Design Rationale REQ 5

<<interface>> Tradable implemented by MagicalItem and Wrench. To follow open-closed principle, the MagicalItem and Wrench classes have share same method but different implementation. By using an interface as an abstraction, it can open for extension of same method but different implementation.

MagicalItem---<<create>>---> BuyAction and Wrench---<<create>>---> BuyAction. The MagicalItem and Wrench are the objects that give Actor(Player) an action to buy. . We can simply create BuyAction inside the Player class, however by doing this, we need to know which object the player wants to buy, it will require additional dependency between Player and MagicalItem and Wrench. Besides, we also need to check whether the object allows player to buy or not, checking the object classes using if-else statement will also increase dependency. To align our design with Reduce Dependency Principle, we discard this alternative and use a different approach that shown in the class diagram above.

Coin has the Status.Spend. It will be added into its capability set that will be used when player want to buy item.