



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 № 3**

**Вариант 12**

**Название:**                    Классы, наследование и полиморфизм

**Дисциплина:**                Языки программирования для работы с большими данными

Студент

ИУ6-23М

(Группа)

\_\_\_\_\_  
(Подпись, дата)

П.А. Мартынюк

(И.О. Фамилия)

Преподаватель

\_\_\_\_\_  
(Подпись, дата)

П.В. Степанов

(И.О. Фамилия)

Москва, 2022

## Цель работы:

Получение навыков работы с классами Java, исследование механизмов наследования и полиморфизма.

## Выполнение:

### Задание 1:

1. Определить класс Вектор размерности  $n$ . Определить несколько конструкторов. Реализовать методы для вычисления модуля вектора, скалярного произведения, сложения, вычитания, умножения на константу. Объявить массив объектов. Написать метод, который для заданной пары векторов будет определять, являются ли они коллинеарными или ортогональными.
2. Определить класс Вектор в  $R^3$ . Реализовать методы для проверки векторов на ортогональность, проверки пересечения не ортогональных векторов, сравнения векторов. Создать массив из  $m$  объектов. Определить, какие из векторов компланарны.

Листинг выполнения подзадачи 1 (файл Vector.java)

```
package task1;

import java.util.ArrayList;
import java.util.Arrays;
import java.util.Objects;

public class Vector {

    private ArrayList<Integer> vector;
    private int dimension;

    public Vector() {
        this.vector = new ArrayList<Integer>();
        this.dimension = 0;
    }

    public Vector(ArrayList<Integer> vector) {
        this.vector = vector;
        this.dimension = this.vector.size();
    }

    public Vector(Integer[] vector) {
        this.vector = new ArrayList<>(Arrays.asList(vector));
        this.dimension = this.vector.size();
    }

    public void addElement(int element){
        this.vector.add(element);
        this.dimension++;
    }
}
```

```

public ArrayList<Integer> getVector() {
    return vector;
}

public void setVector(ArrayList<Integer> vector) {
    this.vector = vector;
}

public int getDimension() {
    return dimension;
}

public void setDimension(int dimension) {
    this.dimension = dimension;
}

@Override
public String toString() {
    return "Vector{" +
        "vector=" + vector +
        ", dimension=" + dimension +
        '}';
}

public double calculateModulus(){
    int sumOfSquares = 0;
    for (Integer elem: this.vector) {
        sumOfSquares += Math.pow(elem, 2);
    }
    return Math.sqrt(sumOfSquares);
}

public int scalarProduct(Vector other){
    if(this.dimension == other.dimension){
        int sumOfMuls = 0;
        for (int i = 0; i < this.dimension; i++) {
            sumOfMuls += this.vector.get(i) * other.vector.get(i);
        }
        return sumOfMuls;
    } else {
        System.out.println("Different demensions of vectors!");
        return -1;
    }
}

public Vector vectorSummation(Vector other){
    if(this.dimension == other.dimension){
        Vector temp = new Vector();
        for (int i = 0; i < this.dimension; i++) {
            temp.addElement(this.vector.get(i) + other.vector.get(i));
        }
        return temp;
    } else {
        System.out.println("Different demensions of vectors!");
        return new Vector();
    }
}

public Vector vectorSubtraction(Vector other) {
    if(this.dimension == other.dimension){
        Vector temp = new Vector();
        for (int i = 0; i < this.dimension; i++) {
            temp.addElement(this.vector.get(i) - other.vector.get(i));
        }
        return temp;
    } else {
        System.out.println("Different demensions of vectors!");
        return new Vector();
    }
}

public Vector constantMultiplication(int k){

