Міністерство освіти і науки України Національний
«Львівська політехніка»
Кафедра ЕОМ

Звіт
до лабораторної роботи № 3
з дисципліни: «Кросплатформні засоби програмування»
«Спадкування та інтерфейси»Варіант - 1

Виконав:
Студент групи КІ-306
Баран В. М.
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.

Варіант завдання:

1. Спортсмен

Код програми:

File Human.java

```
package org.example;

import java.io.FileNotFoundException;
/**
 *  Class App
 *  @version 1.0
 */
public class Human {
    /**
     * @param args
     */
    public static void main(String[] args) throws FileNotFoundException,
InterruptedException
    {
        Sportsman person = new Sportsman("Roberto", 90, 1.85, "lab4.txt");
person.calculateBodyMassIndex();
        person.calculate(10,3); person.say("Hi!");

        person.run(5);

        person.dispose();
    }
}
```

File Sportsman.java

```java
package org.example;

import java.io.FileNotFoundException;
interface ISportsman{
    void run(int sec) throws InterruptedException;
}


/**
 *  Class Sportsman
 *  @version 1.0
 */
public class Sportsman extends Person implements ISportsman{
    /**
     *  Constructor
     *  @param name person name
     *  @param height person height
     *  @param weight person weight
     *  @param outPutFile output file
     *  @throws FileNotFoundException
     */
    public Sportsman(String name, int weight, double height, String outPutFile)
throws FileNotFoundException {
        super(name, weight, height, outPutFile);
    }

    /**
     *  Constructor
     *  @param sec
     *  @throws InterruptedException
     */
    public void run(int sec) throws InterruptedException
    {
        for (int i = 0; i < sec; i++)
        {
            super.PrintMessage(name + " is running("+i+" sec)");


            Thread.sleep(1000);
        }
    }
}
```

File Person.java

```java
package org.example;

import java.io.File;
import java.io.FileNotFoundException; import java.io.PrintWriter;
/**
 *  Class Person
 *  @version 1.0
 */
public abstract class Person {
    public String name;
    int weight;
    double height;
    int age;

    Mouth mouth;
    Brain brain;
    PrintWriter fout;

    /**
     * Constructor
     *
```

```
     * @param name       person name
     * @param height     person height
     * @param weight     person weight
     * @param outPutFile output file
     * @throws FileNotFoundException
     */
    public Person(String name, int weight, double height, String outPutFile)
throws FileNotFoundException {
        this.name = name;
        this.weight = weight;
        this.age = age;
        this.height = height;
        fout = new PrintWriter(new File(outPutFile));
        mouth = new Mouth(fout, name);
        brain = new Brain(name);
    }

    /**
     * Method to say something
     */
    public void say(String msg) {
        mouth.say(msg);
    }

    /**
     * Method to tell someone
     */
    public void tell(Person person, String msg) {
        PrintMessage(name + " tells " + person.name + ": ");
        mouth.say(msg);

    }

    /**
     * Method for calculation
     */
    public int calculate(int a, int b) {
        int result = brain.sum(a, b);
        PrintMessage(name + " adds two numbers: " + a + "," + b);
        PrintMessage("Result: " + result);
        return result;
    }

    /**
     * Method returns age
     */
    public void getAge() {
        PrintMessage(age);
    }

    /**
     * Method returns height
     */
    public void getHeight() {

        PrintMessage(height);
    }

    /**
     * Method returns weight
     */
    public void getWeight() {
        PrintMessage(this.weight);
    }

    /**
     * Method for calculating BMI
     */
    public void calculateBodyMassIndex() {
```

```
        double idx = weight / Math.pow(height, 2);
        PrintMessage("Body mass index: " + idx);
    }

    /**
     * Method exits
     */
    public void dispose() {
        fout.flush();
        fout.close();
    }

    protected <T> void PrintMessage(T msg) {
        System.out.println(msg);
        fout.println(msg);
    }
}
```

File Brain.java

```
package org.example;

/**
 * @author PC
 *
 */
public class Brain { String name;
    /**
     *  Constructor
     *  @param name person name
     */
    public Brain(String name)
    {
        this.name = name;
    }
    /**
     *  Method returns the sum
     */
    public int sum(int a, int b)
    {
        return a + b;
    }
}
```

File Mouth.java

```
package org.example;

/**
 * @author PC
 *
 */
import java.io.PrintWriter;
public class Mouth { private String name; PrintWriter fout;
    /**
     *  Constructor
     *  @param name person name
     *  @param fout PrintWriter
     */
    public Mouth(PrintWriter fout, String name)

    {
        this.name = name;
        this.fout = fout;
```

```
    }
    /**
     *  Method to say something
     */
    public void say(String message)
    {
        String output = name + " says: " + message; System.out.println(output);
fout.println(output);
    }
}
```

Результат виконання програми:

```
Body mass index: 26.296566837107374
Roberto adds two numbers: 10,3
Result: 13
Roberto says: Hi!
Roberto is running(0 sec)
Roberto is running(1 sec)
Roberto is running(2 sec)
Roberto is running(3 sec)
Roberto is running(4 sec)
```

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package org.example

Class Person

java.lang.Object
org.example.Person

Direct Known Subclasses:

Sportsman

public abstract class **Person**
extends Object

Class Person

Field Summary

Fields

Modifier and Type	Field	Description
String	name	

Constructor Summary

Constructors

Constructor	Description
Person(String name, int weight, double height, String outPutFile)	Constructor

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description

Відповіді на контрольні запитання

1. Синтаксис реалізації спадкування.

- class МійКлас implements Інтерфейс {

// тіло класу }

2. 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.

3. Як звернутися до членів суперкласу з підкласу?

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

super.назваПоля; // звернення до поля суперкласу

4. 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.

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

- вибір методу для виклику відбувається під час виконання програми на основі

фактичного типу об’єкта.

6. 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(); }

7. 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 }

8. 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.

9. 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.

10. 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 отримала навички роботи з концепціями спадкування та інтерфейсами в мові програмування Java. Ознайомившись з цими важливими аспектами об'єктно-орієнтованого програмування, я зрозуміла їх роль у створенні більш структурованих і гнучких програм.