1. Single Responsibility Principle (SRP) — The principle of sole responsibility

- Each class now performs only one task.:
- The player is responsible for the actions and state of the player.
- Enemy and its subclasses (Goblin, Orc) define enemies and their behavior.
- Battle controls the mechanics of battles.
- LevelManager generates enemies for each level.
- ItemManager is responsible for creating items, and HealthPotion is responsible for applying them.
- This makes the code easier to maintain and extend.

2. Open/Closed Principle (OCP) — The principle of openness/closeness

- The code is now open for expansion, but closed for modification.:
- To add a new enemy, it is enough to create a new class that implements Enemy, without changing the existing code.
- New items can be added by extending the Item interface, without modifying the ItemManager.

3. Liskov Substitution Principle (LSP) — Barbara Liskov Substitution Principle

- All Enemy subclasses (Goblin, Orc) can replace the parent interface without changing the program logic.
- The player interacts with them through the common Enemy interface, which makes the code universal.

4. Interface Segregation Principle (ISP) — The principle of interface separation

- Instead of one cumbersome interface, the code is divided into:
- Enemy — only for enemies.
- Item — only for items.
- This prevents unnecessary methods from being imposed on classes that don't use them.

5. Dependency Inversion Principle (DIP) — The principle of dependency inversion

- Battle, LevelManager, and ItemManager work through abstractions (Enemy, Item), rather than depending on specific implementations (Goblin, Orc, HealthPotion).
- EnemyFactory creates enemies by encapsulating the type selection logic, which reduces the tight binding to specific classes.
- Class structure after refactoring
- I've divided the code into three main groups:
- game.core (the main logic of the game)
- Player — is responsible for the player's behavior.
- Battle — controls the battle between the player and the enemy.
- LevelManager — controls the generation of levels.
- game.entities (enemies and factory)
- Enemy (interface)
- Goblin and Orc (specific classes of enemies)

- EnemyFactory is a factory for creating enemies.
- game.items
- Item (interface)
- HealthPotion (specific subject)
- ItemManager — manages random items.