```

```

        Vector temp = new Vector();
        for (int i = 0; i < this.dimension; i++) {
            temp.addElement(this.vector.get(i) * k);
        }
        return temp;
    }

    public boolean isCollinear(Vector other){
        if(this.dimension == other.dimension){
            ArrayList<Float> tempArray = new ArrayList<Float>();
            for (int i = 0; i < this.dimension; i++) {
                if(this.vector.get(i) == 0){
                    if(other.vector.get(i) == 0){
                        continue;
                    } else {
                        return false;
                    }
                } else {
                    if(other.vector.get(i) == 0){
                        return false;
                    } else {
                        tempArray.add((float)this.vector.get(i)/other.vector.get(i));
                    }
                }
            }
            boolean isCollinear = true;
            for (int i = 0; i < tempArray.size()-1; i++) {
                if (!Objects.equals(tempArray.get(i), tempArray.get(i + 1))){
                    isCollinear = false;
                    return isCollinear;
                }
            }
            return isCollinear;
        } else {
            System.out.println("Different demensions of vectors!");
            return false;
        }
    }

    public boolean isOrthogonal(Vector other){
        if(scalarProduct(other) == 0){
            return true;
        } else {
            return false;
        }
    }

    public void defineVectorsStatus(Vector other){
        if (this.isCollinear(other)){
            System.out.println("Vectors are collinear");
        } else if(this.isOrthogonal(other)){
            System.out.println("Vectors are orthogonal");
        } else {
            System.out.println("Vectors are neither orthogonal nor collinear");
        }
    }
}

```

#### Листинг выполнения подзадачи 1 (файл MainForVector.java)

```

package task1;
import java.util.ArrayList;
import java.util.Arrays;

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

        Vector vec1 = new Vector(new Integer[]{1, 0, 0});
        Vector vec2 = new Vector(new Integer[]{2, 0, 0});

        System.out.println("First vector: " + vec1);
        System.out.println("Second vector: " + vec2);
    }
}

```

```

        System.out.println("Modulus of first vector: " + vec1.calculateModulus());
        System.out.println("Modulus of second vector: " + vec2.calculateModulus());

        System.out.println("Scalar product of vectors: " + vec1.scalarProduct(vec2));
        System.out.println("Summation of vectors: " + vec1.vectorSummation(vec2));
        System.out.println("Subtraction of vectors: " + vec1.vectorSubtraction(vec2));

        System.out.println("Multiplication of first vector by a constant 3: " +
vec1.constantMultiplication(3));
        System.out.println("Multiplication of second vector by a constant -5: " +
vec2.constantMultiplication(-5));

        Vector[] vectorsArray = {new Vector(new Integer[]{1, 2, 3}),
                                new Vector(),
                                new Vector(new ArrayList<>(Arrays.asList(4, 5, 6)))};

        //System.out.println(vec1.isCollinear(vec2));
        //System.out.println(vec1.isOrthogonal(vec2));

        vec1.defineVectorsStatus(vec2);
    }
}

```

```

Run: MainForVector
C:\Users\polin\.jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
First vector: Vector{vector=[1, 0, 0], dimension=3}
Second vector: Vector{vector=[2, 0, 0], dimension=3}
Modulus of first vector: 1.0
Modulus of second vector: 2.0
Scalar product of vectors: 2
Summation of vectors: Vector{vector=[3, 0, 0], dimension=3}
Subtraction of vectors: Vector{vector=[-1, 0, 0], dimension=3}
Multiplication of first vector by a constant 3: Vector{vector=[3, 0, 0], dimension=3}
Multiplication of second vector by a constant -5: Vector{vector=[-10, 0, 0], dimension=3}
Vectors are collinear
Process finished with exit code 0

```

```

Run: MainForVector
C:\Users\polin\.jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
First vector: Vector{vector=[1, 0, 0], dimension=3}
Second vector: Vector{vector=[0, 1, 0], dimension=3}
Modulus of first vector: 1.0
Modulus of second vector: 1.0
Scalar product of vectors: 0
Summation of vectors: Vector{vector=[1, 1, 0], dimension=3}
Subtraction of vectors: Vector{vector=[1, -1, 0], dimension=3}
Multiplication of first vector by a constant 3: Vector{vector=[3, 0, 0], dimension=3}
Multiplication of second vector by a constant -5: Vector{vector=[0, -5, 0], dimension=3}
Vectors are orthogonal
Process finished with exit code 0

