

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



ЗВІТ

з лабораторної роботи № 4

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «СПАДКУВАННЯ ТА ІНТЕРФЕЙСИ»

Виконав: ст. гр. КІ-35

Дністрян Я. В.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю.С

Львів-2022

Мета роботи: ознайомитися з спадкуванням та інтерфейсами у мові Java

ЗАВДАННЯ (Варіант - 4)

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.

4. Піддослідний кіт

Рис.1. Підклас

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.

Текст програми:

Лістинг Cat.java

```
/**
 * Lab 4 package
 */
package KI35.Dnistrian.Lab4;

import java.io.*;
import java.util.*;

/**
 * Abstract class Cat implements cat
 * @author Yaroslav Dnistrian
 */
public abstract class Cat{
    private Eyes eyes;
    private Paws paws;
    private Fur fur;
    private PrintWriter fout;

    /**
     * Constructor
     * @param <code>vision</code> Vision mode
     */
}
```

```

    * @param <code>colorEye</code> The color of our cat's eye
    * @param <code>distanceToPrey</code> Distance to prey
    * @param <code>speed</code> The speed of our cat
    * @param <code>condition</code> The position claw of our cat
    * @param <code>colorFur</code> The color fur of our cat
    * @param <code>lengthFur</code> The length of our cat's fur
    * @throws FileNotFoundException
    */
    public Cat(Eyes.Regime vision,String colorEye,float distanceToPrey,float speed,
Paws.clawCondition condition,String colorFur,int lengthFur) throws
FileNotFoundException
    {
        eyes = new Eyes(vision,colorEye,distanceToPrey);
        paws=new Paws(speed,condition);
        fur = new Fur(colorFur,lengthFur);
        fout = new PrintWriter(new File("Cat.txt"));
    }

    /**
     * Constructor
     * @throws FileNotFoundException
     */
    public Cat() throws FileNotFoundException {
        eyes = new Eyes(Eyes.Regime.dilated,"Brown",120.0F);
        paws=new Paws(35.5F, Paws.clawCondition.out);
        fur = new Fur("White",5);
        fout = new PrintWriter(new File("Cat.txt"));
    }

    /**
     * Method returns the vision mode of our cat
     * @return vision mode
     */
    public Eyes.Regime getVision(){
        return eyes.getRegime();
    }

    /**
     * Method returns the cat's claw condition
     * @return claw condition
     */
    public Paws.clawCondition getClawCondition(){
        return paws.getClaw();
    }

    /**
     * Method returns the speed of our cat
     * @return speed of our cat
     */
    public float getSpeedCat() {
        return paws.getSpeed();
    }

    /**
     * Method returns the distance to prey
     * @return distance to prey
     */
    public float getDistanceToPreyCat() {
        return eyes.getDistanceToPrey();
    }

    /**
     * Method returns the length of cat's fur
     * @return length of cat's fur
     */

```

```

public int getLengthFurCat() {
    return fur.getLegth();
}

/**
 * Method returns the color of cat's eyes
 * @return color of cat's eyes
 */
public String getColorEyeCat() {
    return eyes.getColorEye();
}

/**
 * Method returns the color of cat's eyes
 * @return color of cat's fur
 */
public String getColorFurCat() {
    return fur.getColorFur();
}

/**
 * Method reports the color of the cat's eyes and fur
 * @param <code>eye</code> color of cat's eyes
 * @param <code>fur</code> color of cat's fur
 */
public void colorCat(String eye, String fur) {
    System.out.println("The color of the cat's eyes: " + eye + "\nThe color of
the cat's fur: " + fur );
    fout.println("The color of the cat's eyes: " + eye + "\nThe color of the
cat's fur: " + fur);
}

/**
 * Method reports the behavior of the cat
 * @param <code>vision</code> Vision mode
 * @param <code>condition</code>The position claw of our cat
 */
public void pet(Eyes.Regime vision,Paws.clawCondition condition)
{
    if(vision==Eyes.Regime.dilated &&condition==Paws.clawCondition.out) {
        System.out.println("Do not touch, the cat focused on the prey!!!");
        fout.println("Do not touch, the cat focused on the prey!!!");
    }else if(vision==Eyes.Regime.narrow &&condition==Paws.clawCondition.out) {
        System.out.println("You can pet the cat, but there will be
scratches");
        fout.println("You can pet the cat, but there will be scratches");
    }else {
        System.out.println("You can pet the cat");
        fout.println("You can pet the cat");
    }
}

/**
 * Method calculates the cat's hunting speed
 * @param speed Speed of our cat
 * @param distanceToPrey Distance to prey
 */
public void timeToEat(float speed,float distanceToPrey) {
    float res;
    res= distanceToPrey/speed;
    String time=String.format("Time to hunt for prey: %.2f s. \n" ,res);
    System.out.print(time);
    fout.print(time);
}

