



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 09.04.01/12 Интеллектуальный анализ больших
данных в системах поддержки принятия решений.

О Т Ч Е Т

по лабораторной работе № 4

Вариант № 7

Название: внутренние классы, интерфейсы

Дисциплина: языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

Ф.А. Лучкин

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2024

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

Задание 1: 7. Создать класс Справочная Служба Общественного Транспорта с внутренним классом, с помощью объектов которого можно хранить информацию о времени, линиях маршрутов и стоимости проезда.

Код класса PublicTransportDirectory:

```
package lab4_var1_7;

import java.util.ArrayList;

interface TransportService {
    void setTime(String time);
    void setRoute(String route);
    void setCost(double cost);

    String getTime();
    String getRoute();
    double getCost();
}

public class PublicTransportDirectory {
    private final ArrayList<TransportInfo> transportInfoList = new ArrayList<>();

    public void addTransportInfo(String time, String route, double cost) {
        TransportInfo info = new TransportInfo();
        info.setTime(time);
        info.setRoute(route);
        info.setCost(cost);
        transportInfoList.add(info);
    }

    public ArrayList<TransportInfo> getTransportInfoList() {
        return transportInfoList;
    }

    public static class TransportInfo implements TransportService {
        private String time;
        private String route;
        private double cost;

        @Override
        public void setTime(String time) {
            this.time = time;
        }

        @Override
        public void setRoute(String route) {
            this.route = route;
        }

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

        public String getTime() {
            return time;
        }
    }
}
```

```

        public String getRoute() {
            return route;
        }

        public double getCost() {
            return cost;
        }
    }
}

```

Код класса Main:

```

package lab4_var1_7;

public class Main {
    public static void main(String[] args) {
        PublicTransportDirectory transportDirectory = new
        PublicTransportDirectory();

        // Добавляем информацию о транспорте
        transportDirectory.addTransportInfo("10:00", "Маршрут 1", 2.50);
        transportDirectory.addTransportInfo("11:30", "Маршрут 2", 3.00);

        // Выводим информацию о транспорте
        for (PublicTransportDirectory.TransportInfo info :
        transportDirectory.getTransportInfoList()) {
            System.out.println("Time: " + info.getTime());
            System.out.println("Route: " + info.getRoute());
            System.out.println("Cost: " + info.getCost());
            System.out.println();
        }
    }
}

```

Работа программы показана на рисунке 1.

```

Time: 10:00
Route: Маршрут 1
Cost: 2.5

Time: 11:30
Route: Маршрут 2
Cost: 3.0

```

Рисунок 1 – Работа программы

Задание 2: создать класс Computer (компьютер) с внутренним классом, с помощью объектов которого можно хранить информацию об операционной системе, процессоре и оперативной памяти.

Код класса ComputerDirectory:

```
package lab4_var1_8;

import java.util.ArrayList;

interface ComputerService {
    void setOperatingSystem(String os);
    void setProcessor(String processor);
    void setRAM(int ram);

    String getOperatingSystem();
    String getProcessor();
    int getRAM();
}

public class ComputerDirectory {
    private final ArrayList<ComputerInfo> computerInfoList = new ArrayList<>();

    public void addComputerInfo(String os, String processor, int ram) {
        ComputerInfo info = new ComputerInfo();
        info.setOperatingSystem(os);
        info.setProcessor(processor);
        info.setRAM(ram);
        computerInfoList.add(info);
    }

    public ArrayList<ComputerInfo> getComputerInfoList() {
        return computerInfoList;
    }

    public static class ComputerInfo implements ComputerService {
        private String operatingSystem;
        private String processor;
        private int ram;

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

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

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

        public String getOperatingSystem() {
            return operatingSystem;
        }

        public String getProcessor() {
            return processor;
        }

        public int getRAM() {
            return ram;
        }
    }
}
```

Код класса Main:

