Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

з лабораторної роботи №3

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Спадкування та інтерфейси”

Виконав:

ст. гр. КІ-305

Ніколенко О.В.

Прийняв:

Іванов Ю.С.

Львів – 2023

**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 №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 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.

|  |
| --- |
| Чоботи |

**Варіант № 20**

**Лістинг програми:**

Boots.java

**package** KI305.Nikolenko.Lab3;

**import** java.io.FileNotFoundException;

**interface** IBoots

{

**void** takeOffInsulation();

**void** insulationOnBoots();

}

/\*\*

\* Class <code>Boots</code> extends Shoes and implements IBoots

\* **@author** Oleksandr Nikolenko

\* **@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 KI305.Nikolenko.Lab3;

import java.io.\*;

import java.util.\*;

/\*\*

\* Class <code>ShoesApp</code>

\* @author Oleksandr Nikolenko

\* @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 KI305.Nikolenko.Lab3;

import java.io.\*;

/\*\*

\* Class <code>Shoes</code> implements Shoes

\* @author Oleksandr Nikolenko

\* @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** KI305.Nikolenko.Lab3;

/\*\*

\* Class <code>Repair</code>

\* **@author** Oleksandr Nikolenko

\* **@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** KI305.Nikolenko.Lab3;

/\*\*

\* Class <code>Wash</code>

\* **@author** Oleksandr Nikolenko

\* **@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){

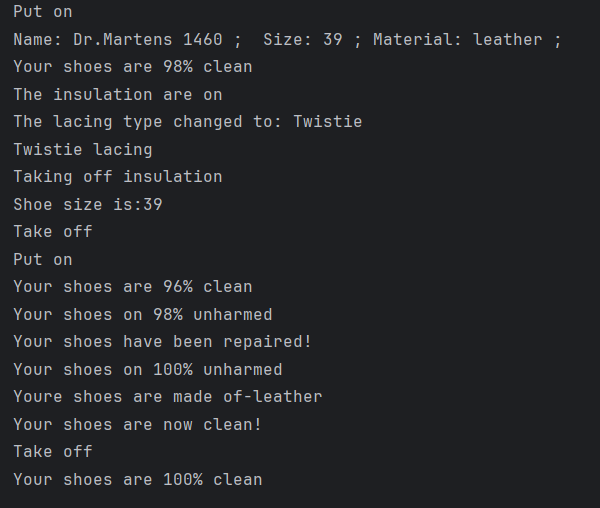
washed = 0;