```

Рисунки 1,2 - Результат выполнения кода решения подзадачи 1

#### Листинг выполнения подзадачи 2 (файл VectorDim3.java)

```

package task1;

public class VectorDim3 {

    private float x_start;
    private float x_end;
    private float y_start;

```

```

private float y_end;
private float z_start;
private float z_end;

public VectorDim3() {
}

public VectorDim3(float x_start, float x_end, float y_start, float y_end, float z_start,
float z_end) {
    this.x_start = x_start;
    this.x_end = x_end;
    this.y_start = y_start;
    this.y_end = y_end;
    this.z_start = z_start;
    this.z_end = z_end;
}

public float scalarProduct(VectorDim3 other){
    return (this.x_end - this.x_start) * (other.x_end - other.x_start) + (this.y_end -
this.y_start) * (other.y_end - other.y_start) + (this.z_end - this.z_start) * (other.z_end -
other.z_start);
}

public boolean isOrthogonal(VectorDim3 other){
    if(scalarProduct(other) == 0.){
        return true;
    } else {
        return false;
    }
}

public boolean isCrossed(VectorDim3 other, float eps){
    /*
    x = x_start*t + x_end*(1 - t)
    y = y_start*t + y_end*(1 - t)
    z = z_start*t + z_end*(1 - t)

    this.x_start*t + this.x_end*(1 - t) = other.x_start*s + other.x_end*(1 - s)
    this.y_start*t + this.y_end*(1 - t) = other.y_start*s + other.y_end*(1 - s)
    */
    double s = ((other.y_end - this.y_end)/(this.y_start - this.y_end) - (other.x_end -
this.x_end)/(this.x_start - this.x_end))/((other.x_start - other.x_end)/(this.x_start -
this.x_end) - (other.y_start - other.y_end)/(this.y_start - this.y_end));
    double t = s*((other.x_start - other.x_end)/(this.x_start - this.x_end)) +
(other.x_end - this.x_end)/(this.x_start - this.x_end);

    double left = this.z_start * t + this.z_end * (1 - t);
    double right = other.z_start * s + other.z_end * (1 - s);

    if((left - right) <= eps){
        return true;
    } else {
        return false;
    }
}

public String compareVectors(VectorDim3 other){
    double firstModulus = Math.sqrt(Math.pow((this.x_end - this.x_start), 2) +
Math.pow((this.y_end - this.y_start), 2) + Math.pow((this.z_end - this.z_start), 2));
    double secondModulus = Math.sqrt(Math.pow((other.x_end - other.x_start), 2) +
Math.pow((other.y_end - other.y_start), 2) + Math.pow((other.z_end - other.z_start), 2));
    if(firstModulus > secondModulus){
        return "First vector is greater";
    } else if(firstModulus < secondModulus){
        return "Second vector is greater";
    } else {
        return "Vector have same length";
    }
}

public boolean areComplanar(VectorDim3 other, VectorDim3 another) {
    float this_x = this.x_end - this.x_start;
    float this_y = this.y_end - this.y_start;

```

```

        float this_z = this.z_end - this.z_start;

        float other_x = other.x_end - other.x_start;
        float other_y = other.y_end - other.y_start;
        float other_z = other.z_end - other.z_start;

        float tmpX = this.y * other.z - this.z * other.y;
        float tmpY = -(this.x * other.z - this.z * other.x);
        float tmpZ = this.z * other.y - this.y * other.x;

        float another_x = another.x_end - another.x_start;
        float another_y = another.y_end - another.y_start;
        float another_z = another.z_end - another.z_start;

        float sumOfMuls = tmpX * another_x + tmpY * another_y + tmpZ * another_z;

        if(sumOfMuls == 0.){
            return true;
        } else {
            return false;
        }
    }

    @Override
    public String toString() {
        return "VectorDim3{" +
            "x_start=" + x_start +
            ", x_end=" + x_end +
            ", y_start=" + y_start +
            ", y_end=" + y_end +
            ", z_start=" + z_start +
            ", z_end=" + z_end +
            '}';
    }
}

