Міністерство освіти і науки України

Національний університет «Львівська політехніка»



Звіт

до лабораторної роботи №3

з дисципліни: “Кросплатформні засоби програмування”

на тему: «Спадкування та інтерфейси»

Варіант №12

Виконала:

ст.гр. КІ-307

Коваль К. О.

Прийняв:

Іванов Ю. С.

**Львів 2023**

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

**Завдання:**

1. Написати та налагодити програму на мові Java, що розширює клас, що реалізований у лабораторній роботі №2, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №2, зробити абстрактним. Розроблений підклас має забезпечувати механізми свого коректного функціонування та реалізовувати мінімум один інтерфейс. Програма має розміщуватися в пакеті Група.Прізвище.Lab3 та володіти коментарями, які дозволять автоматично згенерувати документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленого пакету.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання:

1. Синтаксис реалізації спадкування.

2. Що таке суперклас та підклас?

3. Як звернутися до членів суперкласу з підкласу?

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

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

6. Що таке абстрактний клас та як його реалізувати?

7. Для чого використовується ключове слово instanceof?

8. Як перевірити чи клас є підкласом іншого класу?

9. Що таке інтерфейс?

10. Як оголосити та застосувати інтерфейс?

**Варіант 12: Море**

**Виконання:**

**Sea.java:**

package KI307.Koval.Lab3;

import java.io.FileNotFoundException;

interface WaterActions {

void dirtyWater();

void clearWater();

}

public class Sea extends WaterBody implements WaterActions {

public Sea(String name, int area, String type) {

super(name, area, type);

// Виклик конструктора з класу WaterBody для ініціалізації полів

}

// Реалізація методів інтерфейсу WaterActions

*@Override*

public void dirtyWater() {

this.isPolluted = true;

System.***out***.println("Water is now polluted.");

}

*@Override*

public void clearWater() {

this.isPolluted = false;

System.***out***.println("Water is now clear.");

}

}

**WaterBody.java:**

package KI307.Koval.Lab3;

import java.io.FileWriter;

import java.io.IOException;

/\*\*

\* Represents a body of water with various attributes and behaviors.

\*/

abstract class WaterBody {

public String name;

public int area;

public String type;

public boolean isPolluted;

public int depth;

/\*\*

\* Constructs a WaterBody with specified name, area, and type.

\* @param name The name of the water body.

\* @param area The area covered by the water body.

\* @param type The type of the water body.

\*/

public WaterBody(String name, int area, String type) {

this.name = name;

this.area = area;

this.type = type;

this.isPolluted = false;

this.depth = 0;

}

/\*\*

\* Constructs a WaterBody with a default name and unknown type.

\* @param name The name of the water body.

\*/

public WaterBody(String name) {

this.name = name;

this.area = 0;

this.type = "Unknown";

this.isPolluted = false;

this.depth = 0;

}

/\*\*

\* Sets the name of the water body.

\* @param name The name to set for the water body.

\*/

public void setName(String name) {

this.name = name;

logActivity("Set name to " + name);

}

/\*\*

\* Sets the area of the water body.

\* @param area The area to set for the water body.

\*/

public void setArea(int area) {

this.area = area;

logActivity("Set area to " + area);

}

/\*\*

\* Sets the type of the water body.

\* @param type The type to set for the water body.

\*/

public void setType(String type) {

this.type = type;

logActivity("Set type to " + type);

}

/\*\*

\* Gets the name of the water body.

\* @return The name of the water body.

\*/

public String getName() {

logActivity("Accessed name: " + name);

return name;

}

/\*\*

\* Gets the area of the water body.

\* @return The area of the water body.

\*/

public int getArea() {

logActivity("Accessed area: " + area);

return area;

}

/\*\*

\* Gets the type of the water body.

\* @return The type of the water body.

\*/

public String getType() {

logActivity("Accessed type: " + type);

return type;

}

/\*\*

\* Increases the area of the water body by a specified amount.

\* @param additionalArea The additional area to increase.

\*/

public void increaseArea(int additionalArea) {

this.area += additionalArea;

logActivity("Increased area by " + additionalArea + " to " + area);

}

/\*\*

\* Changes the type of the water body.

\* @param newType The new type to be set.

\*/

public void changeType(String newType) {

this.type = newType;

logActivity("Changed type to " + newType);

}

/\*\*

\* Prints details about the water body including name, area, type, pollution status, and depth.

\*/

public void printDetails() {

System.out.println("Name: " + name);

System.out.println("Area: " + area);

System.out.println("Type: " + type);

System.out.println("Polluted: " + isPolluted);

System.out.println("Depth: " + depth);

logActivity("Printed details");

}

/\*\*

\* Simulates the effect of rainfall on the water body's area.

\* @param rainfall The amount of rainfall.

\*/

public void rainfallEffect(int rainfall) {

if (rainfall > 50) {

increaseArea(rainfall);

logActivity("Rainfall increased area by " + rainfall);

} else {

logActivity("Rainfall was not sufficient to increase area");

}

}

/\*\*

\* Simulates the pollution of the water body.

