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*

*Рудь Володимир*

**Прийняв:**

доцент кафедри ЕОМ

Іванов Ю. С.

**Львів – 2023**

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

**Завдання ( Варіант 18 Копіювальний апарат):**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі No3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі No3, 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.

1. Дати відповідь на контрольні запитання.

**Вихідний код програми**

**Результат виконання програми**

import ModulesScanBar.BarCode;  
import ModulesScanBar.Camera;  
import ModulesScanBar.Display;  
  
*/\*\*  
 \* Bar Code Scanner Application class implements main method for BarCodeScanner  
 class possibilities demonstration  
 \* @author Rud Volodymyr  
 \* @version 1.0  
 \*/*public class ScannerApplication {  
 */\*\*  
 \* @param args  
 \*/* public static void main(String[] args) {  
  
 Display d1 = new Display();  
 Camera c1 = new Camera(1000);  
 ScanBar barScanner1 = new ScanBar(d1,c1);  
 CopyPrinter copyPrinter1 = new CopyPrinter(d1,c1);  
 barScanner1.ScanBarCode("123456789015");  
 barScanner1.ScanBarCode("987654321030");  
 barScanner1.ScanBarCode("000000000000");  
  
 System.*out*.println();  
  
 copyPrinter1.ScanBarCode("123456789015");  
 copyPrinter1.ScanBarCode("987654321030");  
 barScanner1.ScanBarCode("000000000000");  
  
 System.*out*.println(barScanner1.GetInfo());  
 System.*out*.println(copyPrinter1.GetInfo());  
  
  
 }  
}

import ModulesScanBar.Camera;  
import ModulesScanBar.Display;  
  
public class ScanBar extends BarCodeScanner implements ShowInfoBarCodeScanners {  
  
 public ScanBar()  
 {  
 super();  
 }  
 public ScanBar(Display display, Camera camera) {  
 super(display,camera);  
 }  
  
 public boolean ScanBarCode(String code)  
 {  
 int[] temp = new int[12];  
 if(camera.captureImage()) {  
 if (code.length() != 12) {  
 display.noCodeRecognise();  
 return false;  
 }  
 for (int i = 0; i < 12; i++) {  
 temp[i] = Character.*getNumericValue*(code.charAt(i));  
 }  
 if (!scannedBarCode.setBarCode(temp)) {  
 display.WriteBarCodeScanResult(false, code);  
 return false;  
 }  
 else{  
 display.WriteBarCodeScanResult(true, code);  
 return true;  
 }  
  
 }  
 else {  
 display.noCodeRecognise();  
 return false;  
 }  
 }  
  
 @Override  
 public String GetInfo() {  
 return "This is ScanBar class";  
 }  
}

import ModulesScanBar.Camera;  
import ModulesScanBar.Display;  
  
import java.util.List;  
  
public class CopyPrinter extends ScanBar implements ShowInfoBarCodeScanners{  
 private List<String> scannedBarCodes;  
  
 public CopyPrinter(){  
 super();  
 }  
 public CopyPrinter(Display display, Camera camera)  
 {  
 super(display, camera);  
 }  
 @Override  
 public boolean ScanBarCode(String code) {  
 if(super.ScanBarCode(code))  
 {  
 scannedBarCodes.add(code);  
 display.WriteCopyPrinterScanBarCodeResult(true, code);  
 return true;  
 }  
 else{  
 display.WriteCopyPrinterScanBarCodeResult(false, code);  
 return false;  
 }  
 }  
 @Override  
 public String GetInfo()  
 {  
 return "This is CopyPrinter class";  
 }  
  
}

import ModulesScanBar.BarCode;  
import ModulesScanBar.Camera;  
import ModulesScanBar.Display;  
  
  
*/\*\*  
 \* Class <code>BarCodeScanner</code> implements bar code scanner  
 \* @author Volodymyr Rud  
 \* @version 2.0  
 \*/*public abstract class BarCodeScanner{  
 protected BarCode scannedBarCode;  
 protected final Display display;  
 protected final Camera camera;  
  
 */\*\*  
 \* Constructor  
 \*/* public BarCodeScanner()  
 {  
 scannedBarCode = new BarCode();  
 display = new Display();  
 camera = new Camera(360);  
 }  
  