```

Листинг выполнения подзадачи 2 (файл MainForVectorDim3.java)

```

package task1;

import java.util.ArrayList;
import java.util.Random;
import java.util.Scanner;

public class MainForVectorDim3 {
    public static void main(String[] args) {

        VectorDim3 vec1 = new VectorDim3(0, 1, 0, 0, 0, 0);
        VectorDim3 vec2 = new VectorDim3(0, 0, 0, 1, 0, 1);
        VectorDim3 vec3 = new VectorDim3(0, 1, 0, 1, 0, 1);
        VectorDim3 vec4 = new VectorDim3(0, 1, 0, -1, 0, 1);

        System.out.println("First vector: "+vec1);
        System.out.println("Second vector: "+vec2);

        System.out.println("Vectors 1 and 2:");
        System.out.println( vec1.isOrthogonal(vec2) ? "Vectors are orthogonal" : "Vectors
aren't orthogonal");
        System.out.println(vec1.compareVectors(vec2));

        System.out.println("Third vector: "+vec3);
        System.out.println("Fourth vector: "+vec4);

        System.out.println("Vectors 3 and 4:");
        System.out.println( vec3.isOrthogonal(vec4) ? "Vectors are orthogonal" : "Vectors
aren't orthogonal");
        System.out.println( vec3.isCrossed(vec4, 0.5f) ? "Vectors are crossed" : "Vectors
aren't crossed");
        System.out.println(vec3.compareVectors(vec4));

        VectorDim3 vec5 = new VectorDim3(0, 0, 0, 1, 0, 0);
        VectorDim3 vec6 = new VectorDim3(0, 0, 0, 2, 0, 0);
        VectorDim3 vec7 = new VectorDim3(0, 0, 0, 7, 0, 0);
    }
}

```

```

        System.out.println("Vectors 2 and 3:");
        System.out.println(vec1.areComplanar(vec2, vec3) ? "Vectors are complanar" : "Vectors
aren't complanar");
        System.out.println("Vectors 6 and 7:");
        System.out.println(vec5.areComplanar(vec6, vec7) ? "Vectors are complanar" : "Vectors
aren't complanar");

        ArrayList<VectorDim3> vectorArrayList = new ArrayList<>();

        Scanner in = new Scanner(System.in);
        System.out.print("Enter number m: ");
        int m = in.nextInt();
        Random random = new Random();
        float x_start, x_end, y_start, y_end, z_start, z_end;
        float bound_1 = -3.f;
        float bound_2 = 3.f;
        for (int i = 0; i < m; i++) {
            x_start = random.nextFloat(bound_1, bound_2);
            x_end = random.nextFloat(bound_1, bound_2);
            y_start = random.nextFloat(bound_1, bound_2);
            y_end = random.nextFloat(bound_1, bound_2);
            z_start = random.nextFloat(bound_1, bound_2);
            z_end = random.nextFloat(bound_1, bound_2);
            vectorArrayList.add(new VectorDim3(x_start, x_end, y_start, y_end, z_start,
z_end));
        }
        for (int i = 0; i < m-2; i++) {
            for (int j = i+1; j < m-1; j++) {
                for (int k = j+1; k < m; k++) {

System.out.println(vectorArrayList.get(i).areComplanar(vectorArrayList.get(j),
vectorArrayList.get(k)) ? "Vectors are complanar" : "Vectors aren't complanar");

                }
            }
        }
    }
}

```

```

Run: MainForVectorDim3 x
C:\Users\polin\jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA
First vector: VectorDim3{x_start=0.0, x_end=1.0, y_start=0.0, y_end=0.0, z_start=0.0, z_end=0.0}
Second vector: VectorDim3{x_start=0.0, x_end=0.0, y_start=0.0, y_end=1.0, z_start=0.0, z_end=1.0}
Vectors 1 and 2:
Vectors are orthogonal
Second vector is greater
Third vector: VectorDim3{x_start=0.0, x_end=1.0, y_start=0.0, y_end=1.0, z_start=0.0, z_end=1.0}
Fourth vector: VectorDim3{x_start=0.0, x_end=1.0, y_start=0.0, y_end=-1.0, z_start=0.0, z_end=1.0}
Vectors 3 and 4:
Vectors aren't orthogonal
Vectors are crossed
Vector have same length
Vectors 2 and 3:
Vectors aren't complanar
Vectors 6 and 7:
Vectors are complanar
Enter number m: 3
Vectors aren't complanar
Process finished with exit code 0

```

Рисунок 3 - Результат выполнения кода решения подзадачи 2



## Задание 2:

Создать классы, спецификации которых приведены ниже. Определить конструкторы и методы `setТип()`, `getТип()`, `toString()`. Определить дополнительно методы в классе, создающем массив объектов. Задать критерий выбора данных и вывести эти данные на консоль.

1. Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести: а) список пациентов, имеющих данный диагноз; б) список пациентов, номер медицинской карты у которых находится в заданном интервале.
2. Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: а) список абитуриентов, имеющих неудовлетворительные оценки; б) список абитуриентов, средний балл у которых выше заданного; в) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

Листинг выполнения подзадачи 1 (файл Patient.java)

```
package task2;

public class Patient {
    private int id;
    private String name;
    private String surname;
    private String lastname;
    private String address;
    private String phone;
    private int cardNumber;
    private String diagnosis;

    public Patient() {
    }

    public Patient(int id, String name, String surname, String lastname, String address,
String phone, int cardNumber, String diagnosis) {
        this.id = id;
        this.name = name;
        this.surname = surname;
        this.lastname = lastname;
        this.address = address;
        this.phone = phone;
        this.cardNumber = cardNumber;
        this.diagnosis = diagnosis;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }
}
```

```

    }

    public String getName() {
        return name;
    }

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

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getLastname() {
        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getAddress() {
        return address;
    }

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

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public int getCardNumber() {
        return cardNumber;
    }

    public void setCardNumber(int cardNumber) {
        this.cardNumber = cardNumber;
    }

    public String getDiagnosis() {
        return diagnosis;
    }

    public void setDiagnosis(String diagnosis) {
        this.diagnosis = diagnosis;
    }

    @Override
    public String toString() {
        return "Patient{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", surname='" + surname + '\'' +
            ", lastname='" + lastname + '\'' +
            ", address='" + address + '\'' +
            ", phone='" + phone + '\'' +
            ", cardNumber=" + cardNumber +
            ", diagnosis='" + diagnosis + '\'' +
            '}';
    }
}

