

Міністерство освіти і науки України
Національний університет „Львівська політехніка”



Звіт
з лабораторної роботи №4
дисципліни: **“Кросплатформні засоби програмування”**
на тему: **“ Спадкування та інтерфейси “**

Виконав:
ст. гр. КІ-34
Скалій Т.В.
Прийняв:
Іванов Ю.С.

Львів – 2022

Мета: ознайомитися з спадкуванням та інтерфейсами у 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 розробленої програми.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
4. Дати відповідь на контрольні запитання.
- 5.

Варіант № 20

Чоботи

Лістинг програми:

```
package KI34.Skalii.Lab4;
import java.io.FileNotFoundException;

interface IBoots
{
    void takeOffInsulation();
    void insulationOnBoots();
}
/**
 * Class <code>Boots</code> extends Shoes and implements IBoots
 * @author Tetiana Skalii
 * @version 1.0
 */
public class Boots extends Shoes implements IBoots {

    boolean insulation = false;

    /**
     * Constructor
     * @param name
     */
    public Boots() throws FileNotFoundException{
        super("Airmax97", 35, "leather");
    }
    /**
     * Constructor
     * @param name
     * @param size
     * @param material
     */
    public Boots(String name,int size, String material) throws FileNotFoundException{
        super(name,size, material);
    }
}
```

```

/**
 * Method takes off insulation
 */
public void takeOffInsulation()
{
    if(insulation)
    {
        insulation = false;
        System.out.println("Taking off insulation");
        fout.println("Taking off insulation");
        return;
    }
    System.out.println("insulation removed");
}
/**
 * Method adds insulation on boots
 */
public void insulationOnBoots()
{
    if(!insulation)
    {
        insulation = true;
    }
    System.out.println("The insulation are on");
}
}

```

ShoesApp.java

```

/**
 * lab 4 package
 */
package KI34.Skalii.Lab4;

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

/**
 * Class <code>ShoesApp</code>
 * @author Tetiana Skalii
 * @version 1.0
 */
public class ShoesApp {
    /**
     * @param args
     */
    public static void main(String[] args) {
        try {
            Boots Martens = new Boots("Dr.Martens 1460", 39, "leather");
            Martens.Start();
            Martens.showInfo();
            Martens.showClean();
            Martens.insulationOnBoots();
            Martens.changeLacing(2);
            Martens.showLacing();
            Martens.takeOffInsulation();
            Martens.showSize();
            Martens.End();
            Martens.Start();
            Martens.showClean();
            Martens.showRepair();
            Martens.repair();
            Martens.showRepair();
            Martens.showMaterial();
            Martens.clean();
            Martens.End();
        }
    }
}

```

```

        Martens.showClean();
        Martens.dispose();
    }
    catch (Exception e) {
        System.out.println(e);
    }
}
}

```

Shoes.java

```

/**
 * lab 4 package
 */
package KI34.Skalii.Lab4;
import java.io.*;
/**
 * Class <code>Shoes</code> implements Shoes
 * @author Tetiana Skalii
 * @version 1.0
 */
abstract class Shoes {
    boolean isPutOn = false;
    int size = 0;
    String material;
    String[] lacingTypes = { "Hash", "Twistie", "Riding bow", "Lattice", "Zipper" };
    String lacing = "";
    Wash washed = new Wash();
    Repair repaired = new Repair();
    protected PrintWriter fout;
    String name;
    public static int count;
    /**
     * Constructor
     * Creates shoes pair
     */
    public Shoes(String n, int s, String m) throws FileNotFoundException {
        lacing = "Hash";
        size = s;
        name = n;
        material = m;
        fout = new PrintWriter(new File("KI34.Skalii.Lab4.txt"));
        if (s > 37)
            count++;
    }
    /**
     * Method for put on shoes
     */
    public void Start() {
        if (isPutOn != true) {
            isPutOn = true;
            washed.weared();
            repaired.weared();
            System.out.println("Put on");
            fout.println("Put on");
        }
        else {
            System.out.println("You already put on these shoes");
            fout.println("You already put on these shoes");
        }
    }
}

/**
 * Method for taking off the shoes

