Design Rationale (REQ 2)

Tree and Wall ---<<*creates*>>---> *JumpAction*, *Wall* ---<<*creates*>>---> *ConsumeAction* and *Player* ---<<*use*>>---> *CapabilitySet*. In this game, different objects such as trees, walls and super mushrooms attach different actions that actor can perform on them. For example, trees and walls are high ground, they allow the actor, which is the player, to jump onto it. Based on object-oriented, the tree and wall are the objects. We can simply create JumpAction inside the Player class, however by doing this, we need to know which object the player wants to jump, it will require additional dependency between Player and Tree and Wall.

Besides, we also need to check whether the ground allows the player to jump onto it or not, checking the object classes using the if-else statement will also increase dependency. To align our design with the Reduce Dependency Principle, we discard this alternative and use a different approach that is shown in the class diagram above. The wall and tree <<creates>> jumpAction, which has the similar concept of Item class that <<creates>> pickUpAction that has been implemented inside the code given.

Different stages of Tree and Wall have different success rates that Player can jump onto it and if the player fails to jump onto it, they will cause different amounts of damage on the player. Based on the open-closed principle, classes should be open for extension but closed for modification. This is the rationale behind the presence of the FallDamage interface. By doing this, the classes implement it will have same method but different implementation, even there is another high ground will be added into our game in future, we can simply extend it from the interface created above