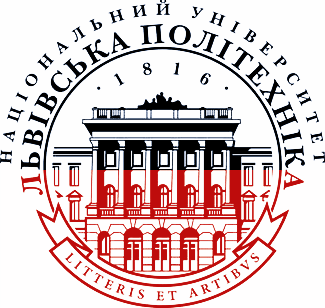
Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ



*Звіт*

до лабораторної роботи №9

З дисципліни: «Кросплатформні засоби програмування»

На тему: «ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON»

Виконала: ст. гр. КІ-305

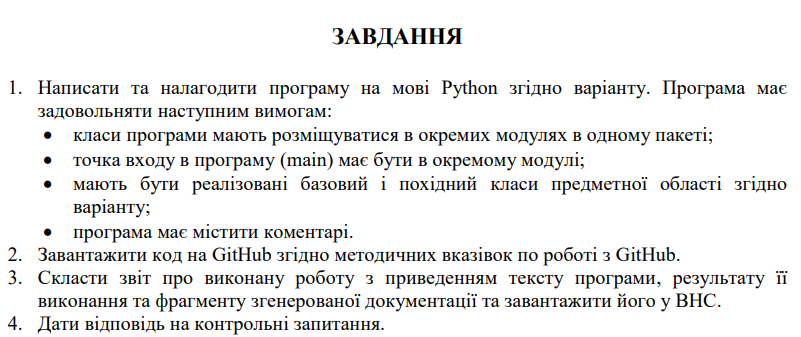
Романюк М. Р.

Прийняв: доцент кафедри ЕОМ

Іванов Ю. С.

Львів 2023

**Мета:** оволодіти навиками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.



Мій варіант №22:

Базовий клас: автомат.  
Похідний клас: штурмова гвинтівка.

**Код програми:**

**Файл Weapon.py**

# Модуль Weapon.py - базовий клас "Автомат"  
  
class Weapon:  
 def \_\_init\_\_(self, name, caliber, fire\_rate, magazine\_capacity=30):  
 """  
 Конструктор базового класу "Автомат"  
  
 :param name: Назва автомата  
 :param caliber: Калібр автомата  
 :param fire\_rate: Швидкість стрільби (кількість пострілів за секунду)  
 :param magazine\_capacity: Ємність магазину (за замовчуванням 30 набоїв)  
 """  
 self.name = name  
 self.caliber = caliber  
 self.fire\_rate = fire\_rate  
 self.magazine\_capacity = magazine\_capacity  
 self.ammo\_in\_magazine = magazine\_capacity  
 self.full\_auto = False # Режим стрільби (автоматичний/одиночний)  
  
 def fire(self):  
 """  
 Метод виконує постріл і виводить інформацію про нього.  
 """  
 if self.full\_auto:  
 print(f"{self.name} стріляє автоматично!")  
 else:  
 if self.ammo\_in\_magazine > 0:  
 print(f"{self.name} пострілив калібром {self.caliber} зі швидкістю {self.fire\_rate} пострілів за секунду.")  
 self.ammo\_in\_magazine -= 1  
 else:  
 print(f"Магазин {self.name} порожній! Потрібна перезарядка.")  
  
 def reload(self):  
 """  
 Метод виконує перезарядку автомата.  
 """  
 print(f"{self.name} перезаряджається...")  
 self.ammo\_in\_magazine = self.magazine\_capacity  
 print(f"Магазин {self.name} повністю заряджений!")  
  
 def toggle\_fire\_mode(self):  
 """  
 Метод змінює режим стрільби (автоматичний або одиночний).  
 """  
 self.full\_auto = not self.full\_auto  
 mode = "автоматичний" if self.full\_auto else "одиночний"  
 print(f"{self.name} переключено в режим стрільби {mode}.")  
  
 def get\_ammo(self):  
 """  
 Метод повертає кількість набоїв у магазині.  
 """  
 return self.ammo\_in\_magazine  
  
 def get\_info(self):  
 """  
 Метод повертає інформацію про зброю.  
 """  
 return f"Зброя: {self.name}, Калібр: {self.caliber}, Швидкість стрільби: {self.fire\_rate}, Кількість набоїв: {self.get\_ammo()}"

**Файл AssaultRifle.py:**

# Модуль AssaultRifle.py - похідний клас "Штурмова гвинтівка"  
  
from Weapon import Weapon # Імпорт базового класу "Автомат"  
  
class AssaultRifle(Weapon):  
 def \_\_init\_\_(self, name, caliber, fire\_rate, magazine\_capacity=30):  
 """  
 Конструктор похідного класу "Штурмова гвинтівка"  
  
 :param name: Назва автомата  
 :param caliber: Калібр автомата  
 :param fire\_rate: Швидкість стрільби (кількість пострілів за секунду)  
 :param magazine\_capacity: Ємність магазину (за замовчуванням 30 набоїв)  
 """  
 super().\_\_init\_\_(name, caliber, fire\_rate, magazine\_capacity)  
 self.grenade\_launcher = False # Наявність гранатомета  
  
 def launch\_grenade(self):  
 """  
 Метод виконує постріл гранатометом, якщо гранатомет є відомої гвинтівці.  
 """  
 if self.grenade\_launcher:  
 print(f"{self.name} вистрелив гранатометом!")  
 else:  
 print(f"{self.name} не має гранатомета!")  
  
 def attach\_grenade\_launcher(self):  
 """  
 Метод додає гранатомет до гвинтівки.  
 """  
 if not self.grenade\_launcher:  
 self.grenade\_launcher = True  
 print(f"До {self.name} додано гранатомет!")  
 else:  
 print(f"{self.name} вже має гранатомет!")