```

```

    */
    public void End() {
        if (isPutOn != false) {
            isPutOn = false;
            System.out.println("Take off");
            fout.println("Take off");
        }
        else {
            System.out.println("You took off those shoes");
            fout.println("You took off those shoes");
        }
    }

    /**
     * Method shows lacing type
     */
    public void showLacing() {
        System.out.println(lacing + " lacing");
        fout.println(lacing + " lacing");
    }

    /**
     * Method shows material type
     */
    public void showMaterial() {
        System.out.println("Youre shoes are made of-" + material);
        fout.println("Youre shoes are made of-" + material);
    }

    /**
     * Method changes lacing type
     */
    public void changeLacing(int type) {
        lacing = lacingTypes[type - 1];
        System.out.println("The lacing type changed to: " + lacing);
        fout.println("The lacing type changed to: " + lacing);
    }

    /**
     * Method shows size
     */
    public void showSize() {
        System.out.println("Shoe size is:" + size);
        fout.println("Shoe size is:" + size);
    }

    /**
     * Method shows info
     */
    public void showInfo() {
        System.out.printf("Name: %s ; Size: %d ; Material: %s ;\n", name, size,
material);
        fout.printf("Name: %s Size: %d Material: %s \n", name, size, material);
    }

    /**
     * Method shows if cleaning needed
     */
    public void showClean() {
        System.out.println(washed.isWashed());
        fout.println(washed.isWashed());
    }

    /**
     * Method cleans the shoes
     */
    public void clean() {
        System.out.println(washed.clean());
        fout.println(washed.clean());
    }
}
/**

```

```

        * Method shows if repair is needed
        */
    public void showRepair() {
        System.out.println(repaired.isRepaired());
        fout.println(repaired.isRepaired());
    }
    /**
     * Method repairs the shoes
     */
    public void repair() {
        System.out.println(repaired.Repair1());
        fout.println(repaired.Repair1());
    }
    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        fout.flush();
        fout.close();
    }

}

```

Repair.java

```
package KI34.Skalii.Lab4;
```

```

/**
 * Class <code>Repair</code>
 * @author Tetiana Skalii
 * @version 1.0
 */
public class Repair {
    int repaired;
    /**
     * Implements repair default
     */
    public Repair(){
        repaired = 100;
    }
    /**
     * Implements repair by value
     */
    public Repair(int x){
        repaired = x;
    }

    /**
     * Method shows if repair is needed
     */
    public String isRepaired(){
        return("Your shoes on "+repaired+"% unharmed");
    }
    /**
     * Method repairs the shoes
     */
    public String Repair1(){
        repaired = 100;
        return("Your shoes have been repaired!");
    }
    /**
     * Method change repair value
     */
    public void weared(){

```

```

        repaired -= 1;
        if (repaired <= 0){
            repaired = 0;
            System.out.println("Your shoes are beyond repair:");
        }
    }
}

```

Wash.java

```

package KI34.Skalii.Lab4;
/**
 * Class <code>Wash</code>
 * @author Tetiana Skalii
 * @version 1.0
 */
public class Wash {
    int washed;
    /**
     * Implements washing by default
     */
    public Wash(){
        washed = 100;
    }
    /**
     * Implements washing by value
     */
    public Wash(int x){
        washed = x;
    }

    /**
     * Method shows if washing is needed
     */
    public String isWashed(){
        return ("Your shoes are "+washed+"% clean");
    }
    /**
     * Method cleans the shoes
     */
    public String clean(){
        washed = 100;
        return ("Your shoes are now clean!");
    }
    /**
     * Method change wash value
     */
    public void weared(){
        washed -= 2;
        if (washed <= 0){
            washed = 0;
        }
    }
}

```

Результат виконання :

```
<terminated> ShoesApp (1) [Java Application] C:\Program Files\Java\jdk-18.C
Put on
Name: Dr.Martens 1460 ; Size: 39 ; Material: leather
Your shoes are 98% clean
The insulation are on
The lacing type changed to: Twistie
Twistie lacing
Taking off insulation
Shoe size is:39
Take off
Put on
Your shoes are 96% clean
Your shoes on 98% unharmed
Your shoes have been repaired!
Your shoes on 100% unharmed
Your shoes are made of-leather
Your shoes are now clean!
Take off
Your shoes are 100% clean
```

Рис.1.Результат виводу на консоль



KI34.Skalii.Lab4: Блокнот

Файл Редагування Формат Вигляд Довідка

```
Put on
Name: Dr.Martens 1460 Size: 39 Material: leather
Your shoes are 98% clean
The lacing type changed to: Twistie
Twistie lacing
Taking off insulation
Shoe size is:39
Take off
Put on
Your shoes are 96% clean
Your shoes on 98% unharmed
Your shoes have been repaired!
Your shoes on 100% unharmed
Your shoes are made of-leather
Your shoes are now clean!
Take off
Your shoes are 100% clean
```

Рис.2.Результат виводу у файл

Згенерована документація

PACKAGE CLASS USE TREE INDEX HELP

PACKAGE: DESCRIPTION | RELATED PACKAGES | CLASSES AND INTERFACES

Package KI34.Skalii.Lab4

package KI34.Skalii.Lab4

Classes

Class	Description
Boots	Class Boots extends Shoes and implements IBoots
Repair	Class Repair
Shoes	Class Shoes implements Shoes
ShoesApp	Class ShoesApp
Wash	Class Wash

