Міністерство освіти і науки України Національний університет «Львівська політехніка»

Звіт

до лабораторної роботи №4

З дисципліни: «Кросплатформенні засоби програмування»

На тему: «Спадкування та інтерфейси»

Варіант 4

Виконав:
Ст. групи КІ-34
Демчик Н.О.

Прийняв:
к.т.н., доцент
Іванов Ю.С.

Львів 2022

Мета: ознайомитися зі спадкуванням та інтерфейсами на мові Java.

ЗАВДАННЯ

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.
2. Автоматично згенерувати документацію до розробленого пакету.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.

Варіант завдання: 4. Піддослідний кіт

Код роботи

CatMain.java

```
public class CatMain {
    public static void main(String[] args) throws Exception {
        ExperimentalCat Simba = new ExperimentalCat(5.5f, "Scottish", "Simba", 4,
"Grey");

        Simba.doing();
        Simba.makeSound();
        Simba.sleeping(2);

        // in 3 types of classes
        Simba.setFoodGram(34);
        Simba.setFlea(true);
        Simba.setFur("thick");
        Simba.setNature("calm");
        Simba.setFoodTime(2);
        // print from all classes
        Simba.printAllCharacteristics();

        Simba.dispose();
    }
}
```

Cat.java

```
import java.io.FileNotFoundException;
import java.io.PrintWriter;

public abstract class Cat {
    private float weight;
    private String name;
    private int age;
    private String color;
    private PrintWriter fout;
    private Breed classBreed;
    private FoodAddiction addiction;
    private FleaInfestation flea;

    public Cat(float weight, String breed, String name, int age) throws
FileNotFoundException {
        this.weight = weight;
        this.name = name;
        this.age = age;
        classBreed = new Breed(breed);
        addiction = new FoodAddiction();
        flea = new FleaInfestation();
    }

    public Cat(float weight, String breed, String name, int age, String color)
throws FileNotFoundException {
        this(weight, breed, name, age);
        this.color = color;
    }

    protected void setFout(PrintWriter x) {
        fout = x;
    }

    public void setWeight(float newWeight) {
        this.weight = newWeight;
    }

    public float getWeight() {
        return weight;
    }

    public String getName() {
        return name;
    }

    public int getAge() {
        return age;
    }

    public String getColor() {
```

```
        return color;
    }

    public void myMood(String mood) {
        System.out.println("I am: " + mood + ". And this is me ");
        fout.println("I am: " + mood + ". And this is me ");
        fout.flush();
    }

    public void makeSound() {
        System.out.println("Meow");
        fout.println("Meow");
        fout.flush();
    }

    public void doNotSleepAt3AM() {
        System.out.println("I'm scraping the door");
        fout.println("I'm scraping the door");
        fout.flush();
    }

    public void sleeping(int time) {
        System.out.println("Sleeping for " + time + "hours");
        fout.println("Sleeping for " + time + "hours");
        fout.flush();
    }

    public void eat(String food) {
        System.out.println("I'm eating " + food);
        fout.println("I'm eating " + food);
        fout.flush();
    }

    public void watchInTheWindow() {
        System.out.println("Oh wow, there is a bird!!");
        fout.println("Oh wow, there is a bird!!");
        fout.flush();
    }

    public boolean isIll() {
        if (age < 8) {
            return false;
        } else {
            return true;
        }
    }
    // 3 classes methods

    public void setFur(String fur) {
        classBreed.setFur(fur);
    }

    public void setNature(String nature) {
```

```
        classBreed.setNature(nature);
    }

    public void setFoodTime(int foodTime) {
        addiction.setFoodTimes(foodTime);
    }

    public void setFoodGram(int foodGram) {
        addiction.setFoodGrams(foodGram);
    }

    public void setFlea(boolean yes) {
        flea.setFlea(yes);
    }

    public void setPercentage(int per) {
        flea.setPercentageOfFlea(per);
    }

    public void printAllCharacteristics() {
        System.out.println("Full Properties START: --------------------");
        fout.println("Full Properties START: --------------------");

        // this class
        System.out.println("Name: " + getName());
        fout.println("Name: " + getName());

        System.out.println("Age: " + getAge());
        fout.println("Age: " + getAge());

        if (color != null) {
            System.out.println("Color: " + getColor());
            fout.println("Color: " + getColor());
        }

        System.out.println("Weight: " + getWeight());
        fout.println("Weight: " + getWeight());

        System.out.println("Am i ill - " + (isIll() ? "Yes" : "No"));
        fout.println("Am i ill - " + (isIll() ? "Yes" : "No"));

        // 3 classes
        classBreed.printBreed();
        fout.println("Breed is : " + classBreed.getBreed());
        fout.println("Fur is : " + classBreed.getFur());
        fout.println("Nature is : " + classBreed.getNature());

        flea.printFlea();
        fout.println("Has flea ? : " + flea.getFlea());
        fout.println("Possibility to get flea : " + flea.getFlea());

        addiction.printFood();
```

```
        fout.println("How many times needs to eat ? : " +
addiction.getFoodTimes());
        fout.println("Food in grams for cat for 1 bowl: " +
addiction.getFoodGrams());

        System.out.println("Full Properties END: --------------------");
        fout.println("Full Properties END: --------------------");
        fout.flush();
    }

    public void dispose() {
        fout.close();
    }
}
```

