The SOLID principles apply

1. The principle of Single Responsibility (SRP)

   Each class has a single responsibility. For example:

   - Player.java manages the attributes and actions of the player.
   - CombatManager.java controls the logic of combat.
   - LevelManager.java controls the passage of the level.
   - ScoreManager.java tracks and updates the player's scores.

2. The principle of openness/closeness (OCP)

   - The enemy system is designed to be extensible. New types of enemies (e.g. vampires, skeletons) can be added by implementing IEnemy without changing the existing code.
   - Elements such as HealthElixir and MagicScroll use a similar extensible approach by implementing IItem.

3. The Liskov Substitution Principle (LSP)

   - All enemies (zombies, vampires, skeletons) are inherited from Enemy.java and correctly redefine methods from the IEnemy interface.
   - The items implement the IItem function so that they can be used interchangeably in the player's inventory.

4. The principle of interface Separation (ISP)

   - The IEnemy interface defines only the basic methods related to combat.
   - The IItem interface is only responsible for using items, preventing unnecessary dependencies.

5. The principle of dependency inversion (DIP)

   - CombatManager does not depend on specific enemy implementations; instead, it interacts with IEnemy, making it easier to add new types of enemies.
   - The Player class does not directly reference specific item types, but works with IItem, providing flexibility.