```

## Листинг выполнения подзадачи 1 (файл MainForPatient.java)

```
package task2;

import java.util.ArrayList;

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

        Patient[] patientsArray = createPatientsArray();
        System.out.println("Patients:");
        for (Patient p: patientsArray) {
            System.out.println(p);
        }

        Patient[] patientsWithDiabetes = chooseByDiagnosis(patientsArray, "Diabetes");
        System.out.println();
        System.out.println("Patients with Diabetes:");
        for (Patient p: patientsWithDiabetes) {
            System.out.println(p);
        }
        Patient[] patientsInRange = chooseByCardNumber(patientsArray, 130, 140);
        System.out.println();
        System.out.println("Patients with CardNumbers in range 130...140:");
        for (Patient p: patientsInRange) {
            System.out.println(p);
        }
    }

    private static Patient[] createPatientsArray(){
        Patient p1 = new Patient(1,"Ivan", "Ivanov", "Ivanovich", "House 5", "8-968-374-26-47", 132, "Diabetes");
        Patient p2 = new Patient(2,"Petr", "Petrov", "Petrovich", "House 3", "8-969-375-27-74", 148, "COVID-19");
        Patient p3 = new Patient(3,"Dmitry", "Smirnov", "Ivanovich", "House 9", "8-977-234-86-07", 119, "Diabetes");
        Patient p4 = new Patient(4,"Ivan", "Smirnov", "Andreevich", "House 5", "8-978-306-36-43", 135, "COVID-19");
        Patient p5 = new Patient(5,"Alexander", "Ivanov", "Ilich", "House 11", "8-961-333-28-17", 138, "Flu");
        return new Patient[]{p1, p2, p3, p4, p5};
    }

    private static Patient[] chooseByDiagnosis(Patient[] patientsArray, String diagnosis){
        ArrayList<Patient> newPatientsArray = new ArrayList<>();
        for (int i = 0; i < patientsArray.length; i++) {
            if(patientsArray[i].getDiagnosis().equals(diagnosis)){
                newPatientsArray.add(patientsArray[i]);
            }
        }
        return (Patient[]) newPatientsArray.toArray(new Patient[newPatientsArray.size()]);
    }

    private static Patient[] chooseByCardNumber(Patient[] patientsArray, int startBound, int endBound){
        ArrayList<Patient> newPatientsArray = new ArrayList<>();
        for (int i = 0; i < patientsArray.length; i++) {
            if(patientsArray[i].getCardNumber() >= startBound && patientsArray[i].getCardNumber() <= endBound){
                newPatientsArray.add(patientsArray[i]);
            }
        }
        return (Patient[]) newPatientsArray.toArray(new Patient[newPatientsArray.size()]);
    }
}
```

```
Run: MainForPatient
C:\Users\polin\.jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.2\lib\idea_rt.jar=52361:C:\Program Files\JetBrain
Patients:
Patient{id=1, name='Ivan', surname='Ivanov', lastname='Ivanovich', address='House 5', phone='8-968-374-26-47', cardNumber=132, diagnosis='Diabetes'}
Patient{id=2, name='Petr', surname='Petrov', lastname='Petrovich', address='House 3', phone='8-969-375-27-74', cardNumber=148, diagnosis='COVID-19'}
Patient{id=3, name='Dmitry', surname='Smirnov', lastname='Ivanovich', address='House 9', phone='8-977-234-86-07', cardNumber=119, diagnosis='Diabetes'}
Patient{id=4, name='Ivan', surname='Smirnov', lastname='Andreevich', address='House 5', phone='8-978-306-36-43', cardNumber=135, diagnosis='COVID-19'}
Patient{id=5, name='Alexander', surname='Ivanov', lastname='Ilich', address='House 11', phone='8-961-333-28-17', cardNumber=138, diagnosis='Flu'}

Patients with Diabetes:
Patient{id=1, name='Ivan', surname='Ivanov', lastname='Ivanovich', address='House 5', phone='8-968-374-26-47', cardNumber=132, diagnosis='Diabetes'}
Patient{id=3, name='Dmitry', surname='Smirnov', lastname='Ivanovich', address='House 9', phone='8-977-234-86-07', cardNumber=119, diagnosis='Diabetes'}

Patients with CardNumbers in range 130...140:
Patient{id=1, name='Ivan', surname='Ivanov', lastname='Ivanovich', address='House 5', phone='8-968-374-26-47', cardNumber=132, diagnosis='Diabetes'}
Patient{id=4, name='Ivan', surname='Smirnov', lastname='Andreevich', address='House 5', phone='8-978-306-36-43', cardNumber=135, diagnosis='COVID-19'}
Patient{id=5, name='Alexander', surname='Ivanov', lastname='Ilich', address='House 11', phone='8-961-333-28-17', cardNumber=138, diagnosis='Flu'}

Process finished with exit code 0
```

Рисунок 4 - Результат выполнения кода решения подзадачи 1

## Листинг выполнения подзадачи 2 (файл Abiturient.java)

```
package task2;

import java.util.ArrayList;

public class Abiturient {
    private int id;
    private String name;
    private String surname;
    private String lastname;
    private String address;
    private String phone;
    private ArrayList<Integer> marks;

    public Abiturient() {
    }

    public Abiturient(int id, String name, String surname, String lastname, String address,
String phone, ArrayList<Integer> marks) {
        this.id = id;
        this.name = name;
        this.surname = surname;
        this.lastname = lastname;
        this.address = address;
        this.phone = phone;
        this.marks = marks;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public String getSurname() {
        return surname;
    }

    public void setSurname(String surname) {
        this.surname = surname;
    }

    public String getLastName() {
```

```

        return lastname;
    }

    public void setLastname(String lastname) {
        this.lastname = lastname;
    }

    public String getAddress() {
        return address;
    }

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

    public String getPhone() {
        return phone;
    }

    public void setPhone(String phone) {
        this.phone = phone;
    }

    public ArrayList<Integer> getMarks() {
        return marks;
    }

    public void setMarks(ArrayList<Integer> marks) {
        this.marks = marks;
    }

    @Override
    public String toString() {
        return "Abiturient{" +
            "id=" + id +
            ", name='" + name + '\'' +
            ", surname='" + surname + '\'' +
            ", lastname='" + lastname + '\'' +
            ", address='" + address + '\'' +
            ", phone='" + phone + '\'' +
            ", marks=" + marks +
            '}';
    }
}

