Міністерство освіти і науки України
Національний університет «Львівська політехніка»
Кафедра «Електронних обчислювальних машин»

Звіт
з лабораторної роботи № 9
з дисципліни: «Кросплатформенні засоби програмування» на тему: «Основи Об'єктно-Орієнтованого програмування у Python»

Виконав:
студент групи КІ-306
Глухенький Д. Ю.
Прийняв:
доцент кафедри ЕОМ
Іванов Ю. С.

Львів – 2023

Мета роботи: оволодіти навиками реалізації парадигм об’єктноорієнтованого програмування використовуючи засоби мови Python.

Завдання (варіант № 5)

5. Машина

5. Вантажна машина

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.

Вихідний код програми

Файл Main.py

```
from Truck import Truck
from Car import Car


if __name__ == "__main__":
    truck1 = Truck('Truck1', 120, 2000)
    truck1.start()
    truck1.load_cargo(500)
    print(truck1.get_load())
    truck1.load_cargo(300)
    print(truck1.get_load())
    print(truck1.honk())


    car1 = Car("Car1", 180)
    car1.set_speed(80)
    car1.set_speed(90)
    print(car1.get_current_speed())

    truck1.unload_cargo(200)
    print(truck1.get_load())
```

Файл Truck.py

```
from Car import Car


class Truck(Car):
```

```
    def __init__(self, model, max_speed, load_capacity):
        super().__init__(model, max_speed)
        self.load_capacity = load_capacity
        self._load = 0

    def load_cargo(self, weight):
        if self._load + weight < self.load_capacity:
            self._load += weight
        else:
            self._load = self.load_capacity

    def unload_cargo(self, weight):
        if self._load - weight > 0:
            self._load -= weight
        else:
            self._load = 0

    def get_load(self):
        return self._load

    def get_load_capacity(self):
        return self.load_capacity
```

Файл Car.py

```
class Car:
    def __init__(self, model, max_speed):
        self.model = model
        self.max_speed = max_speed
        self._current_speed = 0

    def start(self):
        print(f"Start Car {self.model}")

    def stop(self):
        print(f'Stop Car {self.model}')

    def set_speed(self, speed):
        if speed < self.max_speed:
            self._current_speed = speed
        else:
            print(f"Max speed for this model is {self.max_speed} km/h")

    def get_current_speed(self):
        return self._current_speed

    def honk(self):
        print(f"{self.model} is honking")
```

Результат виконання програми

```
/Users/dimkamg21/univercity/course_3/CPPT/lab9/venv/bin/Python /Users/dimkamg21/univercity/course_3/CPPT/lab9/venv/main.py
Start Car Truck1
500
800
Truck1 is honking
None
90
600

Process finished with exit code 0
```

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.

2. Як імпортувати модуль?
   - import модуль

3. Як оголосити клас?
   - class МійКлас:
     # Тіло класу

4. Що може міститися у класі?
   - атрибути (змінні), методи (функції), конструктори, спеціальні методи (наприклад, __init__, __str__), властивості та інше.

5. 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.

6. Як здійснити спадкування?
   - class ПідКлас(БазовийКлас):
     # Тіло підкласу

7. Які види спадкування існують?
   - одиночне спадкування (коли підклас успадковує лише один базовий клас) та множинне спадкування (коли підклас успадковує більше одного базового класу).

8. 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.

9. 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.

10. 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, здобув важливі навички об'єктноорієнтованого програмування мовою Python. Ознайомився з ключовими аспектами цієї парадигми, включаючи створення та використання класів, роботу з об'єктами, та використання спадкування та поліморфізму для покращення ефективності програм.