

**Міністерство освіти і науки України**  
**Національний університет «Львівська політехніка»**

**Кафедра ЕОМ**

**Звіт**



**Лабораторна робота № 9**

**“ ОСНОВИ ОБ’ЄКТНО-ОРІЄНТОВАНОГО ПРОГРАМУВАННЯ У PYTHON”**  
**з курсу “Кросплатформні засоби програмування”**  
**Варіант: 4**

**Виконав:**

**ст.гр.КІ-205**

**Воробець Тетяна**

**Прийняв:**

**доцент кафедри ЕОМ**

**Олексів М.В.**

**Львів 2024**

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

**Теоретичні відомості:** Спадкування в ООП призначене для розширення функціональності існуючих класів шляхом утворення нових класів на базі вже існуючих. У Java реалізована однокоренева архітектура класів згідно якої всі класи мають єдиного спільного предка (кореневий клас в ієрархії класів) – клас Object. Решта класів мови Java утворюються шляхом успадковування даного класу. Будь-яке спадкування у мові Java є відкритим, при цьому аналогів захищеному і приватному спадкуванню мови C++ не існує. На відміну від C++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова `class` після якого вказується назва підкласу, ключове слово `extends` та назва суперкласу, що розширюється у новому підкласі.

Механізм поліморфізму забезпечує можливість присвоєння об'єктним змінним суперкласу об'єктів похідних класів та звертання з-під цих змінних до перевизначених у підкласі членів суперкласу. У Java всі об'єктні змінні є поліморфними. Поліморфізм реалізується за допомогою механізму динамічного (пізнього) зв'язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається не на етапі компіляції, а під час виконання програми. У Java усі об'єктні змінні є типізовані. Механізми наслідування і поліморфізму дозволяють створювати нові типи (класи та інтерфейси) на базі вже існуючих та присвоювати об'єкти цих типів посиланням на об'єкти супертипу. В цьому випадку об'єкти підтипів мають ті самі елементи, що й об'єкти супертипу, тож таке висхідне приведення типів є безпечним і здійснюється компілятором автоматично. Проте присвоєння посилання на об'єкт підтипу об'єкту супертипу не завжди є коректним, тому таке приведення вимагає явного приведення типів. При такому приведенні типів можливі дві ситуації:

- якщо посилання на об'єкт супертипу реально посилається на об'єкт підтипу, то приведення посилання на об'єкт супертипу до типу підтипу є коректним;
- якщо посилання на об'єкт супертипу посилається на об'єкт супертипу, то приведення посилання на об'єкт супертипу до типу підтипу викличе виключну ситуацію `ClassCastException`.

Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об'єкти свого класу. Вони реалізуються за допомогою ключового слова `abstract`. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити). Абстрактні методи – це методи, що б оголошені з використанням ключового слова `abstract` і не місять тіла.

Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

- Завдання:** 1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
  3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
  4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
  5. Дати відповідь на контрольні запитання.

### **Код програми:**

#### master.py

```
#Class Master implements master
class Master:
    #Constructor
    def __init__(self, name_master="", number=""):
        self.name_master = name_master
        self.number = number

    #Method gets name master
    def get_name_master(self):
        return self.name_master

    # Method sets name master
    def set_name_master(self, name_master):
        self.name_master = name_master

    # Method gets number master
    def get_number(self):
        return self.number

    # Method sets number master
    def set_number(self, number):
        self.number = number
```

#### breed.py

```
#Class Breed implements breed
class Breed:
    # Constructor
    def __init__(self, breed=""):
        self.breed = breed

    # Method gets breed
    def get_breed(self):
        return self.breed

    # Method sets breed
    def set_breed(self, breed):
        self.breed = breed
```

#### collar.py

```
#Class Collar implements collar
class Collar:
    #Constructor
    def __init__(self, collar=False):
        self.collar = collar
```

```

# Method gets presence collar

def get_presence(self):
    return self.collar
# Method sets presence collar
def set_presence(self, collar):
    self.collar = collar

