Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 2

з дисципліни: «Кросплатформні засоби програмування»

на тему: «Класи та пакети»

**Виконав:**

студент групи *КІ-306*

*Савіцький Н. О.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2023

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

**Завдання (варіант № 19)**

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну область згідно варіанту (19. Принтер). Програма має задовольняти наступним вимогам:

• програма має розміщуватися в пакеті Група.Прізвище.Lab3;

• 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 розробленого пакету.
2. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації.
3. Дати відповідь на контрольні запитання

**Вихідний код програми**

**Файл PrinterApp.java**

package KI306.Savitskyi.Lab2;

import static java.lang.System.out;

import java.io.\*;

/\*\*

\* Клас, що містить метод `main` для демонстрації роботи класу `Printer`.

\*/

public class PrinterApp {

/\*\*

\* Головний метод програми, який виконує демонстрацію функціоналу класу `Printer`.

\*

\* @param args Масив аргументів командного рядка (не використовується в цьому випадку).

\*/

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

String model = "Canon MF3010";

boolean multicolour = true;

int inkLevel = 97;

// Створення об'єкта принтера з вказаними параметрами.

Printer printer = new Printer(model, multicolour, inkLevel);

// Виведення статусу принтера.

printer.status();

// Вставлення 14 аркушів паперу та виведення кількості залишених аркушів та рівня чернил.

printer.insertNPaperSheets(14);

out.print("Paper sheets inserted: " + printer.getPaperTrayCapacity() + "\n");

out.print("Current ink level: " + printer.getCartridgeInkLevel() + "\n");

// Увімкнення принтера, виконання друку та виведення стану паперу та рівня чернил.

printer.turnOn();

printer.print();

out.print("Paper sheets inserted: " + printer.getPaperTrayCapacity() + "\n");

out.print("Current ink level: " + printer.getCartridgeInkLevel() + "\n");

// Виконання сканування та друку, заповнення картриджу і виведення стану паперу та рівня чернил.

printer.scanAndPrint();

printer.fillEverything();

out.print("Paper sheets inserted: " + printer.getPaperTrayCapacity() + "\n");

out.print("Current ink level: " + printer.getCartridgeInkLevel() + "\n");

// Завершення роботи з принтером.

printer.dispose();

}

}

**Файл Printer.java**

/\*\*

\* Клас, що представляє принтер.

\*/

package KI306.Savitskyi.Lab2;

import java.io.\*;

/\*\*

\* Клас, що представляє принтер.

\*/

public class Printer {

/\*\*

\* Модель принтера.

\*/

private String model;

/\*\*

\* Стан принтера (увімкнено або вимкнено).

\*/

private boolean isOn;

/\*\*

\* Картридж для принтера.

\*/

private Cartridge cartridge;

/\*\*

\* Лоток для паперу.

\*/

private PaperTray paperTray;

/\*\*

\* Сканер для принтера.

\*/

private Scanner scanner;

/\*\*

\* Потік для запису журналу подій.

\*/

private PrintWriter fout;

/\*\*

\* Конструктор за замовчуванням.

\*

\* @throws FileNotFoundException виникає, якщо не вдається створити файл для журналу.

\*/

public Printer() throws FileNotFoundException {

model = "unknown";

isOn = false;

cartridge = new Cartridge();

paperTray = new PaperTray();

scanner = new Scanner();

fout = new PrintWriter(new File("Log.txt"));

}

/\*\*

\* Конструктор з параметрами.

\*

\* @param mod Модель принтера.

\* @param multicolour Чи підтримується друк кольоровими чернилами.

\* @param ink Рівень чернил у картриджі.

\* @throws FileNotFoundException виникає, якщо не вдається створити файл для журналу.

\*/

public Printer(String mod, boolean multicolour, int ink) throws FileNotFoundException {

model = mod;

isOn = false;

cartridge = new Cartridge(multicolour, ink);

paperTray = new PaperTray();

scanner = new Scanner();

fout = new PrintWriter(new File("Log.txt"));

}

/\*\*

\* Увімкнути принтер.