\*/

public void pollute() {

this.isPolluted = true;

logActivity("Water body is now polluted");

}

/\*\*

\* Cleans the water body from pollution.

\*/

public void clean() {

this.isPolluted = false;

logActivity("Water body is now clean");

}

/\*\*

\* Sets the depth of the water body.

\* @param depth The depth to set.

\*/

public void setDepth(int depth) {

this.depth = depth;

logActivity("Set depth to " + depth);

}

/\*\*

\* Changes the depth of the water body.

\* @param change The change in depth.

\*/

public void depthChange(int change) {

this.depth += change;

logActivity("Changed depth by " + change + " to " + depth);

}

/\*\*

\* Simulates the process of water evaporation from the water body.

\* @param amount The amount of water to be evaporated.

\*/

public void evaporate(int amount) {

if (amount < depth) {

depth -= amount;

logActivity("Evaporated " + amount + " of water");

} else {

depth = 0;

logActivity("Evaporated all water, depth is now 0");

}

}

/\*\*

\* Simulates the process of water absorption by the water body.

\* @param amount The amount of water to be absorbed.

\*/

public void absorbWater(int amount) {

depth += amount;

logActivity("Absorbed " + amount + " of water");

}

/\*\*

\* Logs an activity to a file.

\* @param activity The activity to be logged.

\*/

private void logActivity(String activity) {

try (FileWriter writer = new FileWriter("log.txt", true)) {

writer.write(activity + "\n");

} catch (IOException e) {

e.printStackTrace();

}

}

/\*\*

\* Closes the log file after finishing with logging activities.

\* This method should be called at the end to ensure proper file closure.

\*/

public void closeLogFile() {

// Method to close the file or handle the file closing process

// This method should be called after finishing working with the log file

}

}

**WaterBodyApp.java:**

package KI307.Koval.Lab3;

import KI307.Koval.Lab3.WaterBody;

/\*\*

\* A sample application to demonstrate the functionalities of the WaterBody class.

\*/

public class WaterBodyApp {

public static void main(String[] args) {

Sea river = new Sea("Nile", 6650, "River");

System.***out***.println("Initial Details:");

river.printDetails();

river.setName("Amazon");

river.setArea(7000);

river.setType("River");

river.depthChange(10);

river.absorbWater(50);

river.evaporate(20);

river.pollute();

System.***out***.println("\nModified Details:");

river.printDetails();

System.***out***.println("\nNew Mods:");

river.dirtyWater();

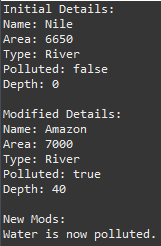
//river.clearWater();

river.closeLogFile();

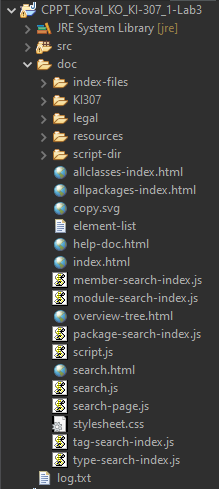
}

}

**Результат:**

****

**Документація:**

****

**Відповіді на контрольні запитання:**

1. \*\*Синтаксис реалізації спадкування в Java:\*\*

```java

class Subclass extends Superclass {

// тіло підкласу

}

```

2. \*\*Суперклас та підклас в Java:\*\*

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

- Підклас (дочірній клас) — клас, який успадковує властивості та методи від суперкласу.

3. \*\*Звернення до членів суперкласу з підкласу в Java:\*\*

- Використання ключового слова `super` перед ім'ям члена суперкласу.

4. \*\*Статичне зв'язування при виклику методу в Java:\*\*

- Статичне зв'язування відбувається під час компіляції.

- Визначається тип змінної, а не тип об'єкта.

5. \*\*Динамічне зв'язування при виклику методу в Java:\*\*

- Динамічне зв'язування відбувається під час виконання програми.

- Визначається тип об'єкта, на який посилається змінна.

6. \*\*Абстрактний клас в Java:\*\*

- Абстрактний клас — це клас, який не може мати екземплярів і може містити абстрактні методи.

- Реалізується за допомогою ключового слова `abstract`.

7. \*\*Використання `instanceof` в Java:\*\*

- `instanceof` перевіряє, чи об'єкт є екземпляром конкретного класу або його підкласу.

- Використовується для безпечного приведення типів.

8. \*\*Перевірка підкласу в Java:\*\*

- Використання ключового слова `instanceof` або використання методу `isAssignableFrom` класу `Class`.

9. \*\*Інтерфейс в Java:\*\*

- Інтерфейс — це колекція абстрактних методів та констант, яку можна реалізувати класами.

- Визначається за допомогою ключового слова `interface`.

10. \*\*Оголошення та застосування інтерфейсу в Java:\*\*

- Оголошення: `interface MyInterface { /\* методи та константи \*/ }`

- Застосування: `class MyClass implements MyInterface { /\* реалізація методів інтерфейсу \*/ }`

**Висновок:** на даній лабораторній роботі я ознайомилась з спадкуванням та інтерфейсами у мові Java.