📝 Module Project | Defeat the Evil Wizard

OVERVIEW

In this mini-project, you will create a hero character and battle the powerful Evil Wizard. You'll use Object-Oriented Programming (OOP) concepts to extend starter code, customize your character, and add new functionality. Through a simple menu system, you'll control your character by choosing actions like attacking, using special abilities, healing, and viewing stats. Your task is to modify the starter code and defeat the Evil Wizard by implementing your own logic and design.

🎯 LEARNING OBJECTIVES

- **Understand OOP Principles:** Practice inheritance, methods, and interactions between objects in Python.
- **Build Interactive Programs:** Implement a menu system for a turn-based game.
- **Design Game Logic:** Manage attacks, healing, and enemy interactions to make gameplay dynamic.

🔧 STARTER CODE

Start Code is attached at the bottom of this document so make sure to scroll down to get it before you start! Extend it to complete the project.

📋 PROJECT REQUIREMENTS

Create Your Own Character Classes:

- Add **two new character classes** in addition to Warrior and Mage. Examples include:
    - **Archer**: A ranged attacker with abilities to shoot arrows and evade attacks.

- **Paladin**: A defensive hero who can heal and shield against attacks.

Implement Special Abilities:

- Each character must have **two unique abilities**, such as:
  - **Archer**: "Quick Shot" (double arrow attack) and "Evade" (avoid next attack).
  - **Paladin**: "Holy Strike" (bonus damage) and "Divine Shield" (blocks the next attack).

Add a Healing Mechanic:

- Implement a `heal()` method that restores health but does not exceed the maximum.

Randomize Attack Damage:

- Modify the `attack()` method to deal **random damage within a range**.

Build a Turn-Based Battle System:

- Players must choose actions such as **attacking, healing, using abilities,** or **viewing stats**.

Evil Wizard Logic:

- The evil wizard should **regenerate health** and **attack the player** after each turn.

Display Victory/Defeat Messages:

- End the game with a **victory message** for the player or a **defeat message** if the wizard wins.

📤 DELIVERABLES

- A **Python script** containing:
  - Four character classes (including two new ones).
  - Two unique abilities for each character.
  - A working **turn-based battle system**.
- Display **victory or defeat messages** at the end.

🚀 BONUS TASKS (Optional):

- Create additional character classes with more complex mechanics.
- Add **random elements** to the wizard's attacks or **advanced abilities** (e.g., summoning minions).

📥 HOW TO SUBMIT

- Submit your completed **Python script** via Google Classroom as a GitHub repository.
- Ensure all features are implemented, and the game is fully functional.

📊 ASSESSMENT CRITERIA (100%)

Criteria	Weight	Description
OOP Implementation	40%	Correct use of classes, inheritance, and methods.
Game Functionality	40%	Smooth gameplay with clear actions and battle logic.
Code Quality	10%	Well-organized code with comments for clarity.
Creativity	10%	Bonus points for creative classes or advanced mechanics.

📚 RESOURCES

- Python OOP Documentation
- Basic Python Syntax

🔧 STARTER CODE

Python

```
# Base Character class
class Character:
    def __init__(self, name, health, attack_power):
        self.name = name
        self.health = health
        self.attack_power = attack_power
        self.max_health = health

    def attack(self, opponent):
        opponent.health -= self.attack_power
        print(f"{self.name} attacks {opponent.name} for
{self.attack_power} damage!")
        if opponent.health <= 0:
            print(f"{opponent.name} has been defeated!")

    def display_stats(self):
        print(f"{self.name}'s Stats - Health:
{self.health}/{self.max_health}, Attack Power:
{self.attack_power}")

# Warrior class (inherits from Character)
class Warrior(Character):
    def __init__(self, name):
        super().__init__(name, health=140, attack_power=25)

# Mage class (inherits from Character)
class Mage(Character):
    def __init__(self, name):
```

```
        super().__init__(name, health=100, attack_power=35)

# EvilWizard class (inherits from Character)
class EvilWizard(Character):
    def __init__(self, name):
        super().__init__(name, health=150, attack_power=15)

    def regenerate(self):
        self.health += 5
        print(f"{self.name} regenerates 5 health! Current health:
{self.health}")

# Create Archer class

# Create Paladin class


def create_character():
    print("Choose your character class:")
    print("1. Warrior")
    print("2. Mage")
    print("3. Archer")
    print("4. Paladin")

    class_choice = input("Enter the number of your class choice:
")
    name = input("Enter your character's name: ")

    if class_choice == '1':
        return Warrior(name)
    elif class_choice == '2':
        return Mage(name)
    elif class_choice == '3':
        pass  # Implement Archer class
    elif class_choice == '4':
        pass  # Implement Paladin class
```

```
    else:
        print("Invalid choice. Defaulting to Warrior.")
        return Warrior(name)

def battle(player, wizard):
    while wizard.health > 0 and player.health > 0:
        print("\n--- Your Turn ---")
        print("1. Attack")
        print("2. Use Special Ability")
        print("3. Heal")
        print("4. View Stats")

        choice = input("Choose an action: ")

        if choice == '1':
            player.attack(wizard)
        elif choice == '2':
            pass  # Implement special abilities
        elif choice == '3':
            pass  # Implement heal method
        elif choice == '4':
            player.display_stats()
        else:
            print("Invalid choice. Try again.")

        if wizard.health > 0:
            wizard.regenerate()
            wizard.attack(player)

        if player.health <= 0:
            print(f"{player.name} has been defeated!")
            break

    if wizard.health <= 0:
        print(f"The wizard {wizard.name} has been defeated by
{player.name}!")
```

```
def main():
    player = create_character()
    wizard = EvilWizard("The Dark Wizard")
    battle(player, wizard)

if __name__ == "__main__":
    main()
```