\*/

public void turnOn() {

isOn = true;

fout.print("Printer was turned on successfully.\n");

}

/\*\*

\* Вимкнути принтер.

\*/

public void turnOff() {

isOn = false;

fout.print("Printer was turned off successfully.\n");

}

/\*\*

\* Вставити певну кількість аркушів паперу в лоток.

\*

\* @param n Кількість аркушів, яку потрібно вставити.

\*/

public void insertNPaperSheets(int n) {

if (paperTray.getPaperCount() + n <= paperTray.getMaxPaperCount()) {

for (int i = 0; i < n; i++) {

paperTray.insertPaper();

}

fout.print(n + " sheets of paper were inserted successfully.\n");

} else {

fout.print("Unable to insert " + n + " sheets of paper.\n");

}

}

/\*\*

\* Роздрукувати документ.

\*/

public void print() {

if (isOn == true) {

if (cartridge.getInkLevel() != 0 && paperTray.getPaperCount() != 0) {

paperTray.extractPaper();

cartridge.decreaseInkLevel();

fout.print("Printing was successful!\n");

} else {

fout.print("Print failed!\n");

}

} else {

fout.print("Print failed!\n");

}

fout.print("Sheets of paper remaining: " + paperTray.getPaperCount() + "\nInk level: " + cartridge.getInkLevel() + "\n");

}

/\*\*

\* Наповнює картридж новими чернилами та вставляє максимальну кількість паперу в лоток.

\*/

public void fillEverything() {

cartridge.putNewCartridge(true);

paperTray.insertAllPaper();

fout.print("New cartridge was put, paper tray was filled.\n");

}

/\*\*

\* Видаляє картридж та вилучає всі аркуші паперу з лотка.

\*/

public void removeEverything() {

cartridge.removeCartridge();

paperTray.extractAllPaper();

fout.print("Cartridge was removed, paper tray is empty.\n");

}

/\*\*

\* Виводить інформацію про стан принтера у файл журналу.

\*/

public void status() {

fout.print("\n\*\*\*" + model + " - status \*\*\*\n" +

"Is on: " + isOn + "\n" +

"Able to print multicolour: " + cartridge.getIsMulticolour() + "\n" +

"Ink level: " + cartridge.getInkLevel() + "\n" +

"Paper tray max capacity: " + paperTray.getMaxPaperCount() + "\n" +

"Number of inserted sheets of paper: " + paperTray.getPaperCount() + "\n" +

"Scanner DPI: " + scanner.getDPI() + "\n" +

"Able to scan multicolour: " + scanner.getIsMulticolour() + "\n\n"

);

}

/\*\*

\* Виконує сканування та записує результат у файл журналу.

\*/

public void scan() {

fout.print("Scanning was successful!\n");

}

/\*\*

\* Виконує сканування та друк документа, викликаючи методи scan() та print().

\*/

public void scanAndPrint() {

scan();

print();

}

/\*\*

\* Вилучає певну кількість аркушів паперу з лотка.

\*

\* @param n Кількість аркушів для вилучення.

\*/

public void extractNPaperSheets(int n) {

if (paperTray.getPaperCount() - n >= 0) {

for (int i = 0; i < n; i++) {

paperTray.extractPaper();

}

fout.print(n + " sheets of paper were extracted successfully.\n");

} else {

fout.print("Unable to extract " + n + " sheets of paper.\n");

}

}

/\*\*

\* Отримує поточну кількість аркушів у лотку для паперу.

\*

\* @return Кількість аркушів у лотку.

\*/

public int getPaperTrayCapacity() {

return paperTray.getPaperCount();

}

/\*\*

\* Отримує рівень чернил у картриджі.

\*

\* @return Рівень чернил у картриджі.

\*/

public int getCartridgeInkLevel() {

return cartridge.getInkLevel();

}

/\*\*

\* Завершує роботу з принтером та закриває файл журналу.

