Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 3

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

**Виконав:**

студент групи *КІ-306*

*Савіцький Н. О.*

**Прийняв:**

доцент кафедри ЕОМ

*Іванов Ю. С.*

Львів – 2023

**Мета роботи:** ознайомитися з спадкуванням та інтерфейсами у мові Java.

**Завдання (варіант № 19)**

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 (багатофункціональний пристрій) 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

**Вихідний код програми**

**Файл MFPApp.java**

package KI306.Savitskyi.Lab3;

import java.io.\*;

/\*\*

\* Клас представляє програму для управління багатофункціональним пристроєм MFP (МФП).

\* Він включає MFP, виконує різні операції та вимикає його.

\*/

public class MFPApp {

/\*\*

\* Головний метод програми, який викликається при запуску.

\* Виконує послідовність дій з MFP.

\*

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

\* **@throws** FileNotFoundException Виникає, якщо не вдається знайти файл.

\*/

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

// Створення об'єкта MFP з можливістю використовувати факс та електронну пошту.

MFP mfp = new MFP(true, true);

// Увімкнення MFP.

mfp.turnOn();

// Заповнення MFP папером та тонером.

mfp.fillEverything();

// Відправлення факсу за вказаним номером.

mfp.sendByFax("766435");

// Відправлення електронного листа на вказану адресу.

mfp.sendByEmail("example@organisation.com");

// Вимкнення MFP.

mfp.turnOff();

// Звільнення ресурсів, пов'язаних з MFP.

mfp.dispose();

}

}

**Файл Printer.java**

/\*\*

\* Клас, що представляє принтер.

\*/

package KI306.Savitskyi.Lab3;

import java.io.\*;

/\*\*

\* Клас, що представляє принтер.

\*/

public abstract 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.

\*/

public 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;

}

}

**Файл MFP.java**

package KI306.Savitskyi.Lab3;

import java.io.FileNotFoundException;

/\*\*

\* Інтерфейс, що представляє собою можливість відправки документів.

\*/

interface Sendable {

/\*\*

\* Метод для відправки документу факсом за вказаним номером телефону.

\*

\* **@param** phoneNumber Номер телефону, на який відправляється факс.

\*/

void sendByFax(String phoneNumber);

/\*\*

\* Метод для відправки документу електронною поштою на вказаний адресу електронної пошти.

\*

\* **@param** email Адреса електронної пошти, на яку відправляється документ.

\*/

void sendByEmail(String email);

}

/\*\*

\* Клас MFP (Multi-Function Printer), який представляє собою багатофункціональний принтер і є реалізацією інтерфейсу Sendable.

\*/

public class MFP extends Printer implements Sendable {

/\*\*

\* Конструктор за замовчуванням для MFP.

\*

\* **@throws** FileNotFoundException Виняток, який виникає, якщо не вдається знайти файл для запису.

\*/

public MFP() throws FileNotFoundException {

faxable = true;

mailable = false;

}

/\*\*

\* Конструктор MFP з можливістю налаштування параметрів факсу та електронної пошти.

\*

\* **@param** fax Параметр, який вказує, чи підтримує MFP факс.

\* **@param** mail Параметр, який вказує, чи підтримує MFP відправку електронною поштою.

\* **@throws** FileNotFoundException Виняток, який виникає, якщо не вдається знайти файл для запису.

\*/

public MFP(boolean fax, boolean mail) throws FileNotFoundException {

faxable = fax;

mailable = mail;

}

/\*\*

\* Метод для відправки документу факсом за вказаним номером телефону.

\*

\* **@param** phoneNumber Номер телефону, на який відправляється факс.

\*/

public void sendByFax(String phoneNumber) {

if (faxable == true) {

super.scan();

super.fout.print("The fax was successfully sent to the number: " + phoneNumber + "\n");

} else {

super.fout.print("The MFP is unable to send fax.\n");

}

}

/\*\*

\* Метод для відправки документу електронною поштою на вказаний адресу електронної пошти.

\*

\* **@param** email Адреса електронної пошти, на яку відправляється документ.

