Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/07 Интеллектуальные системы анализа, обработки и интерпретации больших данных**

ОТЧЕТ

по лабораторной работе № 4

Название: Внутренний класс. Интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М _____ Д. Р. Григорян

Преподаватель _____ П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2023

Цель работы:

Ознакомиться с языком программирования Java и научиться работать с внутренними классами, интерфейсами и наследованием.

Выполнение:

Задача 1.1:

Создать класс Справочная Служба Общественного Транспорта с внутренним классом, с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.

Листинг 1 класса PublicTransportDirectory:

```
import java.util.ArrayList;
class PublicTransportDirectory {
    // Main class for storing directory of public transport
    ArrayList<TransportInfo> directory;
    TransportInfo transportInfo;

    // Constructor for creating new directory
    public PublicTransportDirectory() {
        directory = new ArrayList<>();
    }

    // Add new transport information to directory
    public void addTransportInfo(String time, String route, double price) {
        transportInfo = new TransportInfo(time, route, price);
        directory.add(transportInfo);
    }

    // Get transport information by index
    public TransportInfo getTransportInfo(int index) {
        return directory.get(index);
    }

    // Get total number of transport information in directory
    public int getSize() {
        return directory.size();
    }
    // Inner class for storing information about time, routes and price
    public class TransportInfo {
        private String time,route;
        private double price;

        public TransportInfo(String time, String route, double price) {
            this.time = time;
            this.route = route;
            this.price = price;
        }

        public String getTime() {
            return time;
        }

        public String getRoute() {
            return route;
```

```
        }

        public double getprice() {
            return price;
        }
    }
}
```

Листинг 2 программы main:

```
//7.Создать класс Справочная Служба Общественного Транспорта с внутренним
классом,
//        с помощью объектов которого можно хранить информацию о времени,
//        линиях маршрутов и стоимости проезда
import java.util.*;
public class Main {
    public static void main(String[] args) {
        PublicTransportDirectory system = new PublicTransportDirectory();
        system.addTransportInfo("150 min","From Moscow to Ramenskoe",100);
        system.addTransportInfo("240 min","From Jukovskiy to Ramenskoe",30);
        System.out.println(system.directory.get(1).getTime());
        System.out.println(system.directory.get(0).getRoute());
    }

}
```

Результат приведен на рисунке 1:

```
C:\Users\sonar\.jdks\openjdk-19.0.2\bin\java
240 min
From Moscow to Ramenskoe

Process finished with exit code 0
```

Рисунок 1 – Результат выполнения программы

Задача 1.2:

Создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.

Листинг 3 класса ComputerDirectory:

```
import java.util.ArrayList;

class ComputerDirectory {
    // Main class for storing directory of public transport
    ArrayList<ComputerInfo> directory;
    ComputerInfo ComputerInfo;

    // Constructor for creating new directory
```

```
    public ComputerDirectory() {
        directory = new ArrayList<>();
    }

    // Add new transport information to directory
    public void addComputerInfo(String OS, String processor, double RAM) {
        ComputerInfo = new ComputerInfo(OS, processor, RAM);
        directory.add(ComputerInfo);
    }

    // Get transport information by index
    public ComputerInfo getComputerInfo(int index) {
        return directory.get(index);
    }

    // Get total number of transport information in directory
    public int getSize() {
        return directory.size();
    }
    // Inner class for storing information about time, routes and price
    public class ComputerInfo {
        private String OS,processor;
        private double RAM;


        public ComputerInfo(String OS, String processor, double RAM) {
            this.OS = OS;
            this.processor = processor;
            this.RAM = RAM;
        }

        public String getOS() {
            return OS;
        }

        public double getRAM() {
            return RAM;
        }

        public String getProcessor() {
            return processor;
        }
    }
}
```

Листинг 4 программы main:

```
//8.    Создать класс Computer (компьютер) с внутренним классом,
//        с помощью объектов которого можно хранить информацию об
операционной системе, процессоре и оперативной памяти.
import java.util.*;
public class Main {
    public static void main(String[] args) {
        ComputerDirectory system = new ComputerDirectory();
        system.addComputerInfo("Windows","Intel Core I5",8);
        system.addComputerInfo("Linux","AMD",4);
        System.out.println(system.directory.get(1).getOS());
        System.out.println(system.directory.get(0).getRAM());
    }

}
```

Результат приведен на рисунке 2:

```
C:\Users\sonar\.jdks\openjdk-19.0.2\bin\j
Linux
8.0

Process finished with exit code 0
```