```

## cat.py

```

from breed import Breed
from master import Master
from collar import Collar

#Class Cat implements cat
class Cat:
    #Constructor
    def __init__(self, name="None", breed="None", number="None",
name_master="None", info_collar=False, location="None"):
        self.name = name
        self.breed = Breed(breed)
        self.master = Master(number, name_master)
        self.collar = Collar(info_collar)
        self.energy = 5
        self.food = 5
        self.location = location
        self.log_activity(f"Cat's name: {self.name}, Cat's breed:
{self.breed.get_breed()}, Cat's master: {self.master.get_number()},
{self.master.get_name_master()}, Cat's collar: {self.collar.get_presence()}")

    # Method gets name cat
    def get_name(self):
        return self.name

    # Method sets name cat
    def set_name(self, name):
        self.name = name
        self.log_activity(f"New name for cat's: {self.name}")

    # Method action play cat
    def play(self, game):
        if game == "mouse":
            self.energy -= 1
            self.food += 1
            self.log_activity(f"{self.name} plays with mouse: energy-1, food+1")
        elif game == "bug":
            self.energy += 1
            self.food -= 1
            self.location = "bugs"
            self.log_activity(f"{self.name} plays in bug: energy+1, food-1")
        else:
            self.energy -= 2
            self.food -= 1
            self.location = "outside"
            self.log_activity(f"{self.name} plays outside: energy-2, food-1")

    # Method action sleep cat
    def sleep(self):
        self.energy += 1
        self.log_activity(f"{self.name} sleeps: energy+1")

    # Method action clean cat
    def clean(self):
        self.energy -= 1
        self.log_activity(f"{self.name} cleans: energy-1")

    # Method to activate a night_vision cat's

```

```

def night_vision(self, visor):
    self.energy -= 1
    self.log_activity(f"{self.name} night vision {visor}: energy-1")

# Method to change location cat's
def set_place(self):
    self.log_activity(f"{self.name} in {self.location}")

# Method action eat cat
def eat(self, food):
    if food == "fish":
        self.food += 1
        self.log_activity(f"{self.name} eats fish: food+1")
    elif food == "meat":
        self.food += 2
        self.log_activity(f"{self.name} eats meat: food+2")
    else:
        self.food -= 1
        self.log_activity(f"{self.name} doesn't want to eat candy: food-1")

# Method to show status cat's
def status(self):
    if self.energy == 5 and self.food == 5:
        self.log_activity("Status: Nice")
    elif self.energy < 5 or self.food < 5:
        self.log_activity(f"Status: {self.name} needs to sleep or eat")
    else:
        self.log_activity(f"Status: {self.name} wants to play")
    self.log_activity(f"Energy level: {self.energy}, Food level: {self.food}")

# Method to show information about cat's
def info_cat(self):
    self.log_activity(f"Cat's name: {self.name}, Cat's breed: {self.breed.get_breed()}, Master's number: {self.master.get_number()}, Master's name: {self.master.get_name_master()}, Cat's collar: {self.collar.get_presence()}, Cat's energy: {self.energy}")

# Method to write action in file
def log_activity(self, message):
    with open("cat_activity.log", "a") as f:
        f.write(message + "\n")
    print(message)

```

### experiment\_cat.py

```

from cat import Cat
import random

# Class ExperimentCat implements experiment cat
class ExperimentCat(Cat):
    #Constructor
    def __init__(self, name="None", breed="None", number="None", name_master="None", info_collar=False, location="None"):
        super().__init__(name, breed, number, name_master, info_collar, location)
        self.poison = 0
        self.log_activity("ExperimentCat's name: "+self.name+", ExperimentCat's breed: "+self.breed.get_breed()+", Master's number: "+self.master.get_number()+", Master's name: "+self.master.get_name_master()+", ExperimentCat's collar: "+str(self.collar.get_presence()))

    # Method to put cat in the box
    def box(self):
        self.location = "Box"
        self.log_activity(self.name+" is in the "+self.location)

    # Method to put poison in the box

