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, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ 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: Лабораторная работа №4

Дисциплина: Языки программирования для работы с большими данными

Студент	ИУ6-22М		М.И. Замула
	(Группа)	(Подпись, дата)	(И.О. Фамилия)
Преподаватель			П.В. Степанов
		(Подпись, дата)	(И.О. Фамилия)

Москва, 2023 г.

Вариант: 1

Задание 8

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

Выполнение

Класс Computer

```
package lab4;

public class Computer {
    private String name;
    private Specification specification;

    public Computer(String name, Specification specification) {
        this.name = name;
        this.specification = specification;
    }

    public String getName() {
        return name;
    }

    public Specification getSpecification() {
        return specification;
    }

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

    public void setSpecification(Specification specification) {
        this.specification = specification;
    }

    public static class Specification {
        private String operatingSystem;
        private String processor;
        private int memory;

        public Specification(String operatingSystem, String processor, int
memory) {
            this.operatingSystem = operatingSystem;
            this.processor = processor;
            this.memory = memory;
        }

        public String getOperatingSystem() {
            return operatingSystem;
        }

        public String getProcessor() {
            return processor;
        }

        public int getMemory() {
```

```
            return memory;
        }

        public void setOperatingSystem(String operatingSystem) {
            this.operatingSystem = operatingSystem;
        }

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

        public void setMemory(int memory) {
            this.memory = memory;
        }
    }
}
```

Main

```
package lab4;


public class Lab4_1_8 {
    public static void main(String[] args) {
        // Создание объекта компьютера с именем "Мой компьютер"
        Computer myComputer = new Computer("Мой компьютер", new
Computer.Specification("Windows 10", "Intel Core i5", 8));

// Получение имени компьютера
        System.out.println("Имя компьютера: " + myComputer.getName());

// Получение информации о спецификации компьютера
        Computer.Specification spec = myComputer.getSpecification();
        System.out.println("Операционная система: " +
spec.getOperatingSystem());
        System.out.println("Процессор: " + spec.getProcessor());
        System.out.println("Оперативная память: " + spec.getMemory() + "
ГБ");

// Изменение информации о спецификации компьютера
        spec.setOperatingSystem("Windows 11");
        spec.setMemory(16);
        myComputer.setSpecification(spec);

// Вывод обновленной информации о спецификации компьютера
        System.out.println("Операционная система: " +
myComputer.getSpecification().getOperatingSystem());
    }
}
```

Результаты

```
Имя компьютера: Мой компьютер
Операционная система: Windows 10
Процессор: Intel Core i5
Оперативная память: 8 ГБ
Операционная система: Windows 11
```

Задание 9

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

Выполнение

Класс Park

```
package lab4;

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

public class Park {
    private final 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 record Attraction(String name, String workingHours, double cost) {
    }
}
```

Main

```
package lab4;

import java.util.List;
```

```java
public class Lab4_1_9 {

    public static void main(String[] args) {
        Park park = new Park();
        park.addAttraction("Rollercoaster", "10:00 - 18:00", 499.99);
        park.addAttraction("Ferris Wheel", "11:00 - 20:00", 449.99);
        park.addAttraction("Bumper Cars", "12:00 - 22:00", 599.99);


        List<Park.Attraction> attractions = park.getAttractions();
        for (Park.Attraction attraction : attractions) {
            System.out.println("Attraction: " + attraction.name());
            System.out.println("Working Hours: " +
attraction.workingHours());
            System.out.println("Cost: " + attraction.cost());
            System.out.println();
        }
    }

}
```

Результаты

```
Attraction: Rollercoaster
Working Hours: 10:00 - 18:00
Cost: 499.99

Attraction: Ferris Wheel
Working Hours: 11:00 - 20:00
Cost: 449.99

Attraction: Bumper Cars
Working Hours: 12:00 - 22:00
Cost: 599.99
```

Вариант: 2

Задание 8

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

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

Выполнение

```
package lab4;

public class Lab4_2_8 {
    public static void main(String[] args) {
        CargoShip cargoShip = new Tanker("Tanker01", 10000, 5000);
        cargoShip.move();
        cargoShip.stop();
        System.out.println("Cargo capacity: " +
cargoShip.getCargoCapacity());
        cargoShip.loadCargo();
        System.out.println("Oil capacity: " + ((Tanker)
cargoShip).getOilCapacity());
    }
}

interface Ship {
    void move();

    void stop();
}

abstract class CargoShip implements Ship {
    private final String name;
    private final int cargoCapacity;

    public CargoShip(String name, int cargoCapacity) {
        this.name = name;
        this.cargoCapacity = cargoCapacity;
    }

    public int getCargoCapacity() {
        return cargoCapacity;
    }

    public void loadCargo() {
        System.out.println("Loading cargo...");
    }

    public void move() {
        System.out.println(name + " is moving.");
    }

    public void stop() {
        System.out.println(name + " has stopped.");
```

```
    }
}

class Tanker extends CargoShip {
    private final int oilCapacity;

    public Tanker(String name, int cargoCapacity, int oilCapacity) {
        super(name, cargoCapacity);
        this.oilCapacity = oilCapacity;
    }

    public int getOilCapacity() {
        return oilCapacity;
    }
}
```

Результат

```
Tanker01 is moving.
Tanker01 has stopped.
Cargo capacity: 10000
Loading cargo...
Oil capacity: 5000
```

Задание 8

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

interface Мебель <- abstract class Шкаф <- class Книжный Шкаф

Выполнение

```
package lab4;

public class Lab4_2_9 {
    public static void main(String[] args) {
        Bookcase bookcase = new Bookcase(5, true);
        System.out.println("Number of shelves: " +
bookcase.getNumberOfShelves());
        System.out.println("Has glass doors: " +
bookcase.getHasGlassDoors());
        bookcase.move();
        bookcase.openDoor();
```

```
        bookcase.closeDoor();
    }
}


interface Furniture {
    void move();
}
abstract class Cabinet implements Furniture {
    private int numberOfShelves;

    public Cabinet(int numberOfShelves) {
        this.numberOfShelves = numberOfShelves;
    }

    public int getNumberOfShelves() {
        return numberOfShelves;
    }

    public void setNumberOfShelves(int numberOfShelves) {
        this.numberOfShelves = numberOfShelves;
    }

    public void openDoor() {
        System.out.println("Opening the cabinet door.");
    }

    public void closeDoor() {
        System.out.println("Closing the cabinet door.");
    }
}

class Bookcase extends Cabinet {
    private boolean hasGlassDoors;

    public Bookcase(int numberOfShelves, boolean hasGlassDoors) {
        super(numberOfShelves);
        this.hasGlassDoors = hasGlassDoors;
    }

    public boolean getHasGlassDoors() {
        return hasGlassDoors;
    }

    public void setHasGlassDoors(boolean hasGlassDoors) {
        this.hasGlassDoors = hasGlassDoors;
    }

    @Override
    public void move() {
        System.out.println("Moving the bookcase.");
    }

    @Override
    public void openDoor() {
        if (hasGlassDoors) {
            System.out.println("Opening the glass doors of the bookcase.");
        } else {
            System.out.println("Opening the wooden doors of the bookcase.");
        }
    }

    @Override
```

```
    public void closeDoor() {
        if (hasGlassDoors) {
            System.out.println("Closing the glass doors of the bookcase.");
        } else {
            System.out.println("Closing the wooden doors of the bookcase.");
        }
    }
}
```

Результаты

```
Number of shelves: 5
Has glass doors: true
Moving the bookcase.
Opening the glass doors of the bookcase.
Closing the glass doors of the bookcase.
```