```

```

    }

    /**
     * Method counts the number of scratches by the cat
     */
    public void numberOfScratches() {
        Random numOfScratches = new Random();
        int number = numOfScratches.nextInt(10);
        System.out.println("The number of scratches caused by the cat: " +
number);
        fout.println("The number of scratches caused by the cat:" + number );
    }

    /**
     * Method counts the time of cat petting
     */
    public void catPettingTime() {
        Random time = new Random();
        int number = time.nextInt(20);
        System.out.println("Cat petting time : " + number + " minute");
        fout.println("Cat petting time : " + number + " minute");
    }
}

```

```

    /**
     * Method sets the cat's fur type
     * @param lengthFur The length of our cat's fur
     */
    public void typeOfCatFur(int lengthFur) {
        if(lengthFur>=5) {
            System.out.println("Type of cat fur: Long-haired");
            fout.println("Type of cat fur: Long-haired");
        }
        else {
            System.out.println("Type of cat fur: Short-haired");
            fout.println("Type of cat fur: Short-haired");
        }
    }

    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        fout.close();
    }
}

```

```

class Eyes{
    enum Regime{dilated,narrow};
    private Regime vision;
    private String colorEye;
    private float distanceToPrey;

    /**
     * Constructor
     * @param<code>vision</code> Vision mode
     * @param<code>colorEye</code> The color of our cat's eye
     * @param<code>distanceToPrey</code> Distance to prey
     */
    public Eyes(Regime vision, String colorEye,float distanceToPrey ) {
        this.vision=vision;
        this.colorEye=colorEye;
    }
}

```

```

        this.distanceToPrey=distanceToPrey;
    }

    /**
     * Method sets the vision mode
     * @param vision Vision mode
     */
    public void setRegime(Regime vision) {
        this.vision=vision;
    }

    /**
     * Method returns the vision mode
     * @return vision
     */
    public Regime getRegime() {
        return vision;
    }

    /**
     * Method returns the color eyes of our cat
     * @return color eyes of our cat
     */
    public String getColorEye() {
        return colorEye;
    }

    /**
     * Method sets the distance to prey
     * @param distance Distance to prey
     */
    public void setDistanceToPrey(float distance) {
        this.distanceToPrey=distance;
    }

    /**
     * Method returns the distance to prey
     * @return Distance to prey
     */
    public float getDistanceToPrey() {
        return distanceToPrey;
    }
}

```

```

class Paws{

```

```

    enum clawCondition{out,within};

```

```

    private float speed;
    private clawCondition clawCondition;

```

```

    /**
     * Constructor
     * @param <code>speed</code> The speed of our cat
     * @param <code>condition</code> The position claw of our cat
     */
    public Paws(float speed, clawCondition condition){
        this.speed= speed;
        this.clawCondition=condition;
    }