 */\*\*  
 \* Constructor  
 \* @param display  
 \* @param camera  
 \*/* public BarCodeScanner(Display display, Camera camera)  
 {  
 this.scannedBarCode = new BarCode();  
 this.display = display;  
 this.camera = camera;  
 }  
  
 */\*\*  
 \* Method that returns scannedBarCode  
 \* @return scannedBarCode  
 \*/* public BarCode getScannedBarCode()  
 {  
 return scannedBarCode;  
 }  
  
 */\*\*  
 \* Method that returns true if process of scanning bar code has executed without exceptions  
 \* @param code  
 \* @return true if process of scanning bar code has executed without exceptions  
 \*/* public abstract boolean ScanBarCode(String code);  
}

package ModulesScanBar;  
  
import java.io.File;  
import java.io.IOException;  
import java.io.PrintWriter;  
import java.util.Arrays;  
  
*/\*\*  
 \* Class <code>ModulesScanBar.Display</code> implements display and AutoCloseable  
 \* @author Volodymyr Rud  
 \* @version 1.0  
 \*/*public class Display implements AutoCloseable {  
 //file to write  
 private PrintWriter fout;  
  
 */\*\*  
 \* Constructor initializes file writter variable  
 \*/* public Display() {  
 try {  
 fout = new PrintWriter(new File("DisplayFile.txt"));  
 } catch (IOException e) {  
 e.printStackTrace();  
 }  
 }  
  
 */\*\*  
 \* Method print in file and in console result of calculating of bar code  
 \*/* public void WriteBarCodeScanResult(boolean res, String code) {  
 if (res) {  
 System.*out*.println("Bar Code " + code + " scanned successfully");  
 fout.println("Bar Code " + code + " scanned successfully");  
 fout.flush();  
 } else {  
 System.*out*.println("Bar Code " + code + " is incorrect");  
 fout.println("Bar Code " + code + " is incorrect");  
 fout.flush();  
 }  
 }  
 public void WriteCopyPrinterScanBarCodeResult(boolean res, String code)  
 {  
 if (res)  
 {  
 System.*out*.println("Bar Code " + code + " printed successfully");  
 fout.println("Bar Code " + code + " printed successfully");  
 fout.flush();  
 }  
 else {  
 System.*out*.println("Bar Code " + code + " didn't print");  
 fout.println("Bar Code " + code + " didn't print");  
 fout.flush();  
 }  
 }  
  
 */\*\*  
 \* Method closes the file  
 \* @throws Exception  
 \*/* @Override  
 public void close() throws Exception {  
 if (fout != null) {  
 fout.flush();  
 fout.close();  
 }  
 }  
  
 */\*\*  
 \* Method prints that bar code is not recognized  
 \*/* public void noCodeRecognise() {  
 System.*out*.println("No bar code is recognized");  
 fout.println("No bar code is recognized");  
 fout.flush();  
 }  
}

package ModulesScanBar;  
  
import java.util.List;  
import java.util.Random;  
  
*/\*\*  
 \* Class <code>ModulesScanBar.Camera</code> implements camera  
 \* @author Volodymyr Rud  
 \* @version 1.0  
 \*/*public class Camera {  
 //resolution of camera  
 private int resolution;  
  
 */\*\*  
 \* Constructor  
 \* @param resolution  
 \*/* public Camera(int resolution) {  
 setResolution(resolution);  
  
 }  
  
 */\*\*  
 \* Method that checks on correctness of parameters to set resolution  
 \* @param resolution  
 \*/* public void setResolution(int resolution) {  
 if (resolution > 0) {  
 this.resolution = resolution;  
 }  
 }  
  
 */\*\*  
 \* Method that returns resolution of camera  
 \* @return resolution of camera  
 \*/* public int getResolution() {  
 return resolution;  
 }  
  
 */\*\*  
 \* Method that returns true 7 in 10 chance  
 \* @return true 7 in 10 chance  
 \*/* public boolean captureImage() {  
 Random random = new Random();  
 int randomNumber = random.nextInt(10);  
  
 return true; //randomNumber < 8;  
 }  
}

package ModulesScanBar;  
  
import java.lang.reflect.Array;  
import java.util.Arrays;  
  
