Міністерство освіти і науки України
Національний університет «Львівська політехніка»

Звіт

до лабораторної роботи №3

З дисципліни: «Кросплатформенні засоби програмування»

На тему: «Класи та пакети»

Виконав:

Студент групи КІ-34

Сенета Б.Р.

Львів 2022

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

Виконання роботи

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

Завдання:

ВАРІАНТИ ЗАВДАННЬ

1. Людина
2. Космічний корабель
3. Пес
4. Кіт
5. Машина
6. Літак
7. Комп'ютер
8. Фотоапарат
9. Рослина
10. Будинок
11. Монітор
12. Водойма
13. Телефон
14. Телевізор
15. Корабель
16. Аудіоплеєр
17. Відеоплеєр
18. Сканер
19. Принтер
20. Взуття
21. Пістолет
22. Автомат
23. Плитка для приготування їжі
24. Спорядження альпініста
25. Кондиціонер
26. Шлюпка на веслах
27. Патрон
28. Лампочка
29. Протигаз
30. Локомотив

Клас Main:

```
import KI34.Seneta.Lab3.Printer;

public class App {
    public static void main(String[] args) throws Exception {
        Printer Printer = new Printer();
        Printer.connectPowerCordConnector();
        Printer.connectUSB();
        Printer.turnOnColoredPrint();
        Printer.turnOnStartButton();
        Printer.putSmthOnTable();
        Printer.Printning();
        Printer.disconnectPowerCordConnector();
        Printer.dispose();
    }
}
```

Клас Basin:

```
/**
 * lab 3 package
 */
package KI34.Seneta.Lab3;

import java.io.*;

/**
 * Class <code>Printer</code> implements computer mouse
 *
 * @author Seneta Bohdan
 * @version 1.0
 */
public class Printer {
    private PrinterButton PrinterButton;
    private PrinterTablet PrinterTablet;
    private PrinterPort PrinterPort;
    private PrinterMatrix PrinterMatrix;
    private PrintWriter fout;

    /**
     * Constructor
     *
     * @throws FileNotFoundException
     */
    public Printer() throws FileNotFoundException {
        PrinterButton = new PrinterButton();
        PrinterTablet = new PrinterTablet();
        PrinterPort = new PrinterPort();
        PrinterMatrix = new PrinterMatrix();
```

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

    /**
     * Constructor
     *
     * @throws FileNotFoundException
     */

    public Printer(boolean colored) throws FileNotFoundException {
        PrinterButton = new PrinterButton(colored);
        PrinterTablet = new PrinterTablet();
        PrinterPort = new PrinterPort();
        PrinterMatrix = new PrinterMatrix();
        fout = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Method releases used recourses
     */
    public void dispose() {
        fout.close();
    }

    /**
     * Method implements turning on Printer
     */
    public void turnOnPrinter() {
        if (PrinterPort.get_powerCordConnector()) {
            PrinterButton.set_Power(true);
            if (PrinterButton.get_Power()) {
                System.out.print("The Printer is on\n");
                fout.print("The Printer is on\n");
                fout.flush();
            } else {
                System.out.print("The Printer isn't on\n");
                fout.print("The Printer isn't on\n");
                fout.flush();
            }
        }
    }

    /**
     * Method implements turning off Printer
     */
    public void turnOffPrinter() {
        if (!PrinterPort.get_powerCordConnector()) {
            PrinterButton.set_Power(false);
            if (!PrinterButton.get_Power()) {
                System.out.print("The Printer isn't on\n");
                fout.print("The Printer isn't on\n");
                fout.flush();
            } else {
```

```
                System.out.print("The Printer is on\n");
                fout.print("The Printer is on\n");
                fout.flush();
            }
        }
    }

    /**
     * Method implements turn on colored Printning
     */
    public void turnOnColoredPrint() {
        PrinterButton.set_Colored(true);
        if (PrinterButton.get_Colored()) {
            System.out.print("The colored Print is on\n");
            fout.print("The colored Print is on\n");
            fout.flush();
        } else {
            System.out.print("The colored Print isn't on\n");
            fout.print("The colored Print isn't on\n");
            fout.flush();
        }
    }

    /**
     * Method implements turn off colored Printning
     */
    public void turnOffColoredPrint() {
        PrinterButton.set_Colored(false);
        if (PrinterButton.get_Colored()) {
            System.out.print("The colored Print is on\n");
            fout.print("The colored Print is on\n");
            fout.flush();
        } else {
            System.out.print("The colored Print isn't on\n");
            fout.print("The colored Print isn't on\n");
            fout.flush();
        }
    }