\*/

public void sendByEmail(String email) {

if (mailable == true) {

super.scan();

super.fout.print("The fax was successfully sent to the email: " + email + "\n");

} else {

super.fout.print("The MFP is unable to send email.\n");

}

}

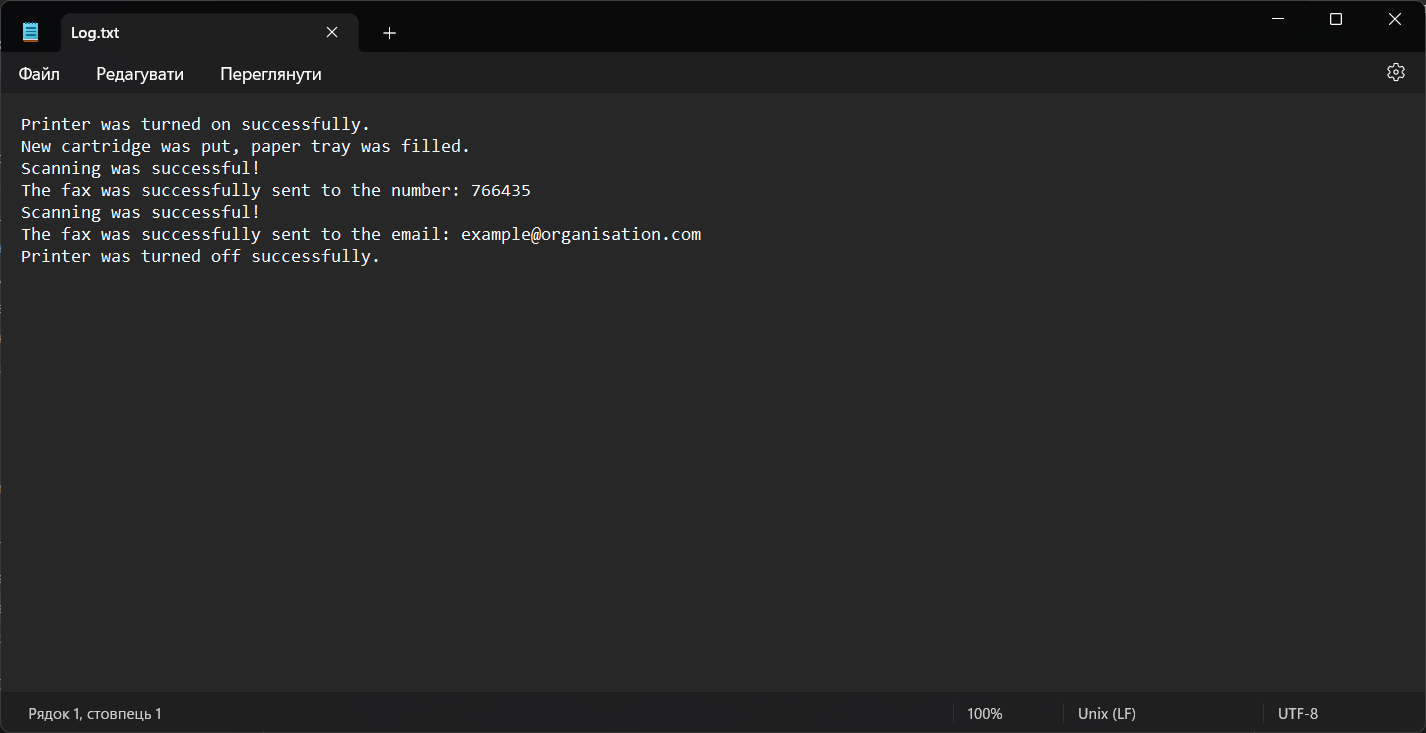
private boolean faxable;

private boolean mailable;