*/\*\*  
 \* Class <code>ModulesScanBar.BarCode</code> implements bar code  
 \* @author Volodymyr Rud  
 \* @version 1.0  
 \*/*public class BarCode {  
 //array of digits of bar code  
 private int[] barCode;  
  
 */\*\*  
 \* Constructor by default. initialize empty array  
 \*/* public BarCode() {  
 barCode = new int[12];  
 }  
  
 */\*\*  
 \*  
 \* @param barCode  
 \* Constuctor  
 \*/* public BarCode(int [] barCode) {  
 setBarCode(barCode);  
 }  
  
 */\*\*  
 \* Method returns array of numbers barCode  
 \* @return array of numbers barCode  
 \*/* public int[] getBarCode() {  
 return barCode;  
 }  
  
 */\*\*  
 \* Method that returns true or false if bar code is calculated correct  
 \* @param code  
 \* @return true or false if bar code is calculated correct  
 \*/* public boolean setBarCode(int[] code) {  
 int sumOdd = 0;  
 int sumEven = 0;  
  
 for (int i = 0; i < 12; i++) {  
 int digit = code[i];  
  
 if ((i + 1) % 2 == 0) {  
 sumEven += digit;  
 } else {  
 sumOdd += digit;  
 }  
 }  
  
 int totalSum = sumOdd + (sumEven \* 3);  
 int calcResult = (10 - (totalSum % 10)) % 10;  
 int checkDigit = code[11];  
  
 if (calcResult == checkDigit) {  
 this.barCode = Arrays.copyOf(code, 12);  
 return true;  
 } else  
 return false;  
 }  
}

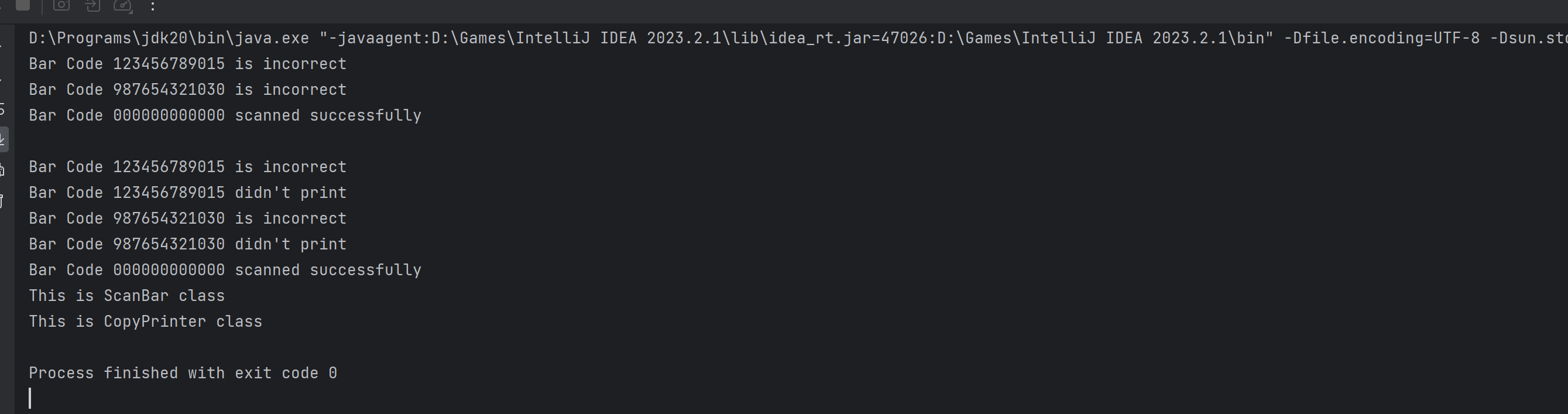
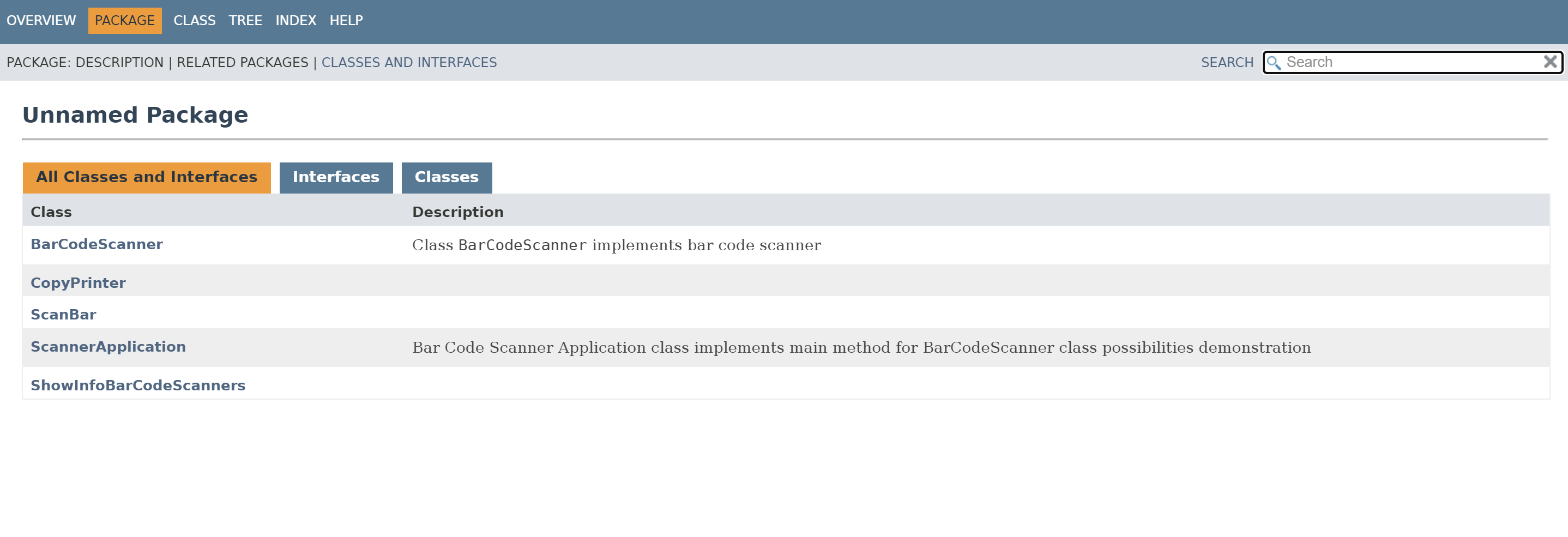
****