\*/

public void dispose() {

fout.close();

}

}

/\*\*

\* Клас, що представляє картридж для принтера.

\*/

class Cartridge {

/\*\*

\* Поле, що вказує, чи підтримується друк кольоровими чернилами.

\*/

private boolean isMulticolour;

/\*\*

\* Поле, що зберігає рівень чернил у картриджі.

\*/

private int inkLevel;

/\*\*

\* Конструктор за замовчуванням, створює картридж з чорними чернилами та повним рівнем.

\*/

public Cartridge() {

isMulticolour = false;

inkLevel = 100;

}

/\*\*

\* Конструктор з параметрами, створює картридж з вказаним типом чернил і рівнем.

\*

\* @param multicolour Тип чернил (кольорові або чорні).

\* @param ink Рівень чернил у картриджі.

\*/

public Cartridge(boolean multicolour, int ink) {

isMulticolour = multicolour;

inkLevel = ink;

}

/\*\*

\* Отримати рівень чернил у картриджі.

\*

\* @return Рівень чернил у картриджі.

\*/

public int getInkLevel() {

return inkLevel;

}

/\*\*

\* Перевірити, чи підтримується друк кольоровими чернилами.

\*

\* @return true, якщо підтримується друк кольоровими чернилами, в іншому випадку - false.

\*/

public boolean getIsMulticolour() {

return isMulticolour;

}

/\*\*

\* Зменшити рівень чернил у картриджі на одиницю.

\*/

public void decreaseInkLevel() {

inkLevel--;

}

/\*\*

\* Видалити картридж і встановити рівень чернил на нуль.

\*/

public void removeCartridge() {

isMulticolour = false;

inkLevel = 0;

}

/\*\*

\* Встановити новий картридж з вказаним типом чернил та повним рівнем.

\*

\* @param multicolour Тип чернил (кольорові або чорні).

\*/

public void putNewCartridge(boolean multicolour) {

isMulticolour = multicolour;

inkLevel = 100;

}

}

/\*\*

\* Клас, що представляє лоток для паперу.

\*/

class PaperTray {

/\*\*

\* Кількість аркушів паперу в лотку.

\*/

private int paperCount;

/\*\*

\* Максимальна кількість аркушів, яку може містити лоток.

\*/

private int maxPaperCount;

/\*\*

\* Конструктор за замовчуванням, створює лоток без паперу і з максимальною кількістю аркушів.

\*/

public PaperTray() {

paperCount = 0;

maxPaperCount = 35;

}

/\*\*

\* Конструктор з параметрами, створює лоток з вказаною кількістю аркушів і максимальною кількістю аркушів.

\*

\* @param paperN Початкова кількість аркушів у лотку.

\* @param maxPaperN Максимальна кількість аркушів, яку може містити лоток.

\*/

public PaperTray(int paperN, int maxPaperN) {

paperCount = paperN;

maxPaperCount = maxPaperN;

}

/\*\*

\* Вставити один аркуш паперу у лоток, якщо не досягнуто максимальну кількість.

\*/

public void insertPaper() {

if (paperCount < maxPaperCount) {

paperCount++;

}

}

/\*\*

\* Вилучити один аркуш паперу із лотка.

\*/

public void extractPaper() {

paperCount--;

}

/\*\*

\* Вставити максимальну кількість аркушів у лоток.

\*/

public void insertAllPaper() {

paperCount = maxPaperCount;

}

/\*\*

\* Вилучити всі аркуші паперу із лотка (зробити лоток порожнім).

\*/

public void extractAllPaper() {

paperCount = 0;

}

/\*\*

\* Отримати поточну кількість аркушів у лотку для паперу.

\*

\* @return Кількість аркушів у лотку.

\*/

public int getPaperCount() {

return paperCount;

}

/\*\*

\* Отримати максимальну кількість аркушів, яку може містити лоток.

\*

\* @return Максимальна кількість аркушів, яку може містити лоток.

