Міністерство освіти і науки України

Національний університет «Львівська Політехніка»

Кафедра ЕОМ

ЗВІТ

з лабораторної роботи №2

з дисципліни: “Кросплатформенні засоби програмування”

на тему: **«Класи та пакети»**

Варіант - **19**

Виконав:

ст. гр. КІ-305

Лихограй А. В.

Прийняв:

доцент кафедри ЕОМ

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 – «Принтер». 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.
2. Автоматично згенерувати документацію до розробленої програми. 3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.
4. Дати відповідь на контрольні запитання.

Вихідний код програми:

InkCartridge.java:

```
package KI305.lykhohrai.Lab2;
import java.util.Scanner;
/*
The class InkCartridge implements a printer cartridge.
@param ink Level nk level in the printer.
*/
```

```
public class InkCartridge {
    private int inkLevel;
    // The default constructor.
    public InkCartridge(){
        this.inkLevel = 0;
    }
    // The constructor with parameters.
    public InkCartridge(int inkLevel) {
        this.inkLevel = inkLevel;
    }
    // The method for entering a new amount of ink.
    public void AddInk() {
        Scanner in = new Scanner(System.in);
        System.out.print("Кількість нового чорнила: ");
        inkLevel = in.nextInt();
    }
    // The method for using ink.
    public void UseInk(int length) {
        if(length > 2000)
            inkLevel -= 3;
        else if (1000 > length)
            inkLevel -= 2;
        else
                inkLevel -= 1;
    }
    //Get ink level.
    public int GetInkLevel() { return inkLevel; }
    // Reset ink level.
    public void ResetInkLevel() { inkLevel = 0; }
    // Check if the cartridge is empty.
    public boolean isInkEmpty() {return (inkLevel == 0);}
    // Check if the cartridge is full.
   public boolean isInkFull() { return (inkLevel == 100); }
    // Set ink level.
    public void SetInkLevel(int level){ inkLevel = level; }
}
```

PaperTray.java:

```
package KI305.lykhohrai.Lab2;
import java.io.*;
import java.util.*;
/*
The method that implements the paper mechanism.
@param paperCount Amount of paper.
@param maxPaperLevel Max amount of paper.
 */
public class PaperTray  {
    private int paperCount;
    private int maxPaperLevel = 250;
    // The default constructor.
    public PaperTray()   {

        this.paperCount = 0;
    }
    // The constructor with parameters.
    public PaperTray(int count)
    {
        if(count <= maxPaperLevel)
            this.paperCount = count;
        else
            System.out.print("\nПаперу забагато!!!\n");
    }
    // The method for entering a new amount of paper.
    public void AddPaper() {
```

```
        Scanner in = new Scanner(System.in);
        System.out.print("Кількість нового паперу: ");
        int temp = in.nextInt();
        if(temp <= maxPaperLevel)
            paperCount = temp;
        else
            System.out.print("\nПаперу забагато!!!\n");
    }
    // The method for using paper.
    public void UsePaper() {paperCount--;}
    // Get paper count.
    public int GetPaperCount() {
        return paperCount;
    }
    // Reset amount of paper.
    public void ResetPaperCount() {
        paperCount = 0;
    }
    // Check if amount of paper = 0.
    public boolean isPaperEmpty() {
        return(paperCount == 0);
    }
    // Check if amount of paper is full/
    public boolean isPaperFull() {
        return(paperCount == maxPaperLevel);
    }
    // Set paper count.
    public void SetPaperCount(int count){ paperCount = count; }
}
```

NetworkConnection.java:

```
package KI305.lykhohrai.Lab2;
import java.util.Scanner;
/*
The method that implements the network connection.
@param ipAddress IP Address.
@param port.
@param connect Inet conection.
 */
public class NetworkConnection {
    private String ipAddress;
    private String port;
    private boolean connect;
    // The default constructor.
    public NetworkConnection(){
     setIpAddress();
     setPort();
    }
    // The constructor with parameters.
    public NetworkConnection(String ipAddress, String port) {
        this.ipAddress = ipAddress;
        this.port = port;
        connect = false;
    }
    // Connect network.
    public  boolean Connect(String address, String new_port) {
        connect = false;
        if(ipAddress.equals(address) && port.equals(new_port))
        {
            connect = true;
        }
        else
            System.out.print("Ip адрес або порт не співпадають!\n");
```

```
        return connect;
    }
    // Disconnect network.
    public void Disconnect() {
        connect = false;
    }
    // Check is connect.
    public boolean isConnected() {
        return (connect);
    }
    // Get IP address.
    public String getIpAddress() {
        return ipAddress;
    }
    // Get port.
    public String getPort() {
        return port;
    }
    // Set IP address.
    public void setIpAddress(){
        Scanner in = new Scanner(System.in);
        System.out.print("Введіть IP адрес: ");
        ipAddress = in.nextLine();
    }
    // Set port.
    public void setPort(){
        Scanner in = new Scanner(System.in);
        System.out.print("Введіть порт: ");
        port = in.nextLine();
    }
}
```

