

Міністерство освіти і науки України
Національний університет «Львівська політехніка»



Звіт

до лабораторної роботи №4

З дисципліни: «Кросплатформенні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконав:

Студент групи КІ-34

Романів В. А.

Прийняв:

Іванов Ю. С.

Львів 2022

Мета: ознайомитися зі спадкуванням та інтерфейсами на мові Java.

Виконання роботи

ЗАВДАННЯ

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab4 та 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:

18. Копіювальний апарат

Клас CopierApp

```
package KI34.Romaniv.Lab4;

import java.io.FileNotFoundException;

public class CopierApp {
    public static void main(String[] args) throws FileNotFoundException {
        Copier copier = new Copier();
        copier.connectPowerCordConnector();
        copier.turnOnStartButton();
        copier.turnOnScanner();
        copier.putSmthOnTable();
        copier.set_copy(true);
        copier.startCopping(true);
        copier.dispose();
        copier.turnOffScanner();
        copier.disconnectPowerCordConnector();
    }
}
```

Клас Copier

```
package KI34.Romaniv.Lab4;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.PrintWriter;

public class Copier extends Scanner implements Copy{

    private boolean turnOnCopy;
    private boolean isCopied;

    private PrintWriter fout1;

    public Copier() throws FileNotFoundException {
        super();
        fout1 = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Method releases used recourses
     */
    public void dispose()
    {
        fout1.close();
    }

    void set_copy(boolean s_cpy){
        this.turnOnCopy = s_cpy;
    }

    boolean get_copy(){
        return this.turnOnCopy;
    }

    void set_coppeid(boolean s_c) {
        this.isCopied = s_c;
    }

    @Override
    public boolean isCopied() {
        return isCopied;
    }
}
```

```

@Override
public void startCopping(boolean s_c) {
    scanning();
    if(get_copy()){
        System.out.print("Start copping...\n");
        System.out.print("Coppied\n");
        set_coppeid(s_c);
        fout1.print("Start copping...\n");
        fout1.print("Coppied\n");
        fout1.flush();
    }
}
}

```

Клас Scanner

```

package KI34.Romaniv.Lab4;

/**
 * lab 3 package
 */
import java.io.*;

/**
 * Class <code>Scanner</code> implements computer mouse
 * @author Romaniv Vitalii
 * @version 1.0
 */
abstract class Scanner{
    private ScannerButton scannerButton;
    private ScannerTablet scannerTablet;
    private ScannerPort scannerPort;
    private ScannerMatrix scannerMatrix;
    private PrintWriter fout;

    /**
     * Constructor
     * @throws FileNotFoundException
     */
    public Scanner() throws FileNotFoundException
    {
        scannerButton = new ScannerButton();
        scannerTablet = new ScannerTablet();
    }
}

```

```

        scannerPort = new ScannerPort();
        scannerMatrix = new ScannerMatrix();
        fout = new PrintWriter(new File("Log.txt"));
    }

    /**
     * Constructor
     * @throws FileNotFoundException
     */

    public Scanner(boolean colored) throws FileNotFoundException
    {
        scannerButton = new ScannerButton(colored);
        scannerTablet = new ScannerTablet();
        scannerPort = new ScannerPort();
        scannerMatrix = new ScannerMatrix();
        fout = new PrintWriter(new File("Log.txt"));
    }

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

    /**
     * Method implements turning on Scanner
     */
    public void turnOnScanner(){
        if(scannerPort.get_powerCordConnector()) {
            scannerButton.set_Power(true);
            if (scannerButton.get_Power()) {
                System.out.print("The scanner is on\n");
                fout.print("The scanner is on\n");
                fout.flush();
            } else {
                System.out.print("The scanner isn't on\n");
                fout.print("The scanner isn't on\n");
                fout.flush();
            }
        }
    }
}

```

```

/**
 * Method implements turning off Scanner
 */
public void turnOffScanner(){
    if(!scannerPort.get_powerCordConnector()) {
        scannerButton.set_Power(false);
        if (!scannerButton.get_Power()) {
            System.out.print("The scanner isn't on\n");
            fout.print("The scanner isn't on\n");
            fout.flush();
        } else {
            System.out.print("The scanner is on\n");
            fout.print("The scanner is on\n");
            fout.flush();
        }
    }
}

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

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