```

### Листинг выполнения подзадачи 2 (файл MainForAbiturient.java)

```

package task2;

import java.util.*;

public class MainForAbiturient {

    public static void main(String[] args) {

        Abiturient[] abiturientsArray = createAbiturientsArray();
        System.out.println("Abiturients:");
        for (Abiturient a: abiturientsArray) {
            System.out.println(a);
        }
        Abiturient[] abiturientsWithNeuds = chooseWithNeuds(abiturientsArray);
        System.out.println();
        System.out.println("Abiturients with neuds:");
        for (Abiturient a: abiturientsWithNeuds) {
            System.out.println(a);
        }

        Abiturient[] abiturientsWithHigherAVG = chooseHigherAVGMark(abiturientsArray, 4f);
        System.out.println();
        System.out.println("Abiturients with average mark higher then 4:");
        for (Abiturient a: abiturientsWithHigherAVG) {
            System.out.println(a);
        }
    }
}

```

```

        Abiturient[] abiturientsBestN = chooseBest(abiturientsArray, 2);
        System.out.println();
        System.out.println("Best 2 abiturients:");
        for (Abiturient a: abiturientsBestN) {
            System.out.println(a);
        }
    }

    private static Abiturient[] createAbiturientsArray(){
        Abiturient a1 = new Abiturient(1,"Ivan", "Ivanov", "Ivanovich", "House 5", "8-968-374-
26-47", new ArrayList<Integer>(Arrays.asList(3, 2, 5)));
        Abiturient a2 = new Abiturient(2,"Petr", "Petrov", "Petrovich", "House 3", "8-969-375-
27-74", new ArrayList<Integer>(Arrays.asList(4, 4, 5)));
        Abiturient a3 = new Abiturient(3,"Dmitry", "Smirnov", "Ivanovich", "House 9", "8-977-
234-86-07", new ArrayList<Integer>(Arrays.asList(5, 4, 5)));
        Abiturient a4 = new Abiturient(4,"Ivan", "Smirnov", "Andreevich", "House 5", "8-978-
306-36-43", new ArrayList<Integer>(Arrays.asList(3, 2, 4)));
        Abiturient a5 = new Abiturient(5,"Alexander", "Ivanov", "Ilich", "House 11", "8-961-
333-28-17", new ArrayList<Integer>(Arrays.asList(5, 5, 5)));
        return new Abiturient[]{a1, a2, a3, a4, a5};
    }

    private static Abiturient[] chooseWithNeuds(Abiturient[] abiturientsArray){
        ArrayList<Abiturient> newAbiturientsArray = new ArrayList<>();
        for (int i = 0; i < abiturientsArray.length; i++) {
            if (abiturientsArray[i].getMarks().contains(2)) {
                newAbiturientsArray.add(abiturientsArray[i]);
            }
        }
        return (Abiturient[]) newAbiturientsArray.toArray(new
Abiturient[newAbiturientsArray.size()]);
    }

    private static Abiturient[] chooseHigherAVGMark(Abiturient[] abiturientsArray, float
mark){
        ArrayList<Abiturient> newAbiturientsArray = new ArrayList<>();
        for (int i = 0; i < abiturientsArray.length; i++) {
            float avg = 0;
            for (Integer m : abiturientsArray[i].getMarks()) {
                avg += m;
            }
            avg = avg/abiturientsArray[i].getMarks().size();
            if (avg > mark){
                newAbiturientsArray.add(abiturientsArray[i]);
            }
        }
        return (Abiturient[]) newAbiturientsArray.toArray(new
Abiturient[newAbiturientsArray.size()]);
    }

    private static Abiturient[] chooseBest(Abiturient[] abiturientsArray, Integer n){
        ArrayList<Abiturient> newAbiturientsArray = new ArrayList<>();

        SortedMap<Float, ArrayList<Abiturient>> map = new TreeMap<>();
        for (int i = 0; i < abiturientsArray.length; i++){
            float avg = 0;
            for (Integer m : abiturientsArray[i].getMarks()) {
                avg += m;
            }
            avg = avg / abiturientsArray[i].getMarks().size();

            if (map.containsKey(avg)) {
                map.get(avg).add(abiturientsArray[i]);
            } else {
                map.put(avg, new ArrayList<>());
                map.get(avg).add(abiturientsArray[i]);
            }
        }
        System.out.println(map);

        int j = 0;

