Міністерство освіти і науки України

Національний університет «Львівська політехніка»



**ЗВІТ**

з лабораторної роботи №2

з дисципліни: “Кросплатформні засоби програмування”

на тему: “Класи та пакети”

Виконав:

ст. гр. КІ-305

Ніколенко О.В.

Прийняв:

Іванов Ю.С.

Львів – 2023

**Мета роботи:** ознайомитися з процесом розробки класів та пакетів мовою Java.

**Завдання:**

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту. Програма має задовольняти наступним вимогам:

* програма має розміщуватися в пакеті Група.Прізвище.Lab2;
* клас має містити мінімум 3 поля, що є об’єктами класів, які описують складові частини предметної області;
* клас має містити кілька конструкторів та мінімум 10 методів;
* для тестування і демонстрації роботи розробленого класу розробити клас-драйвер;
* методи класу мають вести протокол своєї діяльності, що записується у файл;
* розробити механізм коректного завершення роботи з файлом (не надіятися на метод finalize();
* програма має володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

1. Автоматично згенерувати документацію до розробленої програми.
2. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
3. Дати відповідь на контрольні запитання.

|  |
| --- |
| Взуття |

**Варіант № 20**

**Лістинг програми:**

*ShoesApp.java*

**package** KI305.Nikolenko.Lab2;

**import** java.io.\*;

/\*\*

\* Shoes Application class implements main method for Shoes class possibilities demonstration

\* **@author** Oleksandr Nikolenko

\* **@version** 1.0

\*/

**public** **class** ShoesApp {

/\*\*

\* **@param** args

\* **@throws** FileNotFoundException

\*/

**public** **static** **void** main(String[] args) **throws** FileNotFoundException {

**try**{

Shoes AirForce = **new** Shoes(38, "Leather");

AirForce.Start();

AirForce.showClean();

AirForce.showLacing();

AirForce.changeLacing(2);

AirForce.showLacing();

AirForce.End();

AirForce.Start();

AirForce.showClean();

AirForce.showRepair();

AirForce.showSize();

AirForce.repair();

AirForce.showRepair();

AirForce.showMaterial();

AirForce.clean();

AirForce.End();

AirForce.showClean();

AirForce.dispose();

}

**catch** (Exception e) {

System.***out***.println(e);

}

}

}

*Shoes.java*

**package** KI305.Nikolenko.Lab2;

**import** java.io.\*;

/\*\*

\* Class <code>Shoes</code>

\* **@author** Oleksandr Nikolenko

\* **@version** 1.0

\*/

**public** **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();

**private** PrintWriter fout;

/\*\*

\* Constructor

\* Creates shoes pair

\*/

**public** Shoes(**int** s, String m)**throws** FileNotFoundException{

lacing = "Hash";

size = s;

material = m;

fout = **new** PrintWriter( **new** File("KI305.Nikolenko.Lab2.txt"));

}

/\*\*

\* 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 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.Lab2;

/\*\*

\* 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.Lab2;

/\*\*

\* 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){

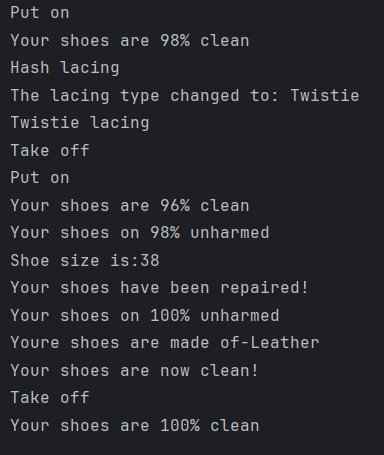
washed = 0;