\*/

public int getMaxPaperCount() {

return maxPaperCount;

}

}

/\*\*

\* Клас, що представляє сканер для принтера.

\*/

class Scanner {

/\*\*

\* Роздільна здатність сканера (DPI - dots per inch).

\*/

private int DPI;

/\*\*

\* Поле, що вказує, чи підтримується сканування кольорових зображень.

\*/

private boolean isMulticolour;

/\*\*

\* Конструктор за замовчуванням, створює сканер із роздільною здатністю 1600 DPI та підтримкою кольорових зображень.

\*/

public Scanner() {

DPI = 1600;

isMulticolour = true;

}

/\*\*

\* Конструктор з параметрами, створює сканер із вказаною роздільною здатністю і підтримкою кольорових зображень.

\*

\* @param dpi Роздільна здатність сканера (DPI).

\* @param multicolour Підтримка кольорових зображень (true - підтримується, false - не підтримується).

\*/

public Scanner(int dpi, boolean multicolour) {

DPI = dpi;

isMulticolour = multicolour;

}

/\*\*

\* Встановити максимальну роздільну здатність сканера (2400 DPI).

\*/

public void setMaxDPI() {

DPI = 2400;

}

/\*\*

\* Встановити мінімальну роздільну здатність сканера (300 DPI).

\*/

public void setMinDPI() {

DPI = 300;

}

/\*\*

\* Перевірити, чи підтримується сканування кольорових зображень.

\*

\* @return true, якщо підтримується сканування кольорових зображень, в іншому випадку - false.

\*/

public boolean getIsMulticolour() {

return isMulticolour;

}

/\*\*

\* Отримати поточну роздільну здатність сканера (DPI).

\*

\* @return Роздільна здатність сканера (DPI).

\*/

public int getDPI() {

return DPI;

}

}

**Результат виконання програми**

Консоль:

Paper sheets inserted: 14

Current ink level: 97

Paper sheets inserted: 13

Current ink level: 96

Paper sheets inserted: 35

Current ink level: 100

Log.txt:

\*\*\*Canon MF3010 - status \*\*\*

Is on: false

Able to print multicolour: true

Ink level: 97

Paper tray max capacity: 35

Number of inserted sheets of paper: 0

Scanner DPI: 1600

Able to scan multicolour: true

14 sheets of paper were inserted successfully.

Printer was turned on successfully.

Printing was successful!

Sheets of paper remaining: 13

Ink level: 96

Scanning was successful!