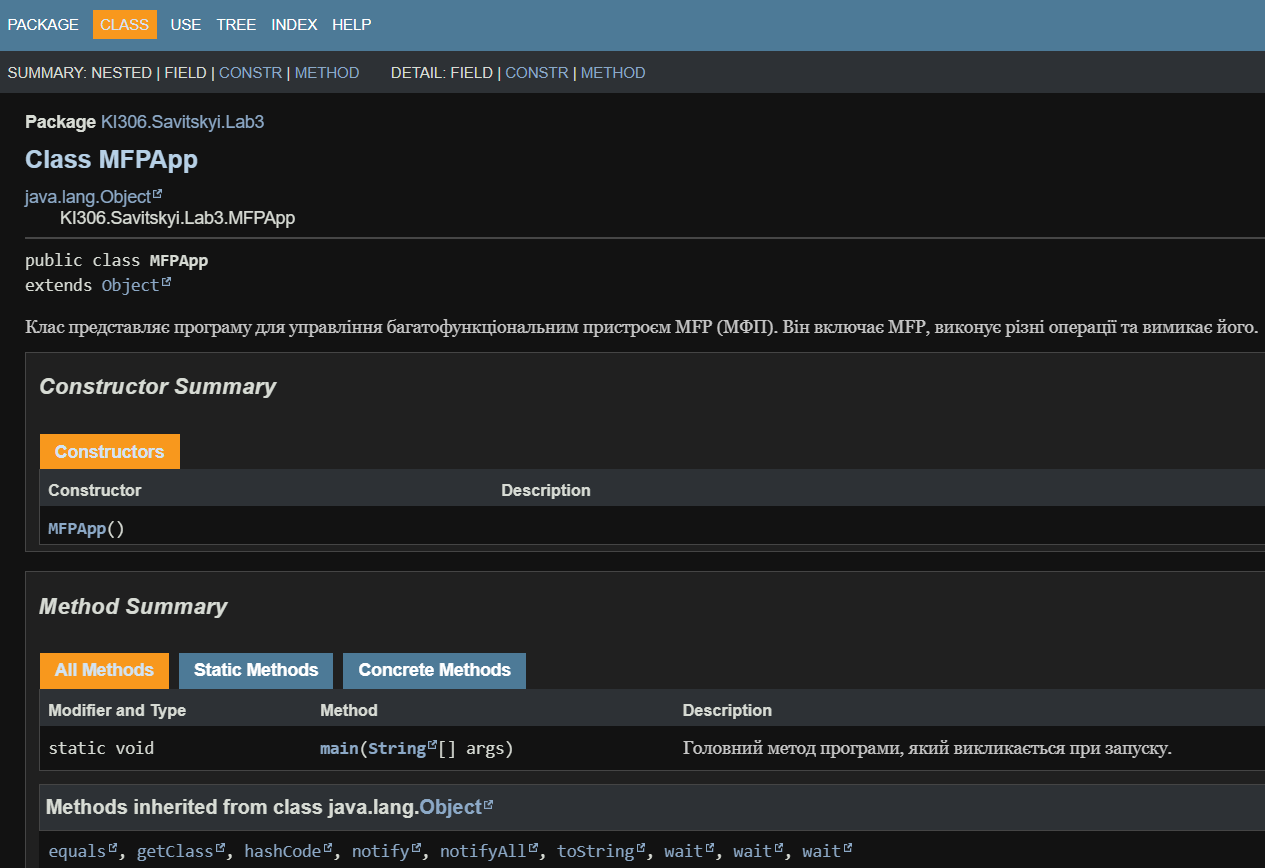
}

**Результат виконання програми**

Log.txt:



**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Синтаксис реалізації спадкування.

* class МійКлас implements Інтерфейс {

// тіло класу

}

1. Що таке суперклас та підклас?

* суперклас - це клас, від якого інший клас успадковує властивості та методи. Підклас - це клас, який успадковує властивості та методи від суперкласу.

1. Як звернутися до членів суперкласу з підкласу?

* super.назваМетоду([параметри]); // виклик методу суперкласу

super.назваПоля; // звернення до поля суперкласу

1. Коли використовується статичне зв’язування при виклику методу?

* Статичне зв’язування використовується, коли метод є приватним, статичним, фінальним або конструктором. В таких випадках вибір методу відбувається на етапі компіляції.

1. Як відбувається динамічне зв’язування при виклику методу?

* вибір методу для виклику відбувається під час виконання програми на основі фактичного типу об’єкта.

1. Що таке абстрактний клас та як його реалізувати?

* це клас, який має один або більше абстрактних методів (методів без реалізації). Щоб створити абстрактний клас, використовується ключове слово abstract. Приклад:

abstract class АбстрактнийКлас {

abstract void абстрактнийМетод();

}

1. Для чого використовується ключове слово instanceof?

* для перевірки, чи об'єкт належить до певного класу або інтерфейсу.

Синтаксис:

if (об'єкт instanceof Клас) {

// код, який виконується, якщо об'єкт належить до класу

}

1. Як перевірити чи клас є підкласом іншого класу?

* В Java використовується ключове слово extends, щоб вказати, що клас є підкласом іншого класу. Перевірити, чи один клас є підкласом іншого класу можна шляхом аналізу ієрархії успадкування.

1. Що таке інтерфейс?

* це абстрактний тип даних, який визначає набір методів, але не надає їх реалізацію. Всі методи інтерфейсу є загальнодоступними та автоматично є public. Інтерфейси використовуються для створення контрактів, які класи повинні реалізувати.

1. Як оголосити та застосувати інтерфейс?

* Для оголошення інтерфейсу використовується ключове слово interface. Синтаксис:

interface Інтерфейс {

// оголошення методів та констант

}

* Для застосування інтерфейсу в класі використовується ключове слово implements.

Синтаксис:

class МійКлас implements Інтерфейс {

// реалізація методів інтерфейсу

}

**Висновок**

У ході виконання даної лабораторної роботи, я отримав навички роботи з концепціями спадкування та інтерфейсами в мові програмування Java. Ознайомившись з цими важливими аспектами об'єктно-орієнтованого програмування, я зрозумів їх роль у створенні більш структурованих і гнучких програм.