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**

О Т Ч Е Т

по лабораторной работе № 4

Название: Внутренние классы, интерфейсы

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М _____ Т.И. Кадыров
(Группа) (Подпись, дата) (И.О. Фамилия)

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

Москва, 2024

Цель работы: освоить базовые принципы работы с внутренними классами и интерфейсами на языке Java.

Вариант: 8.

Задание 1: Создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.

Код решения приведен в листинге 1.

Листинг 1 — реализация решения

```
public class Computer {
    public class Specs {
        private String os;
        private String processor;
        private int ram;

        public Specs(String os, String processor, int ram) {
            this.os = os;
            this.processor = processor;
            this.ram = ram;
        }

        public String getOS() {
            return os;
        }

        public void setOS(String os) {
            this.os = os;
        }

        public String getProcessor() {
            return processor;
        }

        public void setProcessor(String processor) {
            this.processor = processor;
        }

        public int getRAM() {
            return ram;
        }

        public void setRAM(int ram) {
            this.ram = ram;
        }

        @Override
        public String toString() {
            return "OS: " + os + ", Processor: " + processor + ", RAM: " + ram + "GB";
        }
```

```
    }
    private Specs specs;
    public Computer(String os, String processor, int ram) {
        this.specs = new Specs(os, processor, ram);
    }
    public Specs getSpecs() {
        return specs;
    }
}

public class Main {
    public static void main(String[] args) {

        Computer computer = new Computer("Windows", "Intel i5", 16);
        System.out.println(computer.getSpecs());
    }
}
```

Задание 2: Создать класс Park (парк) с внутренним классом, с помощью объектов которого можно хранить информацию об аттракционах, времени их работы и стоимости.

Код решения приведен в листинге 2.

Листинг 2 — реализация решения

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

public class Park {
    public class Attraction {
        private String name;
        private String workingHours;
        private double cost;

        public Attraction(String name, String workingHours, double cost) {
            this.name = name;
            this.workingHours = workingHours;
            this.cost = cost;
        }

        public String getName() {
            return name;
        }

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

        public String getWorkingHours() {
            return workingHours;
        }
```

```
        public void setWorkingHours(String workingHours) {
            this.workingHours = workingHours;
        }

        public double getCost() {
            return cost;
        }

        public void setCost(double cost) {
            this.cost = cost;
        }

        @Override
        public String toString() {
            return "Attraction: " + name + ", Working Hours: " + workingHours + ", Cost: $" + cost;
        }
    }

    private List<Attraction> attractions;

    public Park() {
        attractions = new ArrayList<>();
    }

    public void addAttraction(String name, String workingHours, double cost) {
        Attraction attraction = new Attraction(name, workingHours, cost);
        attractions.add(attraction);
    }

    public List<Attraction> getAttractions() {
        return attractions;
    }
}

public class Main {
    public static void main(String[] args) {
        Park park = new Park();
        park.addAttraction("Roller Coaster", "10:00 AM - 8:00 PM", 25.0);
        park.addAttraction("Ferris Wheel", "11:00 AM - 9:00 PM", 15.0);

        System.out.println("Park Attractions Information:");
        for (Park.Attraction attraction : park.getAttractions()) {
            System.out.println(attraction);
        }
    }
}
```

Задание 3: interface Корабль <- class Грузовой Корабль <- class Танкер.

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

```
public interface Ship {
```

```
    void sail();
}

public class CargoShip implements Ship {
    private int load;
    @Override
    public void sail() {
        System.out.println("CargoShip is sailing");
    }

    public void loadCargo() {
        System.out.println("CargoShip is loading");
    }

    public CargoShip(int load) {
        this.load = load;
    }
}

public class Tanker extends CargoShip {

    @Override
    public void sail() {
        System.out.println("Tanker is sailing");
    }

    @Override
    public void loadCargo() {
        System.out.println("Tanker is loading cargo");
    }

    public Tanker(int load) {
        super(load);
    }
}

public class Main {
    public static void main(String[] args) {

        Tanker tanker = new Tanker(150);
        tanker.sail();
        tanker.loadCargo();
    }
}
```

Задание 4: interface Мебель <- abstract class Шкаф <- class Книжный Шкаф.

Код решения приведен в листинге 4.

Листинг 4 — реализация решения

```
interface Furniture {
```

```
    void display();
}

public abstract class CupBoard implements Furniture {
    int capacity;
    @Override
    public void display() {
        System.out.printf("Capacity of cupboard is: %d\n", capacity);
    }

    abstract void open();

    public CupBoard(int capacity) {
        this.capacity = capacity;
    }
}

public class BookshelfCupboard extends CupBoard {
    @Override
    public void open() {
        System.out.println("Bookshelf cupboard is being opened.");
    }

    public BookshelfCupboard(int capacity) {
        super(capacity);
    }
}

public class Main {
    public static void main(String[] args) {

        BookshelfCupboard bookshelfCupboard = new BookshelfCupboard(10);
        bookshelfCupboard.display();
        bookshelfCupboard.open();
    }
}
```

Вывод: в ходе лабораторной работы были освоены базовые принципы работы с внутренними классами и интерфейсами на языке Java.