**Файл main.py:**

# Модуль Main.py - точка входу в програму  
  
from Weapon import Weapon  
from AssaultRifle import AssaultRifle  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 ak47 = AssaultRifle("AK-47", "7.62x39mm", 600)  
 ak47.toggle\_fire\_mode()  
 ak47.fire()  
 ak47.fire()  
 ak47.reload()  
 ak47.launch\_grenade()  
 ak47.attach\_grenade\_launcher()  
 ak47.launch\_grenade()  
  
 print(ak47.get\_info())  
  
 m16 = AssaultRifle("M16", "5.56x45mm", 700)  
 m16.toggle\_fire\_mode()  
 m16.fire()  
 m16.reload()  
  
 print(m16.get\_info())

# Результати роботи програми:

# 

# Відповіді на контрольні запитання

1. Що таке модулі?

Модулі - це файли Python, які містять функції, класи і змінні, які можна імпортувати і використовувати в інших програмних файлах. Модулі дозволяють організовувати та розділяти код на логічні блоки, що полегшує підтримку та перевикористання коду.

2. Як імпортувати модуль?

Імпортування модуля в Python виконується за допомогою ключового слова import. Наприклад:

import my\_module

3. Як оголосити клас?

class MyClass:

def \_\_init\_\_(self, param1, param2):

self.param1 = param1

self.param2 = param2

def my\_method(self):

# код методу

4. Що може міститися у класі?

У класі можуть міститися атрибути (змінні), методи (функції), конструктори, властивості та інші елементи, які визначають поведінку об'єктів цього класу.

5. Як називається конструктор класу?

Конструктор класу - це спеціальний метод з ім'ям \_\_init\_\_, який викликається при створенні нового об'єкта класу. Він використовується для ініціалізації атрибутів об'єкта.

6. Як здійснити спадкування?

Спадкування (наслідування) - це механізм в ООП, коли один клас може успадковувати атрибути та методи іншого класу. Це дозволяє створювати ієрархію класів і використовувати властивості та методи батьківського класу в похідних класах.

Спадкування в ООП (об'єктно-орієнтованому програмуванні) в Python здійснюється шляхом створення похідного класу (підкласу), який успадковує атрибути і методи від батьківського класу (суперкласу).

class ChildClass(ParentClass):

# Оголошення методів та атрибутів класу

Де:

ChildClass - це похідний клас (підклас), який ви створюєте.

ParentClass - це батьківський клас (суперклас), від якого ви бажаєте успадкувати властивості і методи.

7. Які види спадкування існують?

Існують два види спадкування:

Одинарне спадкування: Клас успадковує властивості і методи від одного батьківського класу.

Множинне спадкування: Клас успадковує властивості і методи від більше ніж одного батьківського класу. У Python підтримується множинне спадкування, але потребує обережного користування для уникнення конфліктів.

8. Які небезпеки є при множинному спадкуванні, як їх уникнути?

Небезпеки при множинному спадкуванні:

Конфлікти імен: Можуть виникати конфлікти імен методів або атрибутів між батьківськими класами.

Складність розуміння іерархії класів: Зі збільшенням кількості батьківських класів може стати складніше розуміти, які властивості і методи успадковуються.

Щоб уникнути цих небезпек, важливо створювати чітку ієрархію класів, використовувати абстрактні класи і інтерфейси, і обирати правильний підхід до проектування класів.

9. Що таке класи-домішки?

Класи-домішки (міксини) - це класи, які не мають своїх власних об'єктів і використовуються для розширення функціональності інших класів. Вони додають до класу певну функціональність, але не створюють власних об'єктів. Класи-домішки можуть бути успадковані і використовувані в інших класах для включення специфічної функціональності.

10. Яка роль функції super() при спадкуванні?

Функція super() при спадкуванні використовується для виклику методів батьківського класу в похідних класах. Вона дозволяє отримати доступ до методів та атрибутів батьківського класу та розширювати їх функціональність в похідному класі. super() допомагає уникнути конфліктів імен та забезпечує правильний порядок виконання методів при множинному спадкуванні.

**Висновок:** у ході виконання лабораторної роботи я оволоділа навичками реалізації парадигм об’єктно-орієнтованого програмування використовуючи засоби мови Python.