Рисунок 2 – Результат выполнения программы

Задача 2.1:

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов.

interface Врач <- class Хирург <- class Нейрохирург

Листинг 5 класса Doctor:

```
abstract class Doctor {
    private String name;
    private int age;

    public Doctor(String name, int age) {
        this.name = name;
        this.age = age;
    }

    public void work() {
        System.out.println(toString() + "Я просто врач");
    }

    @Override
    public String toString() {
        return "Я " + name + ", мне " + age + " лет. " ;
    }
}
```

Листинг 6 класса Surgeon:

```
class SurgenDoctor extends Doctor {
    public SurgenDoctor(String name, int age) {
        super(name, age);
    }
    // переопределение метода базового класса
    @Override
    public void work() {
        System.out.println( toString() + "Я работаю хирургом");
    }
}
```

Листинг 7 класса NeuroSurgeon:

```
class NeuroSurgeon extends Doctor{

    public NeuroSurgeon(String name, int age) {
        super(name, age);
    }
    // переопределение метода базового класса
    @Override
    public void work(){
        System.out.println(toString() + "Я работаю нейрохирургом");
    }
}
```

Листинг 8 программы main:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        //Doctor doctor = new Doctor("Антон", 18);

        Doctor Surgeon = new SurgenDoctor("Алексей", 19);// восходящее
преобразование к базовому типу
        Doctor NeuroSurgeon = new NeuroSurgeon("Игорь", 20); // восходящее
преобразование к базовому типу

        List<Doctor> hospital = Arrays.asList(Surgeon, NeuroSurgeon);
        for (Doctor d : hospital) {
            d.work();// полиморфный вызов метода
        }
    }
}
```

Результат приведен на рисунке 3:

```
C:\Users\sonar\.jdks\openjdk-19.0.2\bin\java.e
Я Алексей, мне 19 лет. Я работаю хирургом
Я Игорь, мне 20 лет. Я работаю нейрохирургом

Process finished with exit code 0
```

Рисунок 3 – Результат выполнения программы

Задача 2.2:

Реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов.

interface Корабль <- class Грузовой Корабль <- class Танкер.

Листинг 9 класса Tanker:

```
class Tanker extends Ship{

    public Tanker(String name, int width, int length, int SideHeight) {
        super(name, width,length,SideHeight);
```

```
    }
    // переопределение метода базового класса
    @Override
    public void work(){
        System.out.println(toString() + " Это танкер");
    }
}
```

Листинг 10 класса Ship:

```
abstract class Ship {
    private String name;

    private int width,length,SideHeight;

    public Ship(String name, int width, int length, int sideHeight) {
        this.name = name;
        this.width = width;
        this.length = length;
        SideHeight = sideHeight;
    }
    public void work() {
        System.out.println(toString() + "Это корабль");
    }

    @Override
    public String toString() {
        return "Этот корабль имеет название - " + name + ". Его Длина:" +
length + " Ширина: " + width+ " Высота борта: " + SideHeight ;
    }
}
```

Листинг 11 класса CargoShip:

```
class CargoShip extends Ship {
    public CargoShip(String name, int width, int length, int SideHeight) {
        super(name, width,length,SideHeight);
    }
    // переопределение метода базового класса
    @Override
    public void work() {
        System.out.println( toString() + " Это грузовой корабль");
    }
}
```

Листинг 12 программы main:

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        //Ship ship = new Ship("Антон", 18);

        Ship Cargo = new CargoShip("Беда", 190,500,250);// восходящее
преобразование к базовому типу
        Ship TankShip = new Tanker("Путь", 560,890,450); // восходящее
преобразование к базовому типу

        List<Ship> sea = Arrays.asList(Cargo, TankShip);
        for (Ship d : sea) {
            d.work();// полиморфный вызов метода
```

```
            }
        }
}
```

Результат приведен на рисунке 4:

```
C:\Users\sonar\.jdks\openjdk-19.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Commun
Этот корабль имеет название - Беда. Его Длина:500 Ширина: 190 Высота борта: 250 Это грузовой корабль
Этот корабль имеет название - Путь. Его Длина:890 Ширина: 560 Высота борта: 450 Это танкер

Process finished with exit code 0
```

Рисунок 4 – Результат выполнения программы

Вывод: в ходе выполнения лабораторной работы были написаны программы согласно выданному заданию. Реализованы абстрактные классы/интерфейсы, наследование, полиморфизм для заданных классов.