Printer.java:

```
package KI305.lykhohrai.Lab2;
import java.io.*;
import java.util.*;
/*
The method that implements the printer.
@param model Name of printer.
@param resource Resource work of printer.
@param work Is printer work.
 */
public class Printer {
    private String model;
    private int resourse;
    private boolean work;
    private PaperTray paperTray;
    private InkCartridge inkCartridge;
    private NetworkConnection networkConnection;

    // The default constructor.
    public Printer() {
        model = "unknow";
        resourse = 0;
        paperTray = new PaperTray();
        inkCartridge = new InkCartridge();
        networkConnection = new NetworkConnection();

    }

    // The constructor with parameters.
    public Printer(String model, InkCartridge inkCartridge, PaperTray
paperTray, NetworkConnection networkConnection, int resourse) {
        this.model = model;
        this.paperTray = paperTray;
```

```
        this.inkCartridge = inkCartridge;
        this.networkConnection = networkConnection;
        this.resourse = resourse;
    }
    // Print document.

    public void printDocument() {
        writeLogs("Print document.\n");
        Scanner in = new Scanner(System.in);
        System.out.print("Введіть назву документу: ");
        String document = in.nextLine();
        System.out.print("Введіть кількість копій: ");
        int count = in.nextInt();
        System.out.print("Введіть ваш IP адрес (178.212.111.75): ");
        String address;
        address = in.next();
        System.out.print("Введіть порт (1024): ");
        String port = in.next();
        if (!isWork() || (count == 0) ||
!networkConnection.Connect(address, port)) {
            System.out.print("Пристрій не ввімкнуто!!!\n");
            writeLogs("Printer has problem!\n");
        } else {
            System.out.print("Розпочато друк документу: "+ document+"\n");
            if (paperTray.GetPaperCount() >= count &&
inkCartridge.GetInkLevel() > 10) {
                for (int i = 1; i <= count; i++) {
                    System.out.print("\nДрук " + i + " сторінки. \n");
                    workPrinter(document);
                    useInkPaper(document);
                }
            } else
                System.out.print("Замало чорнила або паперу!\n");
        }
    }

    // Print text to the console.
    public void workPrinter(String document) {
        writeLogs("Printing document.\n");
        try {
            File file = new File(document);

            if (!file.exists()) {
                System.out.println("Файл не знайдено.");
                return;
            }
            FileReader fileReader = new FileReader(file);
            BufferedReader bufferedReader = new
BufferedReader(fileReader);

            String line;
            System.out.print("\n");
            while ((line = bufferedReader.readLine()) != null) {
                System.out.println("\t" + line);
            }
            bufferedReader.close();
        } catch (IOException e) {
            System.out.println("Виникла помилка при читанні файлу: " +
e.getMessage());
        }
    }

    // Count ink for printing.
    public int countInk(String text) {
        writeLogs("Count ink.\n");
        if (text == null || text.isEmpty()) {
```

```
            return 0;
        }
        return text.length();
    }

    // Use ink and paper.
    public void useInkPaper(String text) {
        writeLogs("Use ink and paper.\n");
        paperTray.UsePaper();
        inkCartridge.UseInk(countInk(text));
    }

    // Check if printer is work.
    public boolean isWork() {
        writeLogs("Is printer work?" + work + ".\n");
        return (work);
    }

    // Power on.
    public void powerOn() {
        writeLogs("Printer is on.\n");
        if (resourse == 0)
            System.out.print("Пристрій потрібно відкалібрувати!\n");
        else {
            System.out.print("Принтер ввімкнено!\n");
            work = true;
            resourse--;
        }
    }

    // Power off.
    public void powerOff() {
        writeLogs("Printer is off.\n");
        System.out.print("Принтер вимкнено! \n");
        work = false;
        networkConnection.Disconnect();
    }

    // Calibrate printer.
    public void calibratePrinter() {
        writeLogs("Calibrate printer\n");
        System.out.print("Принтер було відкалібровано! \n");
        inkCartridge.ResetInkLevel();
        paperTray.ResetPaperCount();
        resourse = 100;
    }

    // Clean printer heads.
    public void cleanPrintHeads() {
        writeLogs("Clear printer\n");
        paperTray.ResetPaperCount();
        inkCartridge.ResetInkLevel();
        resourse = 0;
        System.out.print("Принтер очищено! \n");
    }

    // Print "About printer".
    public void AboutPrinter() {
        writeLogs("About printer.\n");
        System.out.print("\n\nПро принтер:\n");
        System.out.print("Назва: " + GetPrinterName() + "\n");
        CheckPaperLevel();
        CheckInkLevel();
        System.out.print("IP адреса:"+networkConnection.getIpAddress()+
"\n");
    }
```

```
    // Check ink level.
    public void CheckInkLevel() {
        writeLogs("Check ink level.\n");
        System.out.print("Залишилось чорнила:" +
inkCartridge.GetInkLevel() + "\n");
    }

    // Check paper level.
    public void CheckPaperLevel() {
        writeLogs("Check paper level.\n");
        System.out.print("Залишилось паперу:" + paperTray.GetPaperCount()
+ "\n");
    }

    // Add paper.
    public void addPaper() {
        writeLogs("Add paper.\n");
        paperTray.AddPaper();
    }

    // Add ink.
    public void addInkLevel() {
        writeLogs("Add ink.\n");
        inkCartridge.AddInk();
    }

    // Set printer name.
    void SetPrinterName(String name) {
        writeLogs("Set printer name.\n");
        model = name;
    }

    // Get printer name.
    String GetPrinterName() {
        writeLogs("Get printer name.\n");
        return model;
    }

    // Write logs about work of printer.
    public void writeLogs(String textToWrite) {
        try (BufferedWriter bufferedWriter = new BufferedWriter(new
FileWriter("Logs.txt", true))) {
            // Записуємо переданий текст у файл
            bufferedWriter.write(textToWrite);
        } catch (IOException e) {
            System.err.println("Сталася помилка при записі до файлу: " +
e.getMessage());
        }
    }

    // Set max paper count.
    public void replacePaperTray(PaperTray newPaperTray) {
        newPaperTray.ResetPaperCount();
        newPaperTray.SetPaperCount(250);
    }

    // Set max ink leve.
    public void replaceInkCartridge(InkCartridge newInkCartridge) {
        newInkCartridge.ResetInkLevel();
        newInkCartridge.SetInkLevel(100);
    }
}
```

