Міністерство освіти і науки України

Національний університет „Львівська політехніка”

Кафедра ЕОМ

**Звіт**

до лабораторної роботи № 9

з дисципліни: «Кросплатформні засоби програмування»

На тему: «Основи обʼєктно-орієнтованого програмування у Python»

Виконав:

Студент групи КІ-305

Яцків Адріян

Прийняв:

Іванов Ю. С.

# Львів 2023

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

# ЗАВДАННЯ

1. Написати та налагодити програму на мові Python згідно варіанту. Програма має задовольняти наступним вимогам:
   * класи програми мають розміщуватися в окремих модулях в одному пакеті;
   * точка входу в програму (main) має бути в окремому модулі;
   * мають бути реалізовані базовий і похідний класи предметної області згідно варіанту;
   * програма має містити коментарі.
2. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
4. Дати відповідь на контрольні запитання.

# Варіант завдання:

# Базовий клас: Людина

**Похідний клас:** Спортсмен

# Код програми: Person.py:

class Person:

    def \_\_init\_\_(self, name, age, gender):

        """Initialize a person."""

        self.name = name

        self.age = age

        self.gender = gender

    def info(self):

        """Information about person."""

        return f"Person name is {self.name}. Age: {self.age} years old. Gender: {self.gender}"

**Sporsman.py:**

from Person import Person

import random

class Sportsman(Person):

    def \_\_init\_\_(self, name, age, gender, sport, training\_hours):

        """Initialize an athlete. Inherits from the Person class."""

        super().\_\_init\_\_(name, age, gender)

        self.sport = sport

        self.training\_hours = training\_hours

    def train(self, hours):

        """Simulate athlete training and increase training hours."""

        self.training\_hours += hours

        return f"{self.name} trained for {hours} hours. Total training hours: {self.training\_hours}"

    def compete(self):

        """Simulate athlete competing and decrease energy level."""

        """Introduce a random chance of winning based on training hours """

        chance\_of\_winning = min(0.5 + 0.1 \* (self.training\_hours // 10), 0.9)

        if random.random() < chance\_of\_winning:

            return f"{self.name} won the competition! Congratulation."

        else:

            return f"{self.name} lost the competition. Better luck next time."

# main.py:

from Person import Person

from Sportsman import Sportsman

if \_\_name\_\_ == "\_\_main\_\_":

    # Create a regular person

    person1 = Person("Ivan", 30, "Male")

    print(person1.info())

    # Create an Sportsman

    sportsman1 =  Sportsman("John Doe", 28, "Male", "Runner", 1250)

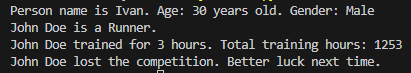
    sportsman2 =  Sportsman("David Blake", 23, "Male", "Swimmer", 760)

    print(f"{sportsman1.name} is a {sportsman1.sport}.")

    print(sportsman1.train(3))

    print(sportsman1.compete())

**Результати роботи програми:**



# Відповіді на контрольні запитання

1. Що таке модулі?

- Модулі в Python - це файли, які містять Python-код. Вони використовуються для організації коду у логічні групи, і можуть містити функції, класи, змінні та інші об'єкти.

1. Як імпортувати модуль? - import модуль
2. Як оголосити клас?

- class МійКлас: # Тіло класу

1. Що може міститися у класі?
   * атрибути (змінні), методи (функції), конструктори, спеціальні методи (наприклад, init , str ), властивості та інше.
2. Як називається конструктор класу?
   * Конструктор класу має ім'я init . Він викликається при створенні нового об'єкта класу і використовується для ініціалізації атрибутів об'єкта.
3. Як здійснити спадкування?
   * class ПідКлас(БазовийКлас):

# Тіло підкласу

1. Які види спадкування існують?
   * одиночне спадкування (коли підклас успадковує лише один базовий клас) та множинне спадкування (коли підклас успадковує більше одного базового класу).
2. Які небезпеки є при множинному спадкуванні, як їх уникнути?
   * Небезпеки при множинному спадкуванні включають в себе можливі конфлікти імен методів або атрибутів між базовими класами, що може призвести до непередбачуваної поведінки. Для уникнення цих проблем можна використовувати аліаси, викликати методи базових класів безпосередньо або використовувати композицію замість спадкування.
3. Що таке класи-домішки?
   * це класи, які містять певний функціонал і можуть бути використані для розширення функціональності інших класів. Вони не призначені для створення об'єктів, але можуть бути включені у інші класи за допомогою спадкування, щоб надати їм певну функціональність.
4. Яка роль функції super() при спадкуванні?
   * для виклику методів базового класу з підкласу. Вона допомагає уникнути явного вказівання імен базових класів та робить код більш гнучким при зміні структури спадкування. Наприклад, super(). init () викликає конструктор базового класу.

# Висновок

У ході виконання даної лабораторної роботи, я здобула важливі навички об'єктно- орієнтованого програмування мовою Python. Ознайомилась з ключовими аспектами цієї парадигми, включаючи створення та використання класів, роботу з об'єктами, та використання спадкування та поліморфізму для покращення ефективності програм.