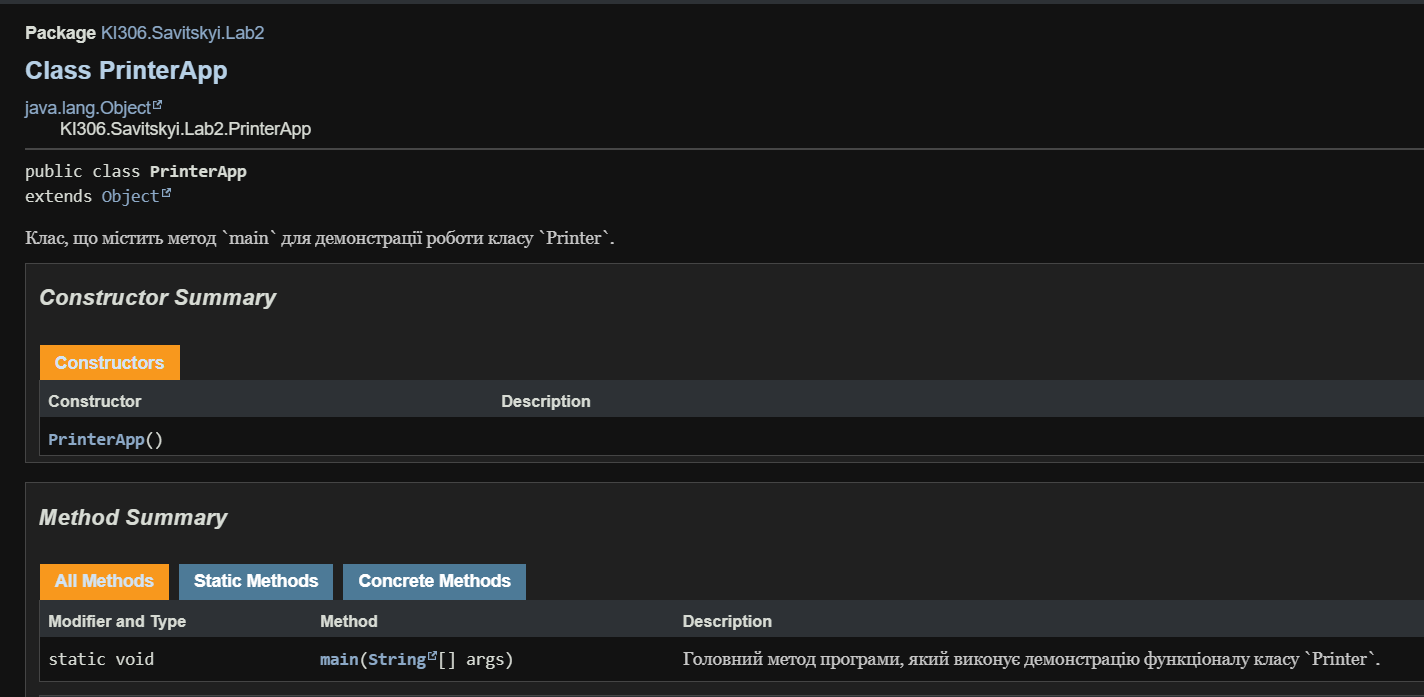
Printing was successful!

Sheets of paper remaining: 12

Ink level: 95

New cartridge was put, paper tray was filled.

**Фрагмент згенерованої документації**

**

**Відповіді на контрольні запитання**

1. Синтаксис визначення класу.

* public class ClassName {

// Class members (fields, methods, constructors)

}

1. Синтаксис визначення методу.

* public returnType methodName(parameters) {

// Method body

}

1. Синтаксис оголошення поля.

* accessModifier dataType fieldName;

1. Як оголосити та ініціалізувати константне поле?

* public static final dataType CONSTANT\_NAME = initial\_value;

1. Які є способи ініціалізації полів?

* Явна ініціалізація при оголошенні поля.
* Ініціалізація у конструкторі класу.
* Ініціалізація у блоку ініціалізації (конструкторі, статичному або звичайному).

1. Синтаксис визначення конструктора.

* public ClassName(parameters) {

// Constructor body

}

1. Синтаксис оголошення пакету.

* package packageName.subpackage;

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

* Вказати повне ім'я класу перед використанням (наприклад, java.util.Date today = new java.util.Date();).
* Використовувати оператор import для підключення класів з інших пакетів, щоб уникнути повторення повного імені класу.

1. В чому суть статичного імпорту пакетів?

* Статичний імпорт дозволяє підключити статичні методи і поля класів без повного імені класу.
* Завдяки статичному імпорту, можна використовувати статичні члени класу, не додаваючи перед ними ім'я класу.

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

* Назви пакетів повинні відповідати структурі каталогів.
* Назви загальнодоступних класів повинні співпадати з назвами файлів, де вони розміщені.
* Після компіляції ієрархія каталогів проекту повинна відповідати ієрархії пакетів.
* Для компіляції та запуску програми слід використовувати шляхи до файлів та пакетів.

**Висновок**

У ході виконання даної лабораторної роботи, я отримав цінні навички розробки класів та пакетів у мові програмування Java. Ця лабораторна робота надала мені можливість ознайомитися з базовими конструкціями Java, такими як оголошення класів, методів та полів. Я навчився правильно структурувати свій код, визначати доступ до класів та їх членів, а також використовувати модифікатори доступу для керування видимістю.