Рис.1 – Скріншот документу згенерованого програмою

**Фрагмент згенерованої документації**



**Відповіді на контрольні запитання**

1. Синтаксис реалізації спадкування.

class ChildClass extends ParentClass {

// Код підкласу тут

}

1. Що таке суперклас та підклас?

Суперклас (або батьківський клас) - це клас, від якого успадковуються властивості та методи.

Підклас (або дочірній клас) - це клас, який успадковує властивості та методи від суперкласу і може додавати власні.

1. Як звернутися до членів суперкласу з підкласу?

За допомогою ключового слова super. Наприклад: super.methodName(); для виклику методу суперкласу.

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

Статичне зв'язування відбувається під час компіляції, коли відомо, який метод викликати на основі типу посилання.

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

Динамічне зв'язування відбувається під час виконання програми і визначається на основі об'єкта, який викликає метод.

1. Що таке абстрактний клас та як його реалізувати?

Абстрактний клас - це клас, який не може бути інстанційований і може містити абстрактні методи.

Для створення абстрактного класу використовується ключове слово abstract.

1. Для чого використовується ключове слово instanceof?

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

1. Як перевірити чи клас є підкласом іншого класу?

Використовується ключове слово extends. Наприклад, class ChildClass extends ParentClass.

1. Що таке інтерфейс?

Інтерфейс - це контракт, який описує методи, які клас повинен реалізувати. В інтерфейсах методи не мають реалізації.

1. Як оголосити та застосувати інтерфейс?

Оголошення інтерфейсу: interface InterfaceName { ... }

Застосування інтерфейсу в класі: class ClassName implements InterfaceName { ... }

**Висновок**

Ознайомився з спадкуванням та інтерфейсами у мові Java. Розробив розширений клас з попередньої лабораторної роботи