****Project Description: Hero's Adventure****

In this project, the player controls a hero who explores a fantasy world, battles monsters, gains experience points, and collects treasures. The game will be entirely text-based, with the player inputting commands via the console.

****Basic Concepts Covered:**** Variables and Data Types Input/Output (I/O) Conditional Statements (if-else) Loops (for, while) Arrays or Collections Methods and Functions OOP Concepts Covered:

****Classes and Objects**** Encapsulation Inheritance Polymorphism Game Features:

The hero can move through different locations (rooms). Random encounters with monsters trigger battles. The hero can attack, and the damage dealt depends on weapon strength and hero's attributes. The hero gains experience points (XP) after defeating monsters. The hero can collect treasures, such as gold coins or powerful items. The game continues until the hero achieves a certain goal or loses all health points (HP). Implementation Steps:

****Setup****:

Create a new Java project in your IDE. Organize your project structure (e.g., separate classes for Hero, Monster, Treasure, GameEngine). Design Classes:

Create classes for Hero, Monster, Treasure, and GameEngine. Define attributes and methods for each class. For example: Hero class: name, HP, attack damage, XP, etc. Monster class: name, HP, attack damage, XP, etc. Treasure class: type (gold, item), value, etc. GameEngine class: game loop, battle logic, movement, etc. Implement Basic Game Logic:

Initialize hero and monster objects. Implement movement between rooms. Handle random encounters with monsters. Implement basic battle mechanics (hero attacks, monster attacks, HP calculation). ****Add OOP Concepts****:

Use inheritance to create subclasses of Monster (e.g., Goblin, Dragon) with specific attributes. Implement polymorphism to handle different types of monsters in battles. Encapsulate class attributes and methods appropriately. ****Enhance Gameplay:****

Implement XP calculation and level-up system for the hero. Create a variety of treasures with different effects. Add complexity to battles (e.g., critical hits, special abilities). Implement a win/lose condition and game over screen. ****Test and Debug:****

Test your game thoroughly to ensure all features work as expected. Debug any issues that arise during testing.