```

```

    /**
     * Method sets the speed of our cat

```

```

        * @param speed Speed of our cat
        */
        public void setSpeed(float speed) {
            this.speed=speed;
        }
        /**
         * Method returns the speed of our cat
         * @return speed
         */
        public float getSpeed(){
            return speed;
        }
        /**
         * Method sets the cat's claw condition
         * @param condition The position claw of our cat
         */
        public void setClaw(clawCondition condition ) {
            this.clawCondition=condition;
        }
        /**
         * Method returns the cat's claw condition
         * @return cat's claw condition
         */
        public clawCondition getClaw() {
            return clawCondition;
        }
    }
}

```

```

class Fur{
    private String colorFur;
    private int lengthFur;
    /**
     * Constructor
     * @param <code>colorFur</code> The color fur of our cat
     * @param <code>lengthFur</code> The length of our cat's fur
     */
    public Fur(String colorFur,int lengthFur) {
        this.colorFur=colorFur;
        this.lengthFur=lengthFur;
    }
    /**
     * Method sets the color fur of our cat
     * @param colorFur color fur of our cat
     */
    public void setColorFur(String colorFur) {
        this.colorFur=colorFur;
    }
    /**
     * Method returns color fur of our cat
     * @return color fur of our cat
     */
    public String getColorFur() {
        return colorFur;
    }
    /**
     * Method sets the length of our cat's fur

```

```

    * @param lengthFur The length of our cat's fur
    */
    public void setLengthFur(int lengthFur) {
        this.lengthFur=lengthFur;
    }
    /**
     * Method returns the length of our cat's fur
     * @return The length of our cat's fur
     */
    public int getLegth() {
        return lengthFur;
    }
}

```

Лістинг CatApp.java

```

package KI35.Dnistrian.Lab4;
import java.io.*;
import java.util.Random;

/**
 * class CatLab implements catCondition, schrodinger
 * @author Yaroslav Dnistrian
 */
public class CatLab extends Cat implements catCondition,schrodinger {
    private int levelOfAgressive=0;
    boolean isSuitable;

    public CatLab() throws FileNotFoundException{
        agree();
    }

    /**
     * Method shows whether the cat is sick or not
     * @param speed the speed of our cat
     */
    public void isSick(float speed) {
        if(speed>=30) {
            System.out.println("Cat isn't sick");
        }
        else System.out.println("Cat is sick");
    }

    /**
     * Method calculates the level of cat anger
     */
    public void agree() {
        if(getVision()==Eyes.Regime.dilated &&
getClawCondition()==Paws.clawCondition.out)
            levelOfAgressive= veryAgree;
        else if(getVision()==Eyes.Regime.narrow &&
getClawCondition()==Paws.clawCondition.out)
            levelOfAgressive=averageAgree;
        else levelOfAgressive=LittleAgree;
    }

    public int getLevelOfAgressive() {
        return levelOfAgressive;
    }
}

```



```

    }

    public boolean getSuitable() {
        return isSuitable;
    }

    /**
     * bethod shows whether our cat is suitable for the experiment
     * @param levelagree
     * @param speed
     */
    public void suitable(int levelagree, float speed) {
        if(levelagree==0 && speed>=30) {
            System.out.println("The cat is suitable for the experiment");
            isSuitable=true;
        }
        else {
            System.out.println("You need to get another cat");
            isSuitable=false;
        }
    }

    /**
     * Method of conducting the experiment
     */
    public void experiment() {
        Random ran = new Random();
        int dead = ran.nextInt(2);
        if(dead==0)
            System.out.println("The cat is dead((");
        else System.out.println("The cat is alive)))");
    }
}

```

Лістинг CatLabApp.java

```

package KI35.Dnistrian.Lab4;

import java.io.*;

public class CatLabApp {
    /**
     *
     * @param args
     * @throws FileNotFoundException
     */
    public static void main(String[] args) throws FileNotFoundException{
        CatLab tom= new CatLab();

        tom.colorCat(tom.getColorEyeCat(),tom.getColorFurCat());
        tom.typeOfCatFur(tom.getLengthFurCat());
        tom.pet(tom.getVision(),tom.getClawCondition());

        if(tom.getVision()==Eyes.Regime.dilated &&
tom.getClawCondition()==Paws.clawCondition.out)