}

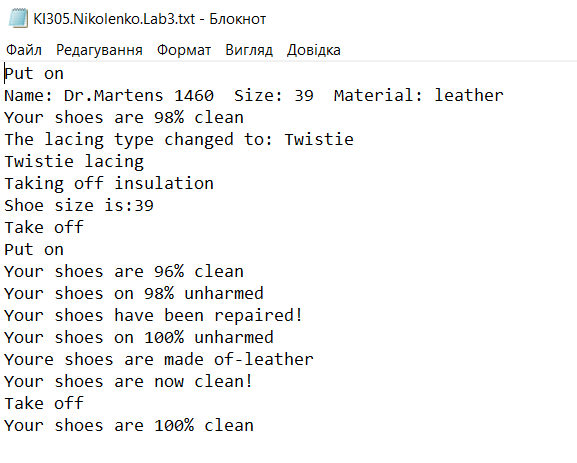
}

}

**Результат виконання :**

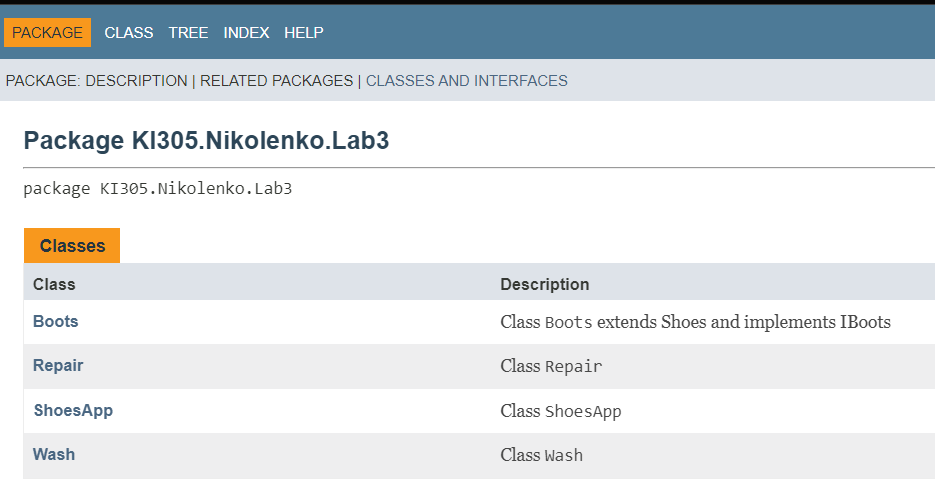
****

*Рис.1.Результат виводу на консоль*

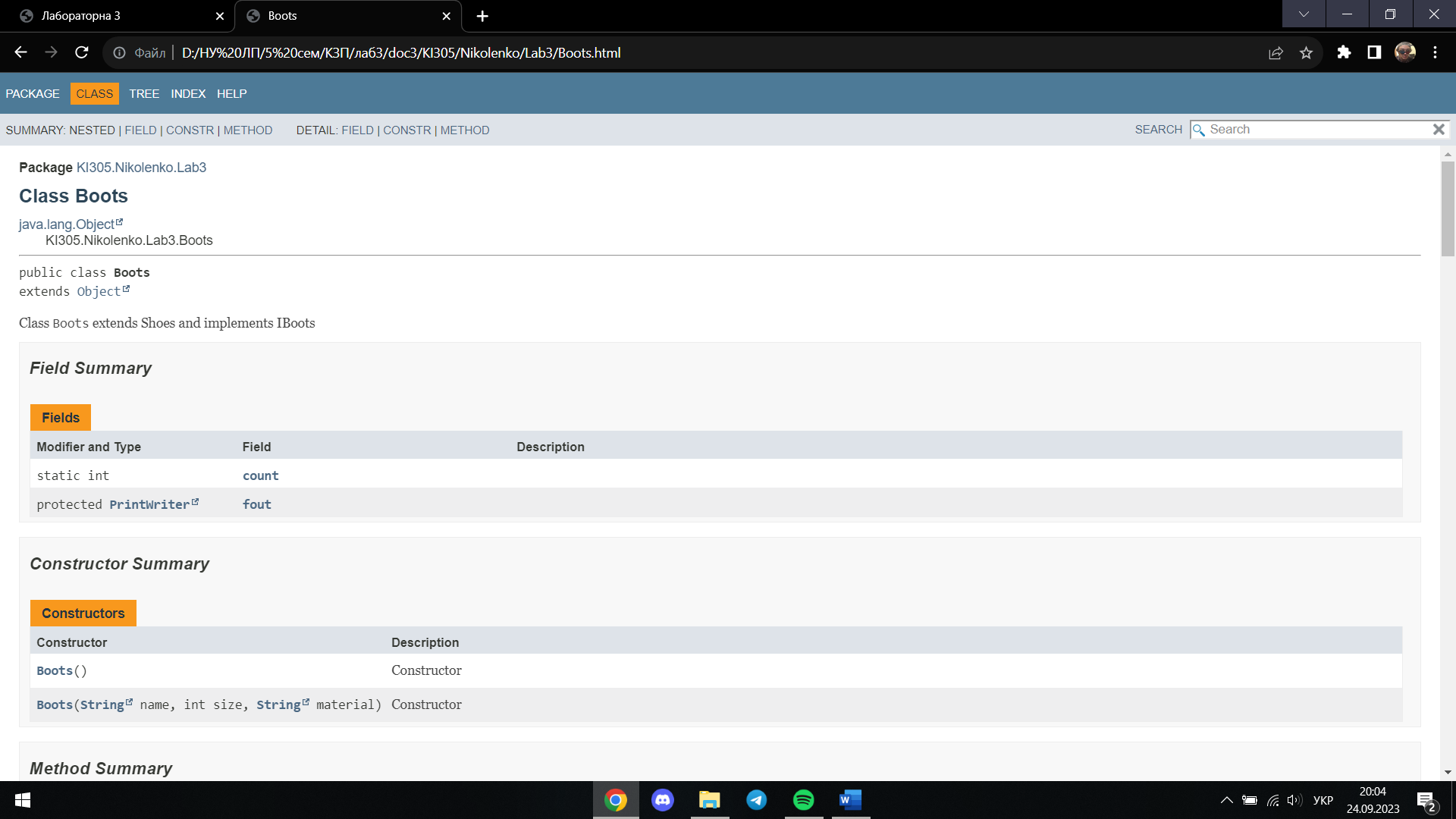
**

*Рис.2.Результат виводу у файл*

**Згенерована документація**

****

*Рис.3.Вміст вкладки Package*



*Рис.4.Вміст вкладки Сlass(Boots)*

**Відповіді на контрольні запитання:**

1. **Синтаксис реалізації спадкування.**

Синтаксис реалізації спадкування:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

1. **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.

1. **Як звернутися до членів суперкласу з підкласу?**

Виклик методу суперкласу:

*super.назваМетоду([параметри]);*

Звертання до поля суперкласу:

*super.назваПоля*

1. **Коли використовується статичне зв’язування при виклику методу?**

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

1. **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.

1. **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.

1. **Для чого використовується ключове слово instanceof?**

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

1. **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.

1. **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.

1. **Як оголосити та застосувати інтерфейс?**

Синтаксис оголошення інтерфейсів:

[public] interface НазваІнтерфейсу

{

Прототипи методів та оголошення констант інтерфейсу

}

**Висновок:**

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