```

```

        fout.print("The colored scan isn't on\n");
        fout.flush();
    }
}

/**
 * Method implements turn on Start button
 */
public void turnOnStartButton() {
    scannerButton.set_Start(true);
    if(scannerButton.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(){
    scannerButton.set_Start(false);
    if(scannerButton.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 scanner table
 * to scan it
 */
public void putSmthOnTable(){
    scannerTablet.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 canWeScan(){
        return scannerTablet.get_Tablet();
    }

    /**
     * Method implements connecting USB
     */
    public void connectUSB(){
        scannerPort.set_USB(true);
        if(scannerPort.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(){
        scannerPort.set_USB(false);
        if(scannerPort.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(){
        scannerPort.set_IEEE1394(true);
        if(scannerPort.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(){
        scannerPort.set_IEEE1394(false);
        if(scannerPort.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(){
        scannerPort.set_connectorForAdditionalBoards(true);
        if(scannerPort.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(){
    scannerPort.set_connectorForAdditionalBoards(false);
    if(scannerPort.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() {
    scannerPort.set_powerCordConnector(true);
    if(scannerPort.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() {
    scannerPort.set_powerCordConnector(false);
    if(scannerPort.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 scanning
 */
public void scanning(){
    if(scannerPort.get_powerCordConnector()){
        if(scannerButton.get_Power()){
            if(scannerButton.get_Start()){
                if(scannerTablet.get_Tablet()){
                    if(scannerButton.get_Colored()){
                        scannerMatrix.Scanned(true);
                        System.out.print("Colored scanning ...\n");
                        System.out.print("Scanned\n");
                        fout.print("Colored scanning ...\n");
                        fout.print("Scanned\n");
                        fout.flush();
                    }else {
                        scannerMatrix.Scanned(true);
                        System.out.print("White\\Black scanning ...\n");
                        System.out.print("Scanned\n");
                        fout.print("White\\Black scanning ...\n");
                        System.out.print("Scanned\n");
                        fout.flush();
                    }
                }else{
                    System.out.print("Nothing to scan try again\n");
                    fout.print("Nothing to scan 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 ScannerButton{
    private boolean isStart;
    private boolean isColored;
    private boolean isPower;

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

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

    /**
     * Constructor with one parameter
     */
    public ScannerButton(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 ScannerTablet{
    private boolean isOnTablet;

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

```

```

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

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

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

    /**
     * Constructor default
     */
    public ScannerPort(){
        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 ScannerMatrix{
    AnalogDigitalDevice ADD;
    private boolean isTransformed;

    /**
     * Constructor default
     */
    public ScannerMatrix(){
        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 scanning
     */
    public boolean Scanned(boolean run){
        if(run == true){
            ADD.set_convert(true);
            return true;
        }
        return false;
    }
}

class AnalogDigitalDevice{
    private boolean isConverted;

    /**
     * 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;
}
}

```

Інтерфейс Copy

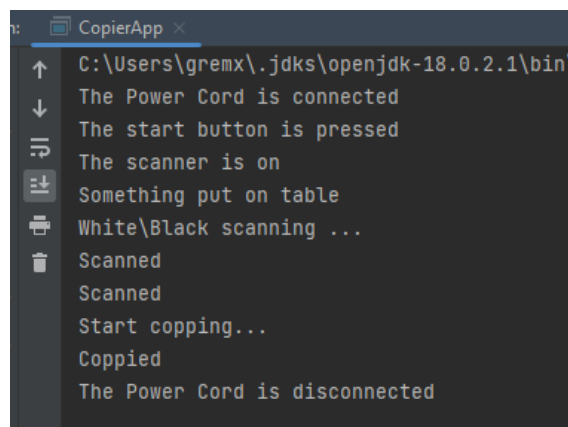
```

package KI34.Romaniv.Lab4;

public interface Copy {
    boolean isCopied();
    void startCopping(boolean s_c);
}

```

Результат виконання програми:

