effect came from Power Star. Furthermore, if Player were to handle it, Player would have too many responsibilities and this violates **SRP**. In addition, by leaving Power Star in the actor's inventory and letting Power Star handle the tracking of turns also reduces dependency between Power Star and Player.

Besides, inside the PowerStar method, we also would override the toString method so that it returns “Power Start (<turnsleft> turns left)”. The reason for doing this is so that it would be very convenient to get Power Star along with the turns left so that it could be easily used in ConsumeItemAction’s menuDescription method as when we print magicalItem, we would automatically get Power Star along with how many turns left to consume it if the magical item is a Power Star, and ultimately this would give us the right sentence in the menu for the user to know how many turns left is there for the user to consume power star. All in all, by implementing all these classes along with its definitions, we would be able to fulfil the requirements for Requirement 4 and the system would work.

 \*Updates that have been done for REQ4 (class diagram & design rationale)

1. Fixed extend arrow notation from SuperMushroom and PowerStar to ConsumableItem, changed from dashed to solid line
2. Added extend arrow notation from Coin to Item
3. Removed PickUpCoinAction and Wallet from this class diagram as it will be showed in REQ5
4. Changed naming of MagicalItem class to ConsumableItem class
5. The INVINCIBLE effect is granted to actors by holding the Power Star after consumption and not directly granted to actors. (Instead of doing actor.addCapability(Status.SUPER), we did this.addCapability(Status.SUPER) in PowerStar).
6. PowerStar handles the calculation of turns left for invincible effect instead of Player, this follows SRP.
7. Initially we would remove the ConsumableItems from inventory/ground immediately after consuming them, it is now still the same for SuperMushroom, however for Power Star, if it is initially on the ground and it is consumed, Power Star would be added into inventory. If it is in the actor's inventory, it would still remain in the actor's inventory. Power Star would then automatically remove itself from the actor's inventory after the invincible effect wears off.
8. There is no need to check if Power Star has been consumed before removing it when turns left is 0 anymore as holding the Power Star after consumption would grant the actor the INVINCIBLE capability.