**Refactoring MonolithicAdventureGame Using SOLID Principles**

**1.Project Overview**

MonolithicAdventureGame is a console-based RPG where a player fights enemies, collects items, gains experience, and levels up.

After refactoring, the code follows SOLID principles, making it modular, maintainable, and scalable.

**2. Project Structure**

Изображение выглядит как текст, снимок экрана

Контент, сгенерированный ИИ, может содержать ошибки.

**3. SOLID Principles Applied**

|  |  |  |  |
| --- | --- | --- | --- |
| **Principle** | | **Implementation in Code** | | --- |  |  | | --- | |  | |
| SRP (Single Responsibility Principle) | Each class has only **one responsibility** (separate classes for Player, Enemy, Items, Combat, Levels, and Score). |
| OCP (Open/Closed Principle) | New enemies and items can be **added without modifying** existing code (using **IEnemy** and **IItem** interfaces). |
| LSP (Liskov Substitution Principle) | **Zombie, Vampire, and Skeleton** can replace **Enemy** without breaking the game. |
| ISP (Interface Segregation Principle) | |  | | --- | |  |  |  | | --- | | Split **IEnemy** and **IItem** instead of using one large interface. | |
| DIP (Dependency Inversion Principle) | **CombatManager** depends on **IEnemy**, not specific enemy classes, making it flexible. |

**4️. Key Files & Classes**

**🔹 MainGame.java (Main File)**

* Initializes the player, score manager, level manager, and combat system.
* Creates enemy instances and triggers battles.
* Levels up the player and uses items.

**🔹 Player.java (Player Class)**

* Stores the player’s name and score.
* Handles enemy defeats and item usage.

**🔹 CombatManager.java (Combat System)**

* Manages battles between the player and enemies.
* Calls the **defeatEnemy()** method from the player class.

**🔹 Enemy.java & Subclasses (Skeleton.java, Zombie.java, Vampire.java)**

* Implements different enemy types.
* Uses an interface (IEnemy) for flexibility.

**🔹 ItemManager.java & Items (GoldCoin.java, HealthElixir.java, MagicScroll.java)**

* Manages item interactions.
* Implements the IItem interface for modularity.