PrinterDriver.java

```
package KI305.lykhohrai.Lab2;
import java.io.*;
public class PrinterDriver {
    public static void main(String[] args) throws FileNotFoundException{
            Printer myPrinter = new Printer("HP2", new InkCartridge(100),
new PaperTray(2), new NetworkConnection(), 100);
            myPrinter.powerOn();
            myPrinter.AboutPrinter();
            myPrinter.printDocument();
            myPrinter.AboutPrinter();
            myPrinter.addPaper();
            myPrinter.printDocument();
            myPrinter.AboutPrinter();
        }
   }
```

Результат виконання програми:

```
"C:\Program Files\Java\jdk-21\bin\java.exe" "-javaagent:C
Введіть IP адрес: 178.212.111.75
Введіть порт: 1024
Принтер ввімкнено!


Про принтер:
Назва: HP2
Залишилось паперу: 2
Залишилось чорнила: 100
IP адреса: 178.212.111.75
Введіть назву документу: MyFile.txt
Введіть кількість копій: 3
Введіть ваш IP адрес (178.212.111.75): 178.212.111.75
Введіть порт (1024): 1024
Розпочато друк документу: MyFile.txt
Замало чорнила або паперу!


Про принтер:
Назва: HP2
Залишилось паперу: 2
Залишилось чорнила: 100
IP адреса: 178.212.111.75

Process finished with exit code 0
```

Рис. 1. Результат роботи програми.

