Design Rationale for REQ1: The Player

Introduction

In this section, I will provide a detailed explanation of the design and implementation of the "Player" Actor and its weapon "Broadsword".

Design Implementation

1. Creating the Player class

    The player class is inherited from the actor providing similar behavior and room for customization. Inheriting from the actor class allows to reuse the code, by overriding and extending the player class gets tailored. The player is given its hit points, display character and name as parameters and passed on to the actor class.

    The gameplay is turn based allowing sequential activities to be carried out, with each turn the player gains 1% of its maximum stamina, this is implemented by the recover stamina method. In each turn the health status and stamina status are displayed.

    This class has a great flexibility with extending also is easier to maintain.

2. Creating the Broadsword class

    This class extends the WeaponItem class taking its basic functionalities and then tailoring it using the parameters to the constructor. This way this is treated as another weapon with the existing code in the WeaponItem class. The parameters that are passed into the parent class is the name, display character, damage, hit rate and verb.

    The tick method takes every turn of the weapon used, in case the skill is used it decreases the number of turns, and if 0 turns left the skill is deactivated.

3. Adding the focus skill

    Starting with the broadsword class two methods are introduced to activate and deactivate the skill, while the related other methods update the damage multiplier according to the skill being activated. This skill can be activated whether the previous still has turns, in this case increase damage multiplier is used with using update damage multiplier, when activating the skill.

4. Creating the Activate Focus Action class

    This class is responsible for the action when using the Focus skill, within in this class the focus skill is activated and then the stamina reduction is done, also this increases the damage multiplier of the weapon broadsword.

Summary

In summary, the detailed design rationale encompasses various object-oriented principles, including encapsulation, the SRP, the Open-Closed Principle, the Interface Segregation Principle, and modular and flexible design practices. The implementation details considered in each design decision ensure code maintainability, code reusability, and overall system flexibility.