Рис.3.Вміст вкладки Package

Package KI34.Skalii.Lab4

Class Shoes

java.lang.Object[Ⓔ]
KI34.Skalii.Lab4.Shoes

Direct Known Subclasses:

Boots

abstract class Shoes
extends Object[Ⓔ]

Class Shoes implements Shoes

Version:

1.0

Author:

Tetiana Skalii

Field Summary

Fields

Modifier and Type	Field	Description
static int	count	
protected PrintWriter [Ⓔ]	fout	
(package private) boolean	isPutOn	
(package private) String [Ⓔ]	lacing	
(package private) String [Ⓔ] []	lacingTypes	
(package private) String [Ⓔ]	material	
(package private) String [Ⓔ]	name	
(package private) Repair	repaired	
(package private) int	size	
(package private) Wash	washed	

Constructor Summary

Constructors

Constructor	Description
Shoes(String [Ⓔ] n, int s, String [Ⓔ] m)	Constructor Creates shoes pair

Method Summary

All Methods

Instance Methods

Concrete Methods

Modifier and Type	Method	Description
void	changeLacing(int type)	Method changes lacing type
void	clean()	Method cleans the shoes
void	dispose()	Method releases used recourses

Рис.4.Вміст вкладки Class(Shoes)

Class Boots extends Shoes and implements IBoots

Version:

1.0

Author:

Tetiana Skalii

Field Summary

Fields inherited from class KI34.Skalii.Lab4.Shoes

count

Constructor Summary

Constructors

Constructor	Description
Boots()	Constructor
Boots(String [Ⓜ] name, int size, String [Ⓜ] material)	Constructor

Method Summary

All MethodsInstance MethodsConcrete Methods

Modifier and Type	Method	Description
void	spikesOnSole()	Method adds spikes on sole
void	takeOffSpikes()	Method takes off spikes

Methods inherited from class KI34.Skalii.Lab4.Shoes

changeLacing, clean, dispose, End, repair, showClean, showInfo, showLacing, showMaterial, showRepair, showSize, Start

Methods inherited from class java.lang.Object[Ⓜ]

equals[Ⓜ], getClass[Ⓜ], hashCode[Ⓜ], notify[Ⓜ], notifyAll[Ⓜ], toString[Ⓜ], wait[Ⓜ], wait[Ⓜ], wait[Ⓜ]

Constructor Details

Boots

public Boots()
throws FileNotFoundException[Ⓜ]

Constructor

Parameters:

name -

Throws:

FileNotFoundException[Ⓜ]

Рис.5.Вміст вкладки Class(Boots)

Відповіді на контрольні запитання:

1. Синтаксис реалізації спадкування.

Синтаксис реалізації спадкування:

```
class Підклас extends Суперклас
{
```

Додаткові поля і методи

}

2. Що таке суперклас та підклас??

В термінах мови Java базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супермножині.

3. Як звернутися до членів суперкласу з підкласу?

Виклик методу суперкласу:

super.назваМетоду([параметри]);

Звертання до поля суперкласу:

super.назваПоля

4. Коли використовується статичне зв'язування при виклику методу?

Статичне зв'язування використовується коли метод є приватним, статичним, фінальним або конструктором.

5. Як відбувається динамічне зв'язування при виклику методу?

Віртуальна машина повинна викликати версію методу, що відповідає фактичному типу об'єкту на який посилається об'єктна змінна.

Оскільки на пошук необхідного методу потрібно багато часу, то віртуальна машина заздалегідь створює для кожного класу таблицю методів, в якій перелічуються сигнатури всіх методів і фактичні методи, що підлягають виклику. При виклику методу віртуальна машина просто переглядає таблицю методів.

6. Що таке абстрактний клас та як його реалізувати?

Абстрактний клас – це клас, для якого не можна створити об'єкти, призначений бути основою для розробки ієрархії класів.

Реалізується за допомогою ключового слова `abstract`.

7. Для чого використовується ключове слово `instanceof`?

Оператор `instanceof` дозволяє визначити, чи вказаний об'єкт належить до заданого типу.

8. Як перевірити чи клас є підкласом іншого класу?

Щоб перевірити чи клас є підкласом іншого класу, потрібно за допомогою `instanceof` порівняти, чи дійсно посилання на об'єкт супертипу посилається на об'єкт підтипу.

9. Що таке інтерфейс?

Це абстрактний тип, який використовується для визначення поведінки, яку класи повинні реалізовувати. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

10. Як оголосити та застосувати інтерфейс?

Синтаксис оголошення інтерфейсів:

```
[public] interface НазваІнтерфейсу
```

```
{
```

```
    Прототипи методів та оголошення констант інтерфейсу
```

```
}
```

Висновок:

На цій лабораторній роботі я ознайомила з спадкуванням та інтерфейсами у мові Java.