ExperimentalCat.java

```
import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;
import java.util.Scanner;

public class ExperimentalCat extends Cat implements Experiments {
    private PrintWriter fout;

    public ExperimentalCat(float weight, String breed, String name, int age) throws
FileNotFoundException {
        super(weight, breed, name, age);
        fout = new PrintWriter(new File("CatLog.txt"));
        setFout(fout);
    }

    public ExperimentalCat(float weight, String breed, String name, int age, String
color)
            throws FileNotFoundException {
        super(weight, breed, name, age, color);
        fout = new PrintWriter(new File("CatLog.txt"));
        setFout(fout);
    }

    public void doing() {
        System.out.println("Doing");
    }

    public void onExperiment() {
        System.out.println("On experiment");
    }

    @Override
    public void printAllCharacteristics() {
        super.printAllCharacteristics();

        Scanner scan = new Scanner(System.in);
        System.out.println("Enter max speed of this cat on training :");
        float speed = scan.nextFloat();
        System.out.println("After using potion max speed: " + catMaxSpeed(speed *
potion));
        fout.println("After using potion max speed: " + catMaxSpeed(speed *
potion));

        System.out.println("After using potion new weight: " + changingWeght());
        fout.println("After using potion new weight: " + changingWeght());
        scan.close();
    }

    @Override
    public double catMaxSpeed(float speed) {
```

```
        return speed;
    }

    @Override
    public float changingWeght() {
        setWeight(getWeight() * potion);
        return getWeight();
    }
}

interface Experiments {
    void onExperiment();

    double catMaxSpeed(float speed);

    float changingWeght();

    float potion = 1.1f; // уявна константа
}
```

FleaInfestation.java

```java
public class FleaInfestation {
    private boolean isFlea;
    private int percentageOfFlea;

    FleaInfestation() {
        isFlea = false;
        percentageOfFlea = 50;
    }

    public void printFlea() {
        System.out.println("Has flea ? : " + isFlea);
        System.out.println("Possibility to get flea : " + percentageOfFlea);
    }

    public boolean getFlea() {
        return isFlea;
    }

    public int getPercentage() {
        return percentageOfFlea;
    }

    public void setFlea(boolean fle) {
        isFlea = fle;
    }

    public void setPercentageOfFlea(int percentage) {
        percentageOfFlea = percentage;
    }
}
```