    /**
     * Method implements turn on Start button
     */
    public void turnOnStartButton() {
        PrinterButton.set_Start(true);
        if (PrinterButton.get_Start()) {
            System.out.print("The start button is pressed\n");
            fout.print("The start button is pressed\n");
            fout.flush();
        } else {
            System.out.print("The start button isn't pressed\n");
            fout.print("The start button isn't pressed\n");
            fout.flush();
        }
```

```
    }

    /**
     * Method implements turn off Start button
     */
    public void turnOffStartButton() {
        PrinterButton.set_Start(false);
        if (PrinterButton.get_Start()) {
            System.out.print("The start button is pressed\n");
            fout.print("The start button is pressed\n");
            fout.flush();
        } else {
            System.out.print("The start button isn't pressed\n");
            fout.print("The start button isn't pressed\n");
            fout.flush();
        }
    }

    /**
     * Method implements putting something on Printer table
     * to Print it
     */
    public void putSmthOnTable() {
        PrinterTablet.set_Tablet(true);
        System.out.print("Something put on table\n");
        fout.print("Something put on table\n");
        fout.flush();
    }

    /**
     * Method implements check if something is on the table
     */
    public boolean canWePrint() {
        return PrinterTablet.get_Tablet();
    }

    /**
     * Method implements connecting USB
     */
    public void connectUSB() {
        PrinterPort.set_USB(true);
        if (PrinterPort.get_USB()) {
            System.out.print("The USB is connected\n");
            fout.print("The USB is connected\n");
            fout.flush();
        } else {
            System.out.print("The USB isn't connected\n");
            fout.print("The USB isn't connected\n");
            fout.flush();
        }
    }

    /**
```

```
     * Method implements disconnecting USB
     */
    public void disconnectUSB() {
        PrinterPort.set_USB(false);
        if (PrinterPort.get_USB()) {
            System.out.print("The USB isn't disconnected\n");
            fout.print("The USB isn't disconnected\n");
            fout.flush();
        } else {
            System.out.print("The USB is disconnected\n");
            fout.print("The USB is disconnected\n");
            fout.flush();
        }
    }

    /**
     * Method implements connecting IEEE 1394
     */
    public void connectIEEE1394() {
        PrinterPort.set_IEEE1394(true);
        if (PrinterPort.get_IEEE1394()) {
            System.out.print("The IEEE1394 PORT is connected\n");
            fout.print("The IEEE1394 PORT is connected\n");
            fout.flush();
        } else {
            System.out.print("The IEEE1394 PORT isn't connected\n");
            fout.print("The IEEE1394 PORT isn't connected\n");
            fout.flush();
        }
    }

    /**
     * Method implements disconnecting IEEE 1394
     */
    public void disconnectIEEE1394() {
        PrinterPort.set_IEEE1394(false);
        if (PrinterPort.get_IEEE1394()) {
            System.out.print("The IEEE1394 PORT isn't disconnected\n");
            fout.print("TThe IEEE1394 PORT isn't disconnected\n");
            fout.flush();
        } else {
            System.out.print("The IEEE1394 PORT is disconnected\n");
            fout.print("The IEEE1394 PORT is disconnected\n");
            fout.flush();
        }
    }

    /**
     * Method implements connecting Additional Boards
     */
    public void connectAdditionalBoards() {
        PrinterPort.set_connectorForAdditionalBoards(true);
        if (PrinterPort.get_connectorForAdditionalBoards()) {
```

```
            System.out.print("The AdditionalBoards is connected\n");
            fout.print("The AdditionalBoards is connected\n");
            fout.flush();
        } else {
            System.out.print("The AdditionalBoards isn't connected\n");
            fout.print("The AdditionalBoards isn't connected\n");
            fout.flush();
        }
    }

    /**
     * Method implements disconnecting Additional Boards
     */
    public void disconnectAdditionalBoards() {
        PrinterPort.set_connectorForAdditionalBoards(false);
        if (PrinterPort.get_connectorForAdditionalBoards()) {
            System.out.print("AdditionalBoards isn't disconnected\n");
            fout.print("AdditionalBoards isn't disconnected\n");
            fout.flush();
        } else {
            System.out.print("AdditionalBoards is disconnected\n");
            fout.print("AdditionalBoards is disconnected\n");
            fout.flush();
        }
    }

