Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

З лабораторної роботи №9

Варіант – 3

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Основи об’єктно-орієнтованого програмування у Python»

Виконав: ст. гр. КІ-306

Братівник Д. А.

Прийняв:

доцент кафедри ЕОМ

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: Пес

Похідний клас: Піддослідний пес

Код програми:

dog.py:

```
class Dog:
    def __init__(self, name, breed):
        """Initialize a dog with a name and breed."""
        self.name = name
        self.breed = breed
        self.energy_level = 100

    def bark(self):
        """Return the sound a dog makes."""
        return "Woof!"

    def play(self, hours):
        """Simulate the dog playing and decrease energy level."""
        energy_consumed = hours * 10
        if self.energy_level >= energy_consumed:
            self.energy_level -= energy_consumed
            return f"{self.name} played for {hours} hours. Energy level:
{self.energy_level}"
        else:
            return f"{self.name} is too tired to play."

    def sleep(self, hours):
        """Simulate the dog sleeping and increase energy level."""
        self.energy_level += hours * 5
        return f"{self.name} slept for {hours} hours. Energy level:
{self.energy_level}"
```

police_dog.py:

```
from dog import Dog

class PoliceDog(Dog):
    def __init__(self, name, breed, department):
        """
        Initialize a police dog with a name, breed, and department.
        Inherits from the Dog class.
        """
        super().__init__(name, breed)
        self.department = department
        self.training_level = 0

    def train(self, hours):
        """Simulate police dog training and increase training level."""
        self.training_level += hours * 3
        return f"{self.name} trained for {hours} hours. Training level:
{self.training_level}"

    def perform_duty(self):
        """Simulate the police dog performing duty and decrease energy level."""
        if self.energy_level >= 30 and self.training_level >= 50:
            self.energy_level -= 30
            self.training_level -= 50
            return f"{self.name} successfully performed duty. Energy level:
{self.energy_level}. Training level: {self.training_level}"
        else:
            return f"{self.name} is not ready to perform duty."
```

main.py:

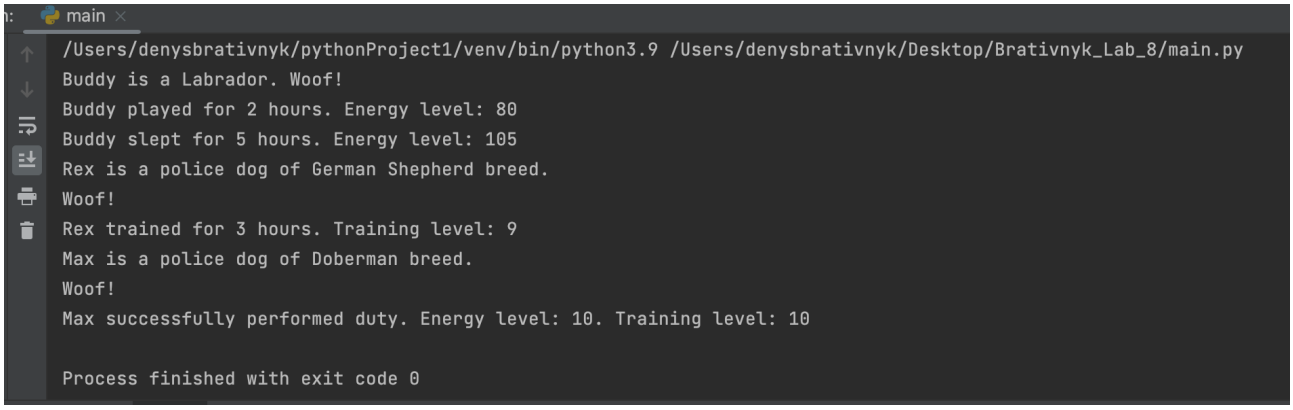
```
from dog import Dog
from police_dog import PoliceDog

if __name__ == "__main__":
    # Створюємо звичайну собаку
    dog1 = Dog("Buddy", "Labrador")
    print(f"{dog1.name} is a {dog1.breed}. {dog1.bark()}")
    print(dog1.play(2))  # Собака грає 2 години
    print(dog1.sleep(5))  # Собака спить 5 годин

    # Створюємо першу піддослідну собаку
    police_dog1 = PoliceDog("Rex", "German Shepherd", "K9 Unit")
    print(f"{police_dog1.name} is a police dog of {police_dog1.breed} breed.")
    print(police_dog1.bark())
    print(police_dog1.train(3))  # Поліцейська собака навчається 3 години

    # Створюємо другу піддослідну собаку, яка успішно виконує обов'язки
    police_dog2 = PoliceDog("Max", "Doberman", "K9 Unit")
    police_dog2.energy_level = 40  # Налаштовуємо рівень енергії
    police_dog2.training_level = 60  # Налаштовуємо рівень навчання
    print(f"{police_dog2.name} is a police dog of {police_dog2.breed} breed.")
    print(police_dog2.bark())
    print(police_dog2.perform_duty())  # Поліцейська собака успішно виконує
обов'язок
```

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.

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, я здобув важливі навички об'єктно-орієнтованого 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.