Breed.java

```java
public class Breed {
    private String breed;
    private String fur;
    private String nature;

    Breed(String breed) {
        this.breed = breed;
        this.fur = "normal";
        this.nature = "normal";
    }

    public void printBreed() {
        System.out.println("Breed is : " + breed);
        System.out.println("Fur is : " + fur);
        System.out.println("Nature is : " + nature);

    }

    public void setFur(String fur) {
        this.fur = fur;
    }

    public void setNature(String nature) {
        this.nature = nature;
    }

    public String getBreed() {
        return breed;
    }

    public String getFur() {
        return fur;
    }

    public String getNature() {
        return nature;
    }
}
```

FoodAddiction.java

```java
public class FoodAddiction {
    private int foodTimesPerDay;
    private double foodInGrams;

    FoodAddiction() {
        foodTimesPerDay = 3;
        foodInGrams = 40.0f;
    }

    public void setFoodTimes(int foodTime) {
        foodTimesPerDay = foodTime;
    }

    public void setFoodGrams(int foodGrams) {
        foodInGrams = foodGrams;
    }

    public int getFoodTimes() {
        return foodTimesPerDay;
    }

    public double getFoodGrams() {
        return foodInGrams;
    }

    public void printFood() {
        System.out.println("How many times needs to eat ? : " + foodTimesPerDay);
        System.out.println("Food in grams for cat for 1 bowl : " + foodInGrams);
    }
}
```

Результат виконання програми

CatLog.txt

```
Meow
Sleeping for 2hours
Full Properties START: ------------------
Name: Simba
Age: 4
Color: Grey
Weight: 5.5
Am i ill - No
Breed is : Scottish
Fur is : thick
Nature is : calm
Has flea ? : true
Possibility to get flea : true
How many times needs to eat ? : 2
Food in grams for cat for 1 bowl: 34.0
Full Properties END: -------------------
After using potion max speed: 38.5
After using potion new weight: 6.655

```

- Консоль

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
Doing
Meow
Sleeping for 2hours
Full Properties START: -------------------
Name: Simba
Age: 4
Color: Grey
Weight: 5.5
Am i ill - No
Breed is : Scottish
Fur is : thick
Nature is : calm
Has flea ? : true
Possibility to get flea : 50
How many times needs to eat ? : 2
Food in grams for cat for 1 bowl : 34.0
Full Properties END: -------------------
Enter max speed of this cat on training :
35
After using potion max speed: 38.5
After using potion new weight: 6.05
```

Відповіді на контрольні запитання:

1. class Підклас extends Суперклас {
   Додаткові поля і методи
   }
2. Найчастіше супер-клас – це базовий клас, а підклас – це похідний клас від суперкласу.
3. Для звернення до методів чи полів суперкласу з підкласу потрібно використати ключове слово *super*.
   super.назваМетоду([параметри]); // виклик методу суперкласу
   super.назваПоля // звертання до поля суперкласу
4. Статичне зв'язування використовується при поліморфізмі. (компіляція).
   Лише тоді, коли метод є приватним, статичним або констрктором.
5. 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.
6. Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об'єкти свого класу. Вони реалізуються за допомогою ключового 49 слова abstract. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити).
7. Використовується для визначення типу об'єкта в момент виконання програми. Посилання на базовий клас.
8. При наслідувані у Java дозволяється перевизначення (перевантаження) методів та полів. При цьому область видимості методу, що перевизначається, має бути не меншою, ніж область видимості цього методу у суперкласі, інакше компілятор видасть повідомлення, про обмеження привілеїв доступу до даних. Перевизначення методу полягає у визначені у підкласі методу з сигнатурою методу суперкласу. При виклику такого методу з-під об'єкта підкласу викличеться метод цього підкласу. Якщо ж у підкласі немає визначеного методу, що викликається, то викличеться метод

суперкласу. Якщо ж у суперкласі даний метод також відсутній, то згенерується повідомлення про помилку.

9. 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.

10. Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу {

Прототипи методів та оголошення констант інтерфейсу

}

Висновок*:* я ознайомився зі спадкуванням та інтерфейсами на мові Java**.**