    /**
     * Method implements connecting Power Cord
     */
    public void connectPowerCordConnector() {
        PrinterPort.set_powerCordConnector(true);
        if (PrinterPort.get_powerCordConnector()) {
            System.out.print("The Power Cord is connected\n");
            fout.print("The Power Cord is connected\n");
            fout.flush();
        } else {
            System.out.print("The Power Cord isn't connected\n");
            fout.print("The Power Cord isn't connected\n");
            fout.flush();
        }
    }

    /**
     * Method implements disconnecting Power Cord
     */
    public void disconnectPowerCordConnector() {
        PrinterPort.set_powerCordConnector(false);
        if (PrinterPort.get_powerCordConnector()) {
            System.out.print("The Power Cord isn't disconnected\n");
            fout.print("The Power Cord isn't disconnected\n");
            fout.flush();
        } else {
            System.out.print("The Power Cord is disconnected\n");
```

```
            fout.print("The Power Cord is disconnected\n");
            fout.flush();
        }
    }

    /**
     * Method implements Printning
     */
    public void Printning() {
        if (PrinterPort.get_powerCordConnector()) {
            if (PrinterButton.get_Power()) {
                if (PrinterButton.get_Start()) {
                    if (PrinterTablet.get_Tablet()) {
                        if (PrinterButton.get_Colored()) {
                            PrinterMatrix.Printned(true);
                            System.out.print("Colored Printning ...\n");
                            System.out.print("Printned\n");
                            fout.print("Colored Printning ...\n");
                            fout.print("Printned\n");
                            fout.flush();
                        } else {
                            PrinterMatrix.Printned(true);
                            System.out.print("White\\Black Printning ...\n");
                            System.out.print("Printned\n");
                            fout.print("White\\Black Printning ...\n");
                            System.out.print("Printned\n");
                            fout.flush();
                        }
                    } else {
                        System.out.print("Nothing to Print try again\n");
                        fout.print("Nothing to Print try again\n");
                        fout.flush();
                    }
                } else {
                    System.out.print("Start button isn't pressed\n");
                    fout.print("Start button isn't pressed\n");
                    fout.flush();
                }
            } else {
                System.out.print("Power button isn't plugged in\n");
                fout.print("Power button isn't plugged in\n");
                fout.flush();
            }
        } else {
            System.out.print("Power isn't plugged in\n");
            fout.print("Power isn't plugged in\n");
            fout.flush();
        }
    }
}

class PrinterButton {
    private boolean isStart;
```

```
    private boolean isColored;
    private boolean isPower;

    /**
     * Constructor default
     */
    public PrinterButton() {
        isStart = false;
        isColored = false;
        isPower = true;
    }

    /**
     * Constructor with three parameters
     */
    public PrinterButton(boolean setStart, boolean setColored, boolean setPower) {
        isStart = setStart;
        isColored = setColored;
        isPower = setPower;
    }

    /**
     * Constructor with one parameter
     */
    public PrinterButton(boolean setColored) {
        isColored = setColored;
    }

    /**
     * Method sets start button
     */
    public void set_Start(boolean setStart) {
        isStart = setStart;
    }

    /**
     * Method sets colored button
     */
    public void set_Colored(boolean setColored) {
        isColored = setColored;
    }

    /**
     * Method sets power button
     */
    public void set_Power(boolean setPower) {
        isPower = setPower;
    }

    /**
     * Method sets start button
     */
    public boolean get_Start() {
```

```
        return isStart;
    }

    /**
     * Method get colored button
     */
    public boolean get_Colored() {
        return isColored;
    }

    /**
     * Method get power button
     */
    public boolean get_Power() {
        return isPower;
    }
}

class PrinterTablet {
    private boolean isOnTablet;

    /**
     * Constructor default
     */
    public PrinterTablet() {
        isOnTablet = false;
    }

    /**
     * Method sets tablet
     */
    public void set_Tablet(boolean sOnTablet) {
        isOnTablet = sOnTablet;
    }

    /**
     * Method get tablet
     */
    public boolean get_Tablet() {
        return isOnTablet;
    }
}

class PrinterPort {
    private boolean USB;
    private boolean IEEE1394;
    private boolean connectorForAdditionalBoards;
    private boolean powerCordConnector;

    /**
     * Constructor default
     */
    public PrinterPort() {
```

```
        USB = false;
        IEEE1394 = false;
        connectorForAdditionalBoards = false;
        powerCordConnector = false;
    }