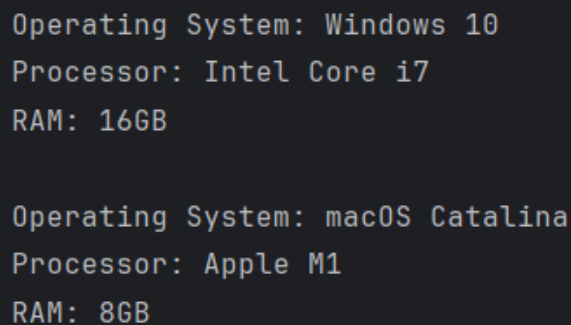
```
package lab4_var1_8;

public class Main {
    public static void main(String[] args) {
        ComputerDirectory computerDirectory = new ComputerDirectory();

        // Добавляем информацию о компьютерах
        computerDirectory.addComputerInfo("Windows 10", "Intel Core i7", 16);
        computerDirectory.addComputerInfo("macOS Catalina", "Apple M1", 8);

        // Выводим информацию о компьютерах
        for (ComputerDirectory.ComputerInfo computerInfo :
computerDirectory.getComputerInfoList()) {
            System.out.println("Operating System: " +
computerInfo.getOperatingSystem());
            System.out.println("Processor: " + computerInfo.getProcessor());
            System.out.println("RAM: " + computerInfo.getRAM() + "GB");
            System.out.println();
        }
    }
}
```

Работа программы показана на рисунке 2.



```
Operating System: Windows 10
Processor: Intel Core i7
RAM: 16GB

Operating System: macOS Catalina
Processor: Apple M1
RAM: 8GB
```

Рисунок 2 – Работа программы

Задание 3: реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов: interface Врач <- class Хирург <- class Нейрохирург.

Код классов:

```
package lab4_var2_7;

// Интерфейс Врач
interface Doctor {
    void treatPatient();
}

// Класс Хирург, реализующий интерфейс Врач
class Surgeon implements Doctor {
    @Override
    public void treatPatient() {
        System.out.println("Хирург проводит операцию");
    }
}
```

```
// Класс Нейрохирург, наследующий от Хирурга
class Neurosurgeon extends Surgeon {
    @Override
    public void treatPatient() {
        System.out.println("Нейрохирург проводит нейрохирургическую операцию");
    }
}
```

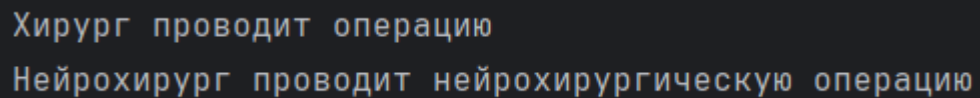
Код класса Main:

```
package lab4_var2_7;

public class Main {
    public static void main(String[] args) {
        Doctor doctor1 = new Surgeon();
        Doctor doctor2 = new Neurosurgeon();

        doctor1.treatPatient();
        doctor2.treatPatient();
    }
}
```

Работа программы показана на рисунке 3.



```
Хирург проводит операцию
Нейрохирург проводит нейрохирургическую операцию
```

Рисунок 3 – Работа программы

Задание 4: реализовать абстрактные классы или интерфейсы, а также наследование и полиморфизм для следующих классов interface Корабль <- class Грузовой Корабль <- class Танкер. Вычислить определитель матрицы.

Код классов:

```
package lab4_var2_8;

// Интерфейс Корабль
interface Ship {
    void sail();
}

// Класс Грузовой Корабль, реализующий интерфейс Корабль
class CargoShip implements Ship {
    @Override
    public void sail() {
        System.out.println("Грузовой корабль плывет по морю");
    }
}

// Класс Танкер, наследующий от Грузового Корабля
class Tanker extends CargoShip {
    @Override
    public void sail() {
        System.out.println("Танкер перевозит нефтепродукты по морю");
    }
}
```

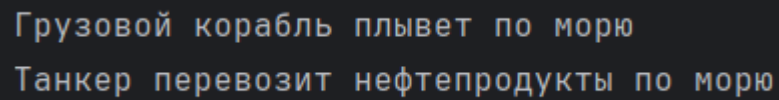
Код модуля Main:

```
package lab4_var2_8;

public class Main {
    public static void main(String[] args) {
        Ship ship1 = new CargoShip();
        Ship ship2 = new Tanker();

        ship1.sail();
        ship2.sail();
    }
}
```

Работа программы показана на рисунке 4.



```
Грузовой корабль плывет по морю
Танкер перевозит нефтепродукты по морю
```

Рисунок 4 – Работа программы

Ссылка на git-репозиторий: https://github.com/FedorLuchkin/Java_bmstu

Вывод: были освоены принципы работы с внутренними классами и интерфейсами на языке программирования Java.