Звіт з лабораторної роботи No15

Тема роботи: Створення класів для моделювання персонажів гри.

Виконав: Кравченко Сергій, ТЦР-12

Дата: 06.06.2025

Мета роботи:

Навчитися створювати складніші класи, використовувати конструктори,

методи і властивості для моделювання персонажів у грі.

Хід роботи:

Постановка завдання:

Завдання 1. "Основний персонаж"

● Створіть клас Character, який має такі властивості: ім'я, рівень,

здоров'я, атака.

● Додайте метод для виводу інформації про персонажа.

Завдання 2. "Бій між персонажами"

● Додайте метод для атаки іншого персонажа, що зменшує його

здоров'я на величину атаки.

● Реалізуйте кілька персонажів і змоделюйте бій між ними.

Завдання 3. "Класи Героїв"

● Створіть класи Warrior, Mage, Archer, що наслідують

Character.

● Додайте для кожного з класів унікальні властивості (наприклад, магія

для Mage, стріли для Archer).

● Реалізуйте метод для відновлення здоров'я кожного персонажа.

Код програми:

class Character:

def init(self, name, level, health, attack):

self.name = name

self.level = level

self.health = health

self.attack = attack

def display\_info(self):

print(f"Ім'я: {self.name}")

print(f"Рівень: {self.level}")

print(f"Здоров'я: {self.health}")

print(f"Атака: {self.attack}")

def attack\_target(self, target):

print(f"{self.name} атакує {target.name} і завдає {self.attack} шкоди!")

target.health -= self.attack

if target.health <= 0:

print(f"{target.name} помер!")

else:

print(f"У {target.name} залишилось {target.health} здоров'я")

class Warrior(Character):

def init(self, name, level, health, attack, armor):

super().init(name, level, health, attack)

self.armor = armor

def display\_info(self):

super().display\_info()

print(f"Броня: {self.armor}")

def restore\_health(self):

self.health += 20

print(f"{self.name} відновив 20 здоров'я. Тепер у нього {self.health} HP")

class Mage(Character):

def init(self, name, level, health, attack, magic\_power):

super().init(name, level, health, attack)

self.magic\_power = magic\_power

def display\_info(self):

super().display\_info()

print(f"Магічна сила: {self.magic\_power}")

def restore\_health(self):

self.health += 15

print(f"{self.name} відновив 15 здоров'я за допомогою магії. Тепер у нього {self.health} HP")

class Archer(Character):

def init(self, name, level, health, attack, arrows):

super().init(name, level, health, attack)

self.arrows = arrows

def display\_info(self):

super().display\_info()

print(f"Кількість стріл: {self.arrows}")

def restore\_health(self):

self.health += 10

print(f"{self.name} відновив 10 здоров'я. Тепер у нього {self.health} HP")

# Демонстрація роботи класів

if name == "main":

print("=== Створення персонажів ===")

warrior = Warrior("Гаррі", 5, 100, 15, 10)

mage = Mage("Мерлін", 7, 80, 20, 50)

archer = Archer("Леголас", 6, 90, 18, 30)

warrior.display\_info()

print()

mage.display\_info()

print()

archer.display\_info()

print("\n=== Бій між персонажами ===")

warrior.attack\_target(mage)

mage.attack\_target(archer)

archer.attack\_target(warrior)

print("\n=== Відновлення здоров'я ===")

warrior.restore\_health()

mage.restore\_health()

archer.restore\_health()

Висновки

У ході виконання лабораторної роботи було створено систему класів для моделювання персонажів гри. Реалізовано базовий клас Character з основними властивостями та методами, а також спеціалізовані класи Warrior, Mage і Archer, які успадковують його функціональність та розширюють її унікальними можливостями. Було продемонстровано взаємодію між персонажами, включаючи атаки та відновлення здоров’я.

Робота показала важливість використання ООП для створення структурованих та модульних програм, де кожен клас відповідає за певну логіку та може бути легко модифікований або розширений. Отримані навички можна застосувати для розробки більш складних ігрових систем або інших проектів, що вимагають моделювання об’єктів з різними властивостями та поведінкою.