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

Лабораторна робота №4

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Спадкування та інтерфейси»

Варіант № 10

Виконав: ст. гр. КІ-34 Лендел Т.В.
Прийняв: Іванов Ю.С

Львів – 2022

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 №3, для реалізації предметної області заданої варіантом. Суперклас, що реалізований у лабораторній роботі №3, 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.

4. Дати відповідь на контрольні запитання.

Код Company.java:

```
package main.kzp.lab4;

import java.io.Serializable;

public class Company implements Serializable {
    private String name;

    public Company() {
    }

    public Company(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    @Override
    public String toString() {
        return "Company: " + name + '\n';
    }
}
```

Код House.java:

```
package main.kzp.lab4;

import main.kzp.lab3.Human;
```

```
import java.io.Externalizable;
import java.io.IOException;
import java.io.ObjectInput;
import java.io.ObjectOutput;

public abstract class House implements Externalizable {
    private String address;
    private Human owner;
    private double squareMeters;

    public String getAddress() {
        return address;
    }

    public Human getOwner() {
        return owner;
    }

    public double getSquareMeters() {
        return squareMeters;
    }

    public void setAddress(String address) {
        this.address = address;
    }

    public void setOwner(Human owner) {
        this.owner = owner;
    }

    public void setSquareMeters(double squareMeters) {
        this.squareMeters = squareMeters;
    }

    @Override
    public void writeExternal(ObjectOutput out) throws IOException {
        out.writeObject(address);
        out.writeDouble(squareMeters);
    }

    @Override
    public void readExternal(ObjectInput in) throws IOException, ClassNot-
FoundException {
        address = (String)in.readObject();
        squareMeters = in.readDouble();
    }

    @Override
    public String toString() {
        return "\nHouse:\n" +
                "\taddress: " + address + '\n' +
                "\towner: " + owner + '\n' +
                "\tsquareMeters: " + squareMeters + "\n";
    }
}
```

Код OfficeCenter.java:

```
package main.kzp.lab4;
```

```java
import java.io.*;
import java.util.ArrayList;
import java.util.List;

public class OfficeCenter extends House implements Externalizable {
    private List<OfficeRoom> officeRooms = new ArrayList<>();

    public void createNewRoom(double pricePerMonth, Company companyRenter){
        officeRooms.add(new OfficeRoom(pricePerMonth, companyRenter));
    }

    public void printAll(){
        System.out.println("#".repeat(30));
        printOfficeCenterData();
        printAllOfficeRooms();
        System.out.println("#".repeat(30));
    }

    public void printOfficeCenterData(){
        System.out.println(super.toString());
    }

    public void printAllOfficeRooms(){
        for(int i = 0; i < officeRooms.size(); ++i)
            System.out.println(String.format("Room #%d\n%s\n", i+1, of-
ficeRooms.get(i)));
    }

    @Override
    public void writeExternal(ObjectOutput out) throws IOException {
        super.writeExternal(out);
        out.writeObject(officeRooms);
    }

    @Override
    public void readExternal(ObjectInput in) throws IOException, ClassNot-
FoundException {
        super.readExternal(in);
        officeRooms = (List<OfficeRoom>)in.readObject();
    }

    private class OfficeRoom implements Serializable {
        private double pricePerMonth;
        private Company companyRenter;

        public OfficeRoom() {
        }

        public OfficeRoom(double pricePerMonth) {
            this.pricePerMonth = pricePerMonth;
        }

        public OfficeRoom(double pricePerMonth, Company companyRenter) {
            this.pricePerMonth = pricePerMonth;
            this.companyRenter = companyRenter;
        }

        public double getPricePerMonth() {
            return pricePerMonth;
```

```
        }

        public Company getCompanyRenter() {
            return companyRenter;
        }

        public void setPricePerMonth(double pricePerMonth) {
            this.pricePerMonth = pricePerMonth;
        }

        public void setCompanyRenter(Company companyRenter) {
            this.companyRenter = companyRenter;
        }

        @Override
        public String toString() {
            return companyRenter.toString() + "Price per month: " + pricePer-
Month;
        }
    }
}
```

Код Main.java:

```
package main.kzp.lab4;

import main.kzp.lab3.Human;

import java.io.*;

public class Main {
    public static void main(String[] args) {
        Human human = new Human();
        human.setFirstName("Geogiy");
        human.setLastName("Sidorov");

        Company company1 = new Company("Apple");
        Company company2 = new Company("BMW");
        Company company3 = new Company("Pepsico");

        OfficeCenter officeCenter = new OfficeCenter();
        officeCenter.setOwner(human);

        officeCenter.createNewRoom(21000, company1);
        officeCenter.createNewRoom(28800, company3);
        officeCenter.createNewRoom(12990, company2);
        officeCenter.createNewRoom(29000, company1);
        officeCenter.createNewRoom(20100, company2);
        officeCenter.createNewRoom(20100, company2);
        officeCenter.createNewRoom(21000, company3);

        officeCenter.printAll();
        serialize(officeCenter);
        officeCenter = deserialize();
        officeCenter.printAll();
    }

    private static void serialize(OfficeCenter officeCenter){
        try(FileOutputStream fos = new FileOutputStream("objectSer.ser");
            ObjectOutputStream oos = new ObjectOutputStream(fos)) {
```

```
            oos.writeObject(officeCenter);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }

    private static OfficeCenter deserialize(){
        try(FileInputStream fis = new FileInputStream("objectSer.ser");
            ObjectInputStream ois = new ObjectInputStream(fis)) {
            return (OfficeCenter)ois.readObject();
        } catch (IOException | ClassNotFoundException e) {
            throw new RuntimeException(e);
        }
    }
}
```

Висновок: Я ознайомився з спадкуванням та інтерфейсами у мові Java.

Контрольні питання:

1. Синтаксис реалізації спадкування.

Відповідь:

class Підклас extends Суперклас

{

Додаткові поля і методи

}

2. Що таке суперклас та підклас?

Відповідь: Суперклас – батьківський клас. Підклас – дочірній.

3. Як звернутися до членів суперкласу з підкласу?

Відповідь: super.назваМетоду([параметри]); super.назваПоля;

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

Відповідь: метод є приватним, статичним, фінальним або конструктором. Механізм статичного зв’язування передбачає визначення методу, який необхідно викликати, на етапі компіляції.

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

Відповідь: метод, що необхідно викликати, визначається по фактичному типу неявного параметру.

6. Що таке абстрактний клас та як його реалізувати?

Відповідь: Це клас який оголошений з ключовим словом abstract. Об’єкт такого класу не може бути створеним, може вміщати абстрактні методи.

7. Для чого використовується ключове слово instanceof?

Відповідь: Для встановлення чи є певний клас спадкоємцем другого.

8. Як перевірити чи клас є підкласом іншого класу?

Відповідь: використати ключове слово instanceof.

9. Що таке інтерфейс?

Відповідь: Інтерфейси вказують що повинен робити клас не вказуючи як саме він це повинен робити. Інтерфейси покликані компенсувати відсутність множинного спадкування у мові Java та гарантують визначення у класах оголошених у собі прототипів методів.

10. Як оголосити та застосувати інтерфейс?

Відповідь: [public] interface НазваІнтерфейсу

{

Прототипи методів та оголошення констант інтерфейсу

}

Застосувати можна імплементуючи його, або створюючи посилання на дочірній об’єкт класу.