Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»

Кафедра ЕОМ

Звіт

Лабораторна робота № 3

З дисципліни «Кросплатформні засоби програмування»

На тему: «Спадкування та інтерфейси»

Виконала: ст. гр. КІ-303

Кілик І.Р.

Перевірив:

доцент кафедри ЕОМ

Іванов Ю.С.

Львів 2025

Мета: ознайомитися з спадкуванням та інтерфейсами у мові Java.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Спадкування

Спадкування в ООП призначене для розширення функціональності існуючих класів шляхом утворення нових класів на базі вже існуючих. У Java реалізована однокоренева архітектура класів згідно якої всі класи мають єдиного спільного предка (кореневий клас в ієрархії класів) – клас Object. Решта класів мови Java утворюються шляхом успадковування даного класу. Будь-яке спадкування у мові Java є відкритим, при цьому аналогів захищеному і приватному спадкуванню мови C++ не існує. На відміну від C++ у Java можливе спадкування лише одного базового класу (множинне спадкування відсутнє). Спадкування реалізується шляхом вказування ключового слова class після якого вказується назва підкласу, ключове слово extends та назва суперкласу, що розширюється у новому підкласі. Синтаксис реалізації спадкування:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

В термінах мови Java базовий клас найчастіше називається суперкласом, а похідний клас – підкласом. Дана термінологія запозичена з теорії множин, де підмножина міститься у супермножині.

При наслідувані 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 визначені 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.

Перевизначення у підкласах елементів суперкласів (полів або методів) призводить до їх приховування новими елементами. Бувають ситуації, коли у методах підкласу необхідно звернутися до цих прихованих елементів

суперкласів. У цій ситуації слід використати ключове слово super, яке вказує, що елемент до якого йде звернення,

розташовується у суперкласі, а не у підкласі. Синтаксис звертання до елементів

суперкласу:

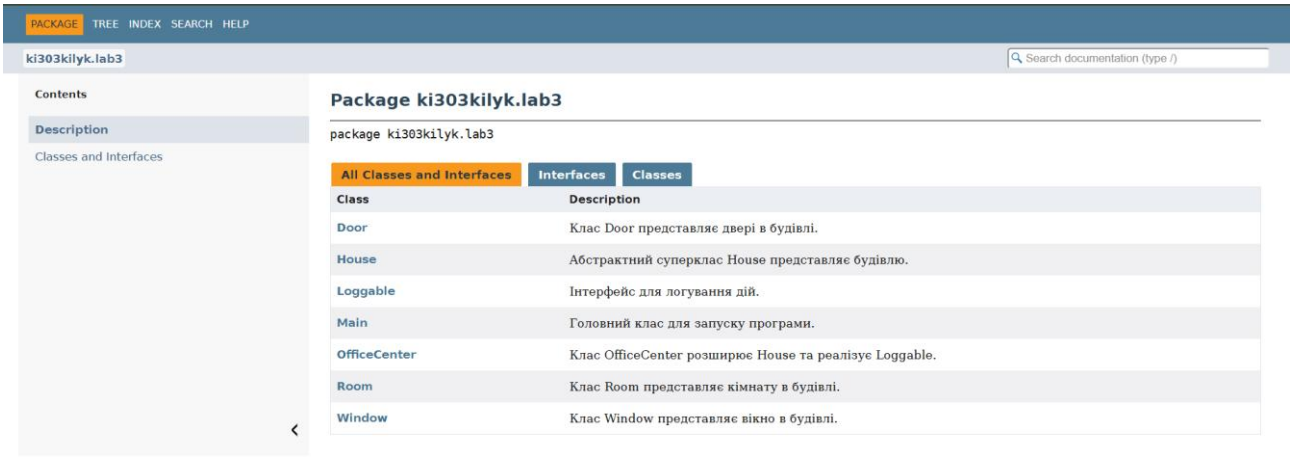
super.назваМетоду([параметри]); // виклик методу суперкласу

super.назваПоля // звертання до поля суперкласу

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 функціонуваннята 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 – 10 офісний центр



Висновок: Завдяки виконанню даної лабораторної роботи, я ознайомилась з спадкуванням та інтерфейсами у мові Java

Додатки

```java
package ki303kilyk.lab3;

/**
 * Клас Door представляє двері в будівлі.
 */
public class Door {
    private final String type;
    private boolean isOpen;

    public Door(String type) {
        this.type = type;
        this.isOpen = false;
    }

    public void open() {
        isOpen = true;
    }

    public void close() {
        isOpen = false;
    }

    public boolean isOpen() {
        return isOpen;
    }

    public String getType() {
        return type;
    }
}
```

```java
package ki303kilyk.lab3;

/**
 * Клас Window представляє вікно в будівлі.
 */
public class Window {
    private final String type;
```

```
    private boolean isOpen;

    public Window(String type) {
        this.type = type;
        this.isOpen = false;
    }

    public void open() {
        isOpen = true;
    }

    public void close() {
        isOpen = false;
    }

    public boolean isOpen() {
        return isOpen;
    }

    public String getType() {
        return type;
    }
}
```

```
package ki303kilyk.lab3;

/**
 * Клас Room представляє кімнату в будівлі.
 */
public class Room {
    private final String name;
    private final double area;

    public Room(String name, double area) {
        this.name = name;
        this.area = area;
    }

    public String getName() {
        return name;
    }

    public double getArea() {
        return area;
    }
}
```

```java
package ki303kilyk.lab3;

/**
 * Абстрактний суперклас House представляє будівлю.
 */
public abstract class House {
    protected Room[] rooms;
    protected Door[] doors;
    protected Window[] windows;

    public House(Room[] rooms, Door[] doors, Window[] windows) {
        this.rooms = rooms;
        this.doors = doors;
        this.windows = windows;
    }

    public abstract void showInfo();
}
```

```java
package ki303kilyk.lab3;

/**
 * Клас OfficeCenter розширює House та реалізує Loggable.
 */
public class OfficeCenter extends House implements Loggable {
    public OfficeCenter(Room[] rooms, Door[] doors, Window[] windows) {
        super(rooms, doors, windows);
    }

    @Override
    public void showInfo() {
        System.out.println("Office Center Info:");
        for (Room room : rooms) {
            System.out.println("Room: " + room.getName() + ", Area: " +
room.getArea());
        }
    }

    @Override
    public void log(String message) {
        System.out.println("Log: " + message);
    }
}
```

```java
package ki303kilyk.lab3;

/**
```

```java
 * Інтерфейс для логування дій.
 */
public interface Loggable {
    void log(String message);
}
```

```java
package ki303kilyk.lab3;



/**
 * Головний клас для запуску програми.
 */
public class Main {
    public static void main(String[] args) {
        Room[] rooms = { new Room("Office1", 25.0), new Room("Office2", 30.0) };
        Door[] doors = { new Door("Wood"), new Door("Metal") };
        Window[] windows = { new Window("Glass"), new Window("Plastic") };

        OfficeCenter officeCenter = new OfficeCenter(rooms, doors, windows);
        officeCenter.showInfo();
        officeCenter.log("Office center created successfully");
    }
}
```