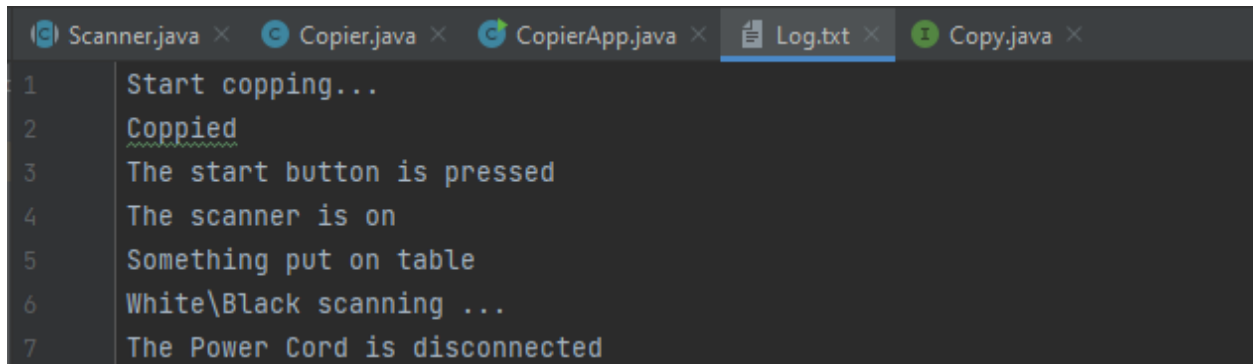


```

C:\Users\greml\jdk\openjdk-18.0.2.1\bin
The Power Cord is connected
The start button is pressed
The scanner is on
Something put on table
White\Black scanning ...
Scanned
Scanned
Start copping...
Copied
The Power Cord is disconnected

```

Створено файл з записом у нього даних:



```
Scanner.java x Copier.java x CopierApp.java x Log.txt x Copy.java x
1 Start copping...
2 Copied
3 The start button is pressed
4 The scanner is on
5 Something put on table
6 White\Black scanning ...
7 The Power Cord is disconnected
```

Відповіді на КЗ:

1. class Підклас extends Суперклас { Додаткові поля і методи }
2. Найчастіше супер-клас – це базовий клас, а підклас – це похідний клас від суперкласу.
3. Для звернення до методів чи полів суперкласу з підкласу потрібно використати ключове слово super. super.назваМетоду([параметри]); // виклик методу суперкласу super.назваПоля // звертання до поля суперкласу
4. Статичне зв'язування використовується при поліморфізмі. (компіляція). Лише тоді, коли метод є приватним, статичним або конструктором.
5. Поліморфізм реалізується за допомогою механізму динамічного (пізнього) зв'язування, який полягає у тому, що вибір методу, який необхідно викликати, відбувається не на етапі компіляції, а під час виконання програми.
6. Абстрактні класи призначені бути основою для розробки ієрархій класів та не дозволяють створювати об'єкти свого класу. Вони реалізуються за допомогою ключового слова abstract. На відміну від звичайних класів абстрактні класи можуть містити абстрактні методи (а можуть і не містити).
7. Використовується для визначення типу об'єкта в момент виконання програми. Посилання на базовий клас.
8. При наслідуванні у Java дозволяється перевизначення (перевантаження) методів та полів. При цьому область видимості методу, що перевизначається, має бути не меншою, ніж область видимості цього методу у суперкласі, інакше компілятор видасть повідомлення, про обмеження привілеїв доступу до даних. Перевизначення методу полягає у визначенні у підкласі методу з

сигнатурою методу суперкласу. При виклику такого методу з-під об'єкта підкласу викличеться метод цього підкласу. Якщо ж у підкласі немає визначеного методу, що викликається, то викличеться метод суперкласу. Якщо ж у суперкласі даний метод також відсутній, то згенерується повідомлення про помилку.

9. Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

10. Синтаксис оголошення інтерфейсів:

```
[public] interface НазваІнтерфейсу {  
    Прототипи методів та оголошення констант інтерфейсу  
}
```

Висновок: створено інтерфейс, абстрактний клас та реалізовано наслідування