```

```

        tom.timeToEat(tom.getSpeedCat(), tom.getDistanceToPreyCat());
        else if(tom.getVision()==Eyes.Regime.narrow &&
tom.getClawCondition()==Paws.clawCondition.out)
            tom.numberOfScratches();
        else
            tom.catPettingTime();
//new functionality
        tom.isSick(tom.getSpeedCat());
        System.out.println("Level of aggression:" + tom.getLevelOfAgressive()+
"%");
        tom.suitable(tom.getLevelOfAgressive(), tom.getSpeedCat());
        if(tom.getSuitable()) {
            tom.experiment();
        }
        tom.dispose();
    }
}

```

Лістинг catCondition.java

```

package KI35.Dnistrian.Lab4;

public interface catCondition {
    int veryAgree = 100;
    int averageAgree = 50;
    int littleAgree = 0;
    public void isSick(float speed);
    public void agree();
}

```

Лістинг schrodinger.java

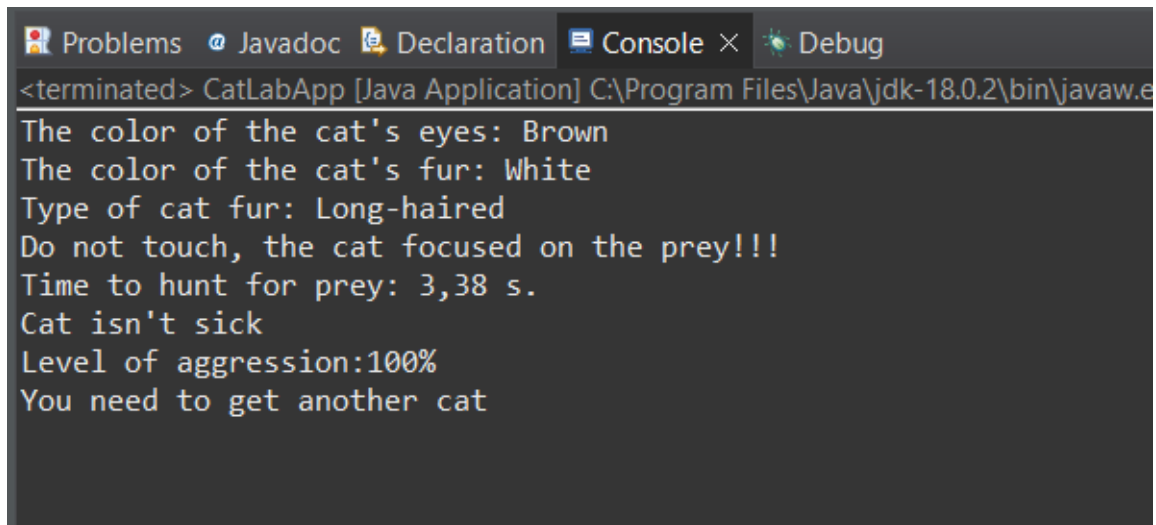
```

package KI35.Dnistrian.Lab4;

public interface schrodinger {
    void suitable(int levelagree, float speed);
    void experiment();
}