}

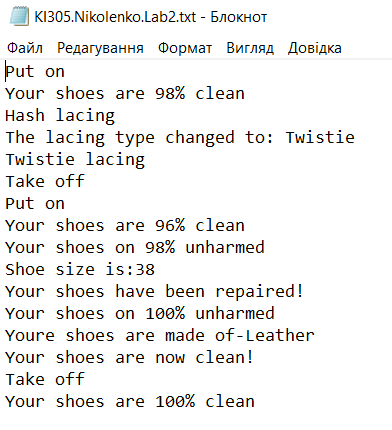
}

}

**Результат виконання програми:**

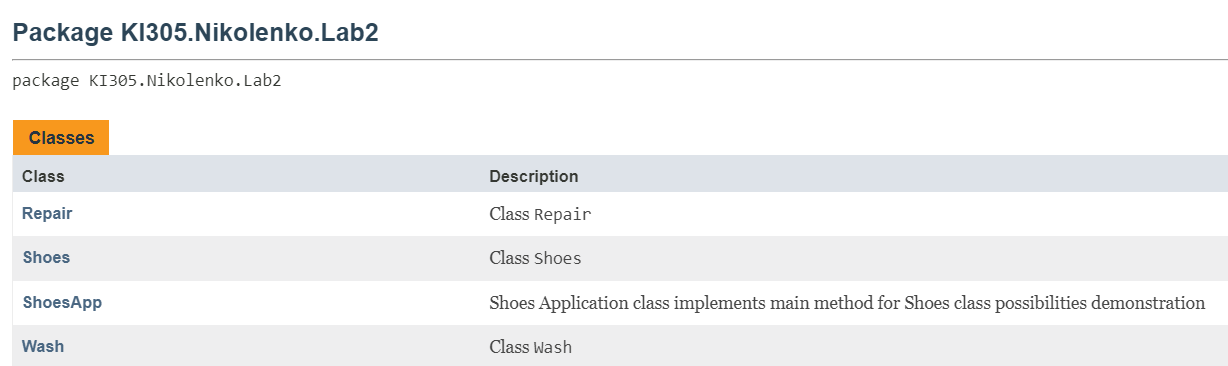
****

*Рис.1.Результат виконання програми (вигляд консолі)*

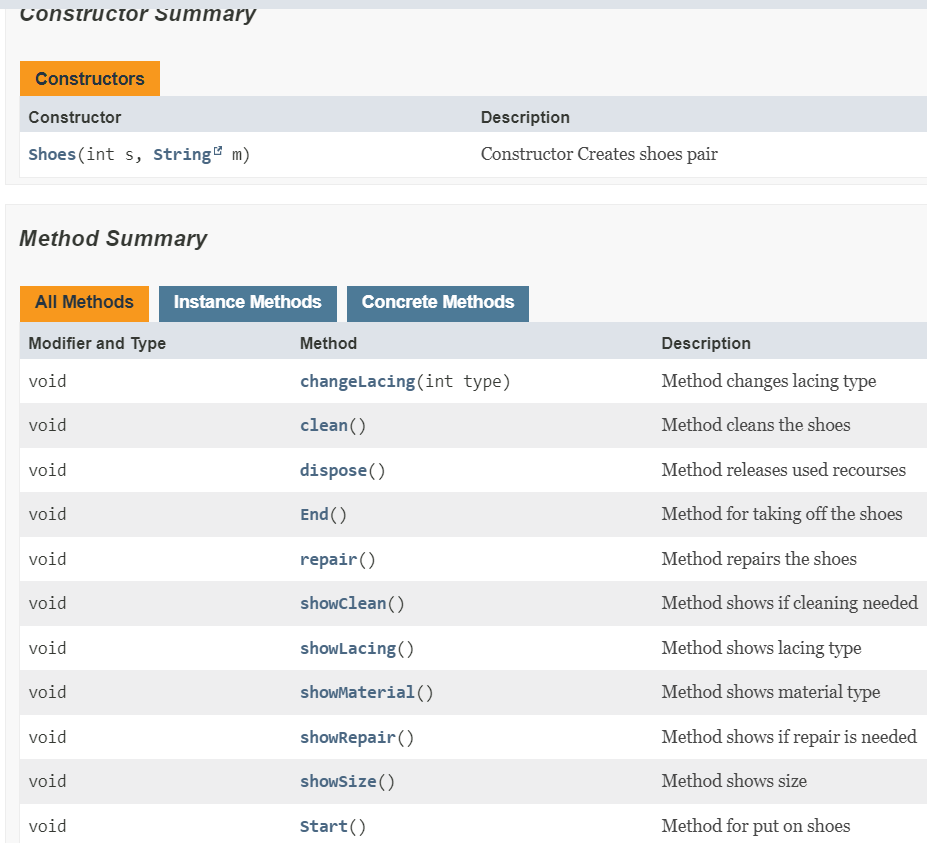


*Рис.2.Результат виконання програми (вивід у файл)*

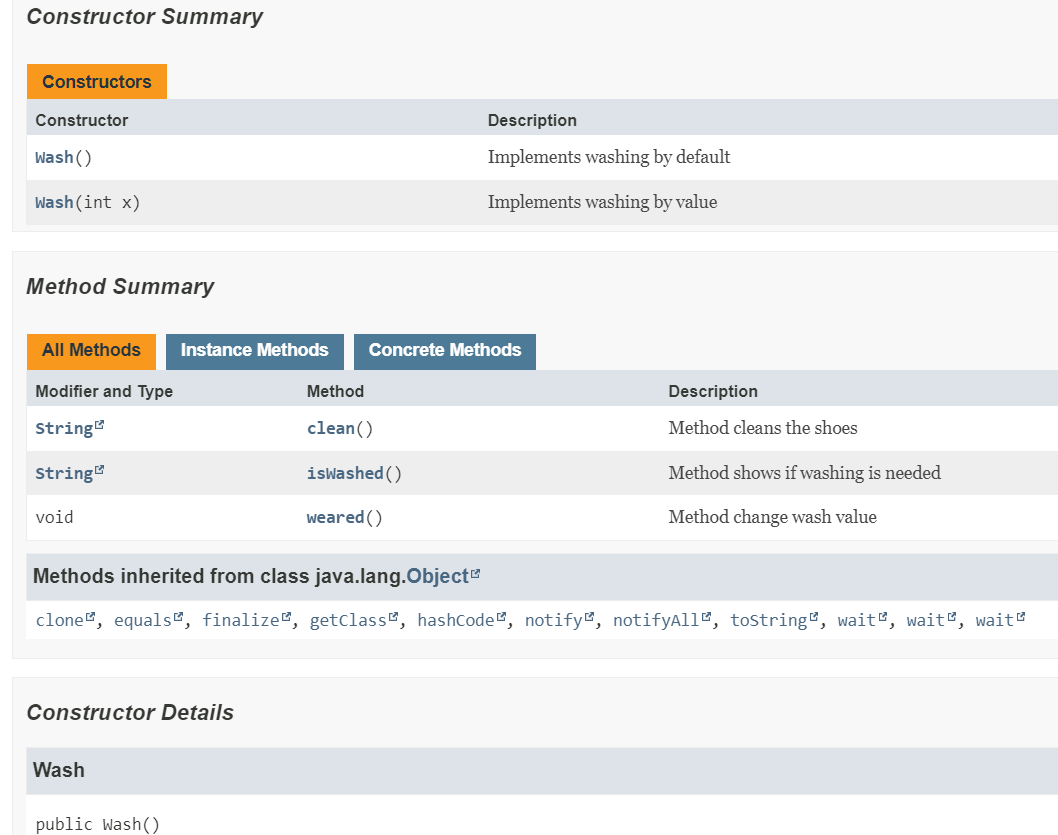
**Згенерована документація:**

****

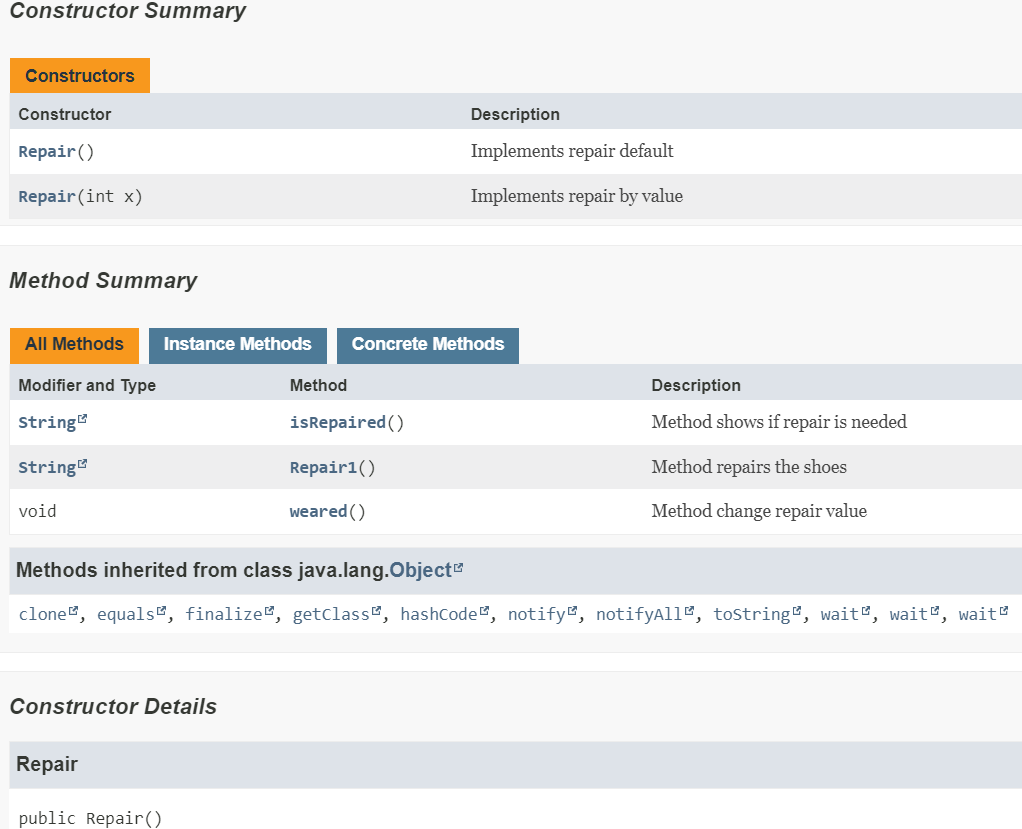
*Рис.3.Вміст вкладки Package*

****

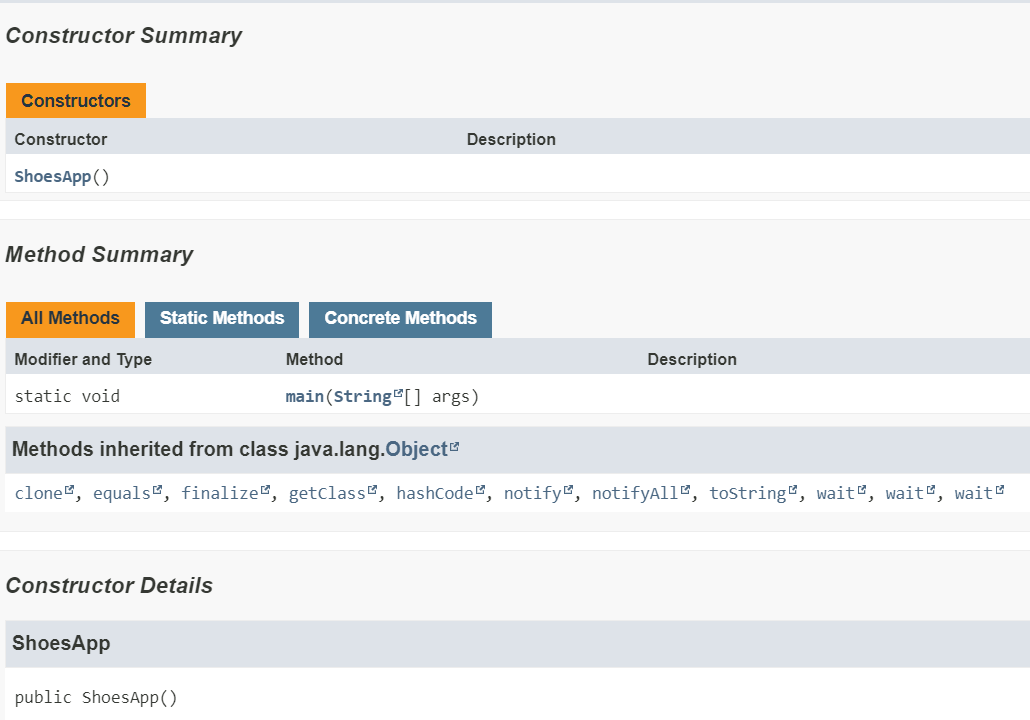
*Рис.4.Вміст вкладки Сlass(Shoes)*

****

*Рис.5.Вміст вкладки Сlass(Wash)*

****

*Рис.6.Вміст вкладки Сlass(Repair)*

****

*Рис.6.Вміст вкладки Сlass(ShoesApp)*

**Відповіді на контрольні запитання:**

1. **Синтаксис визначення класу?**

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