    /**
     * Method sets USB connection
     */
    public void set_USB(boolean sUSB) {
        USB = sUSB;
    }

    /**
     * Method sets IEEE 1394 connection
     */
    public void set_IEEE1394(boolean sIEEE1394) {
        IEEE1394 = sIEEE1394;
    }

    /**
     * Method sets Additional Boards connection
     */
    public void set_connectorForAdditionalBoards(boolean
sConnectorForAdditionalBoards) {
        connectorForAdditionalBoards = sConnectorForAdditionalBoards;
    }

    /**
     * Method sets Cord Connector connection
     */
    public void set_powerCordConnector(boolean sPowerCordConnector) {
        powerCordConnector = sPowerCordConnector;
    }

    /**
     * Method get USB connection
     */
    public boolean get_USB() {
        return USB;
    }

    /**
     * Method get IEEE 1394 connection
     */
    public boolean get_IEEE1394() {
        return IEEE1394;
    }

    /**
     * Method get Additional Boards connection
     */
    public boolean get_connectorForAdditionalBoards() {
```

```
        return connectorForAdditionalBoards;
    }

    /**
     * Method get Cord connection
     */
    public boolean get_powerCordConnector() {
        return powerCordConnector;
    }
}

class PrinterMatrix {
    AnalogDigitalDevice ADD;
    private boolean isTransformed;

    /**
     * Constructor default
     */
    public PrinterMatrix() {
        isTransformed = false;
        ADD = new AnalogDigitalDevice();
    }

    /**
     * Method sets transformation
     */
    public void set_transform(boolean sTransform) {
        isTransformed = sTransform;
    }

    /**
     * Method get transformation
     */
    public boolean get_transform() {
        return isTransformed;
    }

    /**
     * Method implement Printning
     */
    public boolean Printned(boolean run) {
        if (run == true) {
            ADD.set_convert(true);
            return true;
        }
        return false;
    }

}

class AnalogDigitalDevice {
    private boolean isConverted;
```

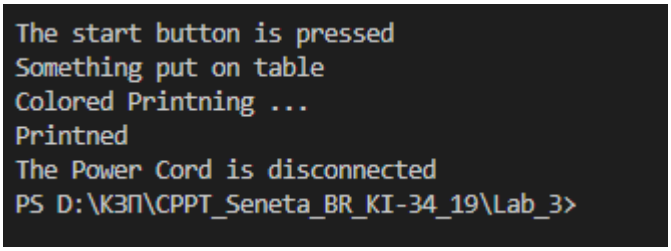
```java
    /**
     * Constructor default
     */
    public AnalogDigitalDevice() {
        isConverted = false;
    }

    /**
     * Method sets convert
     */
    public void set_convert(boolean sConvert) {
        isConverted = sConvert;
    }

    /**
     * Method get convert
     */
    public boolean get_convert() {
        return isConverted;
    }

}
```

Txt-файл з записаною інформацією:

Log.txt

```
1   The Power Cord is connected
2   The USB is connected
3   The colored Print is on
4   The start button is pressed
5   Something put on table
6   Colored Printning ...
7   Printned
8   The Power Cord is disconnected
9
```

Результат виконання



Відповіді на КЗ:

1. [public] class Назва Класу {
   [Методи]
   [Змінні]
   [Поля]
   [Конструктори]
   }
2. [СпецифікаторДоступу] Тип назваМетоду([параметри]) [throws класи] {
   [Тіло методу] [return [значення]];
   }
3. [СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];
4. Використати [final], тобто/наприклад private final int i;
5. Ініціалізацію полів при створенні об’єкту можна здійснювати трьома способами:
   • у конструкторі;
   • явно при оголошені поля;
   • у блоці ініціалізації (виконується перед виконанням конструктора).

Якщо поле не ініціалізується жодним з цих способів, то йому присвоюється значення за замовчуванням.

6. [СпецифікаторДоступу] НазваКласу([параметри]) {
   Тіло конструктора
   }

7. package НазваПакету{.НазваПідпакету};

8. Клас може використовувати всі класи з власного пакету і всі загальнодоступні класи з інших пакетів. 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.

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

10. Файл, каталоги повинні бути строго структурованими. Чітка ієрархія, назви пакетів та підпакетів повинні співпадати з назвами каталогів де вони розміщуються.

Висновок: створено класи, розроблено методи та конструктори до об’єктів.