```

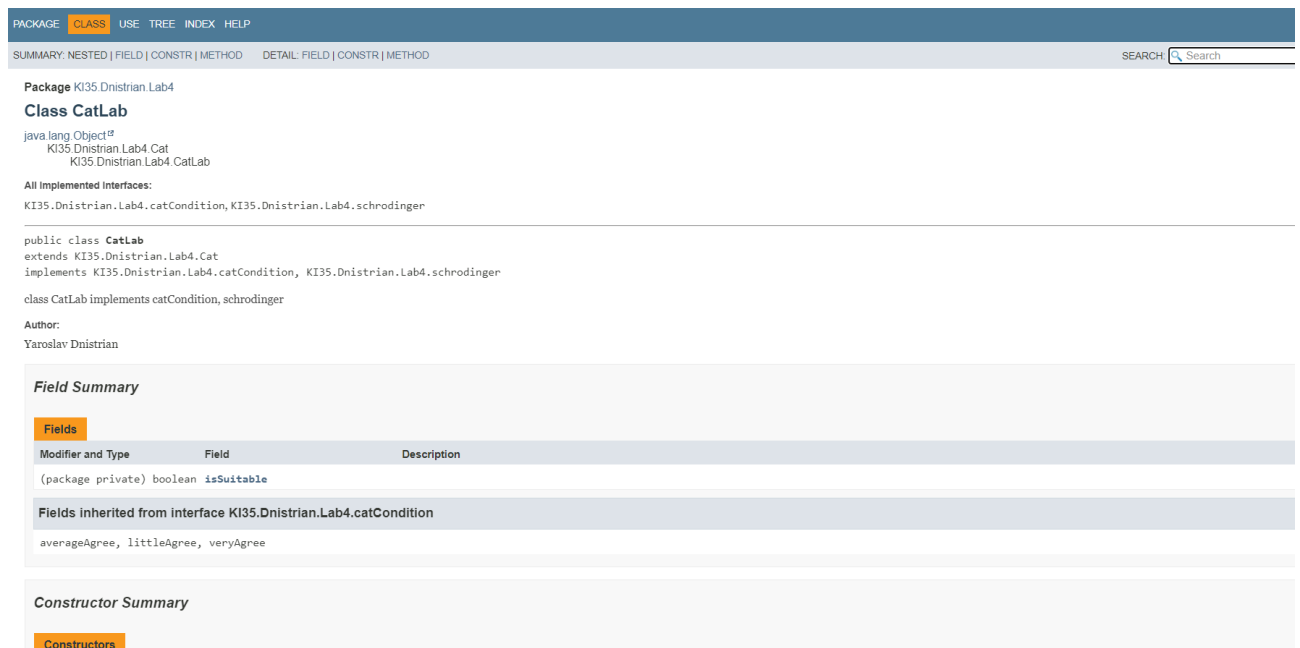
Результат виконання програми



```
<terminated> CatLabApp [Java Application] C:\Program Files\Java\jdk-18.0.2\bin\javaw.e
The color of the cat's eyes: Brown
The color of the cat's fur: White
Type of cat fur: Long-haired
Do not touch, the cat focused on the prey!!!
Time to hunt for prey: 3,38 s.
Cat isn't sick
Level of aggression:100%
You need to get another cat
```

Рис.2.Результати роботи програми

Фрагмент згенерованої документації



PACKAGE **CLASS** USE TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD SEARCH:

Package KI35.Dnistrian.Lab4

Class CatLab

java.lang.Object¹⁹
KI35.Dnistrian.Lab4.Cat
KI35.Dnistrian.Lab4.CatLab

All implemented interfaces:
KI35.Dnistrian.Lab4.catCondition, KI35.Dnistrian.Lab4.schrodinger

public class **CatLab**
extends KI35.Dnistrian.Lab4.Cat
implements KI35.Dnistrian.Lab4.catCondition, KI35.Dnistrian.Lab4.schrodinger
class CatLab implements catCondition, schrodinger

Author:
Yaroslav Dnistrian

Field Summary

Fields

Modifier and Type	Field	Description
(package private) boolean	isSuitable	

Fields inherited from interface KI35.Dnistrian.Lab4.catCondition
averageAgree, littleAgree, veryAgree

Constructor Summary

Constructors

Рис .4. Результат автоматичної генерації документації

Відповіді на контрольні запитання:

1. Що таке абстрактний клас та як його реалізувати?

Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об'єкти свого класу. Вони реалізуються за допомогою ключового слова `abstract`.

2. Що таке інтерфейс?

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

Висновок: виконуючи лабораторну роботу №4, я ознайомився з спадкуванням та інтерфейсами у мові Java