Applying SOLID Principles

Single Responsibility Principle (SRP) - Each class has a single responsibility:

Player manages player attributes and actions.
Enemy represents different enemy types.
Item handles various in-game items.
LevelManager is responsible for managing game levels.
Combat deals with battle logic.
ScoreManager keeps track of the player's score.
This separation ensures that each class focuses on a single function, making the code easier to maintain.

Open/Closed Principle (OCP) - The design allows for easy extension without modifying existing code:

New enemy types can be added by extending the Enemy class.
Additional items can be introduced by implementing new subclasses of Item.
The scoring system can be modified without altering the core game logic.

Liskov Substitution Principle (LSP) - All derived classes (Skeleton, Zombie, Vampire) can replace their parent class (Enemy) without breaking functionality. The same applies to item classes derived from Item.

Interface Segregation Principle (ISP) - Instead of creating large interfaces with unrelated methods, smaller and more specific interfaces are used where necessary, ensuring that classes only implement what they need.

Dependency Inversion Principle (DIP) - High-level modules (Game) do not depend on low-level implementations (Enemy, Item); instead, they depend on abstractions. This decouples components and makes it easier to extend and modify the game.

By following these SOLID principles, the game is now modular, extendable, and easier to maintain.