Фрагмент згенерованої документації:

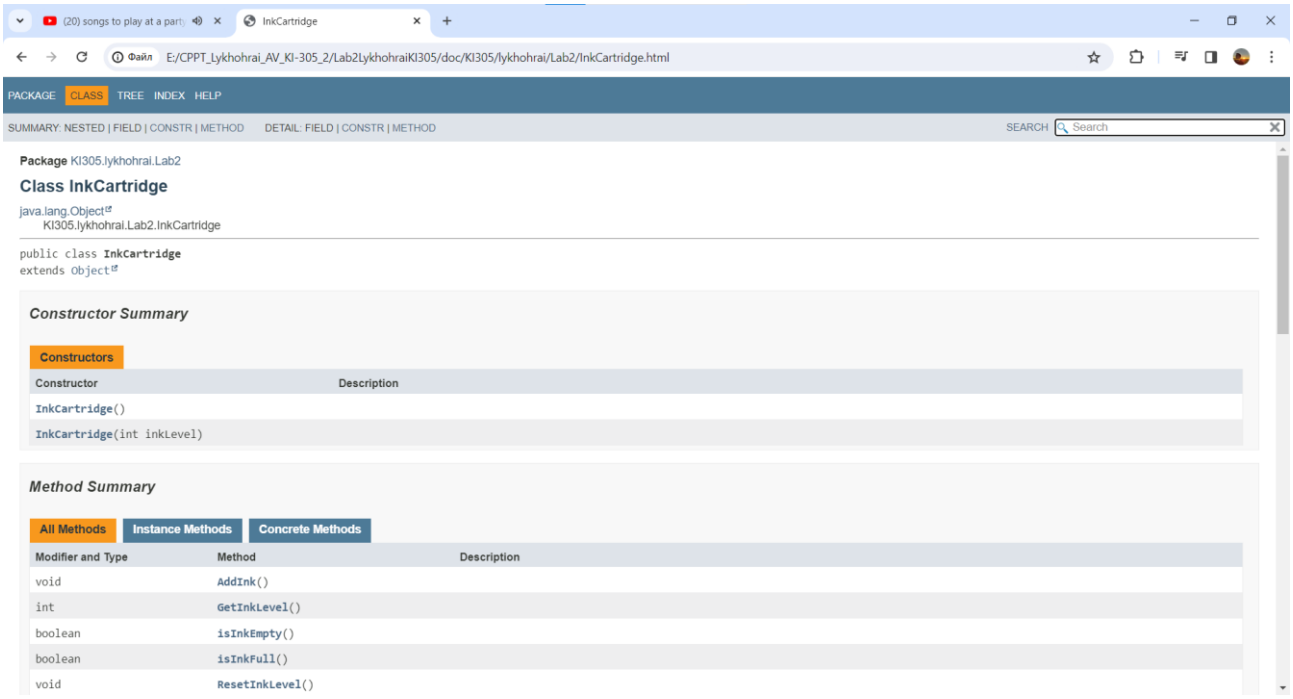

Рис. 2. Фрагмент документації.

Відповіді на контрольні запитання:

1. Синтаксис визначення класу:

```java
access_modifier class ClassName {
    // тіло класу
}
```

2. Синтаксис визначення методу:

```java
access_modifier return_type methodName(parameters) {
    // тіло методу
}
```

3. Синтаксис оголошення поля:

```java
access_modifier data_type fieldName;
```

4. Оголошення та ініціалізація константного поля:

```java
access_modifier static final data_type CONSTANT_NAME = value;
```

5. Способи ініціалізації полів:

- Пряма ініціалізація при оголошенні.
- У конструкторі класу.
- За допомогою блоку ініціалізації.

6. Синтаксис визначення конструктора:

```java
access_modifier ClassName(parameters) {
    // тіло конструктора
}
```

7. Синтаксис оголошення пакету:

```java
package package_name;
```

8. Підключення класів із зовнішніх пакетів:

- Використовуючи ключове слово `import`:

```java
import package_name.ClassName;
```

9. Суть статичного імпорту пакетів:

- Статичний імпорт дозволяє використовувати статичні члени класу без звертання до самого класу:

```java
import static package_name.ClassName.staticMember;
```

10. Вимоги до файлів і каталогів при використанні пакетів:

- Файли класів повинні бути розташовані у відповідних підкаталогах відносно кореневого каталогу пакету.

- Ім'я пакету повинно відповідати структурі каталогів, у якій вони знаходяться.

- Назви файлів і пакетів повинні збігатися.

Висновок: під час виконання лабораторної роботи я ознайомився з класами та пакетами мови Java. Навчився працювати з ними. Розробив програму, де є основний клас «Printer», який в свою чергу має три поля, які є деякими іншими класами: «PaperTray», «InkCatridge», «NetworkConnection».