DESIGN RATIONALE

The overall design choices are made keeping in mind the extensibility of the game. The details of the design choices for each requirement are as follows-

Requirement 1: Player and Estus Flask

Design Choices:

1. **Display Health/Hit Points**
   To display the health/ hit points of the Player, we can implement code in the pre-existing playTurn method of the Player class so there is no need for a new class. The attribute of hit points is already found in the Actor class so a new health system doesn't need to be implemented. We only need to display it. This can be done by calling the methods of the Display class.

2. **Hold Estus Flask:**
   The player needs to hold an Estus Flask, for this feature, we can create a new EstusFlask class that extends the Item class with private attribute quantity. We need to extend the Item class so that we can use it as an item and print out its attribute(number of charges) in the console.

3. **Cannot Drop Estus Flask:**
   The player cannot drop the Estus Flask so we can override the pre-existing DropItemAction method to return null. We need to override the DropItemAction so that the player is not allowed to drop the Estus Flask. This will be a good design choice as we can make use of the pre-existing system to fulfill this requirement.

4. **Drink Estus Flask**
   The player should be able to drink the Estus Flask, so we can create a new class called DrinkEstusFlaskAction that we will put as an allowable action for the EstusFlask class. We need to create this class so that we can use the execute method of the Action class to access the player's instance and add the health of the player.

5. **Display Charges of Estus Flask**
   To display the number of charges of the Estus Flask in the console, we can implement code in the pre-existing MenuDescription method of the Action class. This will be a good design choice as the pre-existing system is sufficient to fulfill this requirement.