```

```

def put_poison(self):
    self.poison = 1
    self.log_activity("Poison is in the Box")

# Method to out poison in the box
def out_poison(self):
    self.poison = 0
    self.log_activity("Poison is out of the Box")

# Method to check status experiment
def check_experiment(self):
    if self.poison == 1 and self.location == "Box":
        result = random.randint(0, 1)
        if result == 0:
            self.log_activity(self.name+" is dead in the experiment")
        else:
            self.log_activity(self.name+" is alive in the experiment")
    else:
        self.log_activity("Conditions for the experiment are not met")

```

## main.py

```

from experiment_cat import ExperimentCat

#Start project

if __name__ == "__main__":
    shredding_cat = ExperimentCat("Shred", "british", "0987654321", "Shreding",
    True, "room")
    shredding_cat.status()
    shredding_cat.set_place()
    shredding_cat.put_poison()
    shredding_cat.check_experiment()
    shredding_cat.box()
    shredding_cat.out_poison()
    shredding_cat.check_experiment()
    shredding_cat.put_poison()
    shredding_cat.check_experiment()

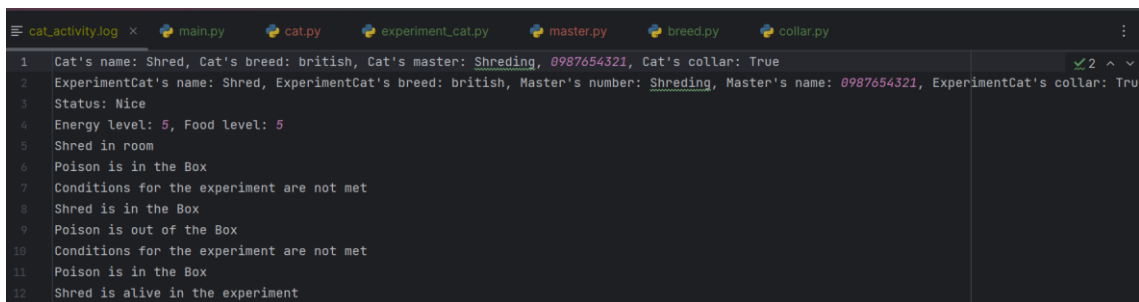
```

## Результат:

```

C:\Users\Tanya\AppData\Local\Programs\Python\Python38\python.exe "D:\5 семестр\Кроссплатформи\Кроссплатформи\Lab9\main.py"
Cat's name: Shred, Cat's breed: british, Cat's master: Shreding, 0987654321, Cat's collar: True
ExperimentCat's name: Shred, ExperimentCat's breed: british, Master's number: Shreding, Master's name: 0987654321, ExperimentCat's collar: True
Status: Nice
Energy level: 5, Food level: 5
Shred in room
Poison is in the Box
Conditions for the experiment are not met
Shred is in the Box
Poison is out of the Box
Conditions for the experiment are not met
Poison is in the Box
Shred is alive in the experiment
Process finished with exit code 0

```



```

cat_activity.log x main.py cat.py experiment_cat.py master.py breed.py collar.py
1 Cat's name: Shred, Cat's breed: british, Cat's master: Shreding, 0987654321, Cat's collar: True
2 ExperimentCat's name: Shred, ExperimentCat's breed: british, Master's number: Shreding, Master's name: 0987654321, ExperimentCat's collar: True
3 Status: Nice
4 Energy level: 5, Food level: 5
5 Shred in room
6 Poison is in the Box
7 Conditions for the experiment are not met
8 Shred is in the Box
9 Poison is out of the Box
10 Conditions for the experiment are not met
11 Poison is in the Box
12 Shred is alive in the experiment

```

## Посилання на репозиторій:

[https://github.com/NikaDe7/CPPT\\_Vorobets\\_TI\\_KI-35\\_1.git](https://github.com/NikaDe7/CPPT_Vorobets_TI_KI-35_1.git)

**Висновок:** оволоділа навиками реалізації парадигм об'єктно-орієнтованого програмування використовуючи засоби мови Python.