[public] class НазваКласу

{

[конструктори]

[методи]

[поля]

}

1. **Синтаксис визначення методу?**

Синтаксис оголошення методу наступний:

[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]

{

[Тіло методу]

[return [значення]];

}

1. **Синтаксис оголошення поля?**

Синтаксис оголошення поля наступний:

[СпецифікаторДоступу] [static] [final] Тип НазваПоля [=ПочатковеЗначення];

1. **Як оголосити та ініціалізувати константне поле?**

Синтаксис оголошення та ініціалізування константного поля наступний:

[СпецифікаторДоступу] [final] Тип НазваПоля [= ПочатковеЗначення];

1. **Які є способи ініціалізації полів?**

Ініціалізацію полів при створенні об’єкту можна здійснювати трьома способами:

* у конструкторі;
* явно при оголошені поля;
* у блоці ініціалізації (виконується перед виконанням конструктора).

1. **Синтаксис визначення конструктора?**

Синтаксис оголошення конструктора:

[СпецифікаторДоступу] НазваКласу([параметри])

{

Тіло конструктора

}

1. **Синтаксис оголошення пакету?**

Синтаксис оператора package:

package НазваПакету{.НазваПідпакету};

1. **Як підключити до програми класи, що визначені в зовнішніх пакетах?**

Доступ до класів з інших пакетів можна отримати двома шляхами:

1. вказуючи повне ім’я пакету перед іменем кожного класу.
2. використовуючи оператор import, що дозволяє підключати як один клас так і всі загальнодоступні класи пакету, позбавляючи необхідності записувати імена класів з вказуванням повної назви пакету перед ними.
3. **В чому суть статичного імпорту пакетів?**

Статичний імпорт дозволяє не вживати явно назву класу при звертанні до статичного поля або методу класу.

1. **Які вимоги ставляться до файлів і каталогів при використанні пакетів?**

Використання пакетів вимагає, щоб файли і каталоги проекту та їх ієрархія були строго структурованими. Так назви пакету і його підпакетів мають співпадати з назвами каталогів, де вони розміщуються. Назви загальнодоступних класів мають співпадати з назвами файлів, де вони розміщуються. Ієрархія каталогів і файлів проекту має співпадати з ієрархією пакетів. Після компіляції ієрархія каталогів, де містяться файли класів, співпадає з ієрархією каталогів проекту.

**Висновок:**

На цій лабораторній роботі я ознайомився з процесом розробки класів та пакетів мовою Java.