```

```

        int avg_num = -1;
        List<Float> floatList = new ArrayList<Float>(map.keySet());
        Collections.reverse(floatList);
        while (j < n){
            avg_num++;
            for (int i = 0; i < map.get(floatList.get(avg_num)).size(); i++) {
                newAbiturientsArray.add(map.get(floatList.get(avg_num)).get(i));
                j++;
            }
        }

        return (Abiturient[]) newAbiturientsArray.toArray(new
Abiturient[newAbiturientsArray.size()]);
    }
}

```

```

Run: MainForAbiturient
C:\Users\polin\jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.2\lib\idea_rt.jar=52417:C:\Program Files\JetBrain
Abiturients:
Abiturient{id=1, name='Ivan', surname='Ivanov', lastname='Ivanovich', address='House 5', phone='8-968-374-26-47', marks=[3, 2, 5]}
Abiturient{id=2, name='Petr', surname='Petrov', lastname='Petrovich', address='House 3', phone='8-969-375-27-74', marks=[4, 4, 5]}
Abiturient{id=3, name='Dmitry', surname='Smirnov', lastname='Ivanovich', address='House 9', phone='8-977-234-86-07', marks=[5, 4, 5]}
Abiturient{id=4, name='Ivan', surname='Smirnov', lastname='Andreevich', address='House 5', phone='8-978-306-36-43', marks=[3, 2, 4]}
Abiturient{id=5, name='Alexander', surname='Ivanov', lastname='Ilich', address='House 11', phone='8-961-333-28-17', marks=[5, 5, 5]}

Abiturients with neuds:
Abiturient{id=1, name='Ivan', surname='Ivanov', lastname='Ivanovich', address='House 5', phone='8-968-374-26-47', marks=[3, 2, 5]}
Abiturient{id=4, name='Ivan', surname='Smirnov', lastname='Andreevich', address='House 5', phone='8-978-306-36-43', marks=[3, 2, 4]}

Abiturients with average mark higher then 4:
Abiturient{id=2, name='Petr', surname='Petrov', lastname='Petrovich', address='House 3', phone='8-969-375-27-74', marks=[4, 4, 5]}
Abiturient{id=3, name='Dmitry', surname='Smirnov', lastname='Ivanovich', address='House 9', phone='8-977-234-86-07', marks=[5, 4, 5]}
Abiturient{id=5, name='Alexander', surname='Ivanov', lastname='Ilich', address='House 11', phone='8-961-333-28-17', marks=[5, 5, 5]}
{3.0=[Abiturient{id=4, name='Ivan', surname='Smirnov', lastname='Andreevich', address='House 5', phone='8-978-306-36-43', marks=[3, 2, 4]}], 3.3333333=[Abitur
Best 2 abiturients:
Abiturient{id=5, name='Alexander', surname='Ivanov', lastname='Ilich', address='House 11', phone='8-961-333-28-17', marks=[5, 5, 5]}
Abiturient{id=3, name='Dmitry', surname='Smirnov', lastname='Ivanovich', address='House 9', phone='8-977-234-86-07', marks=[5, 4, 5]}

Process finished with exit code 0

```

Рисунок 5 - Результат выполнения кода решения подзадачи 2

### Задание 3:

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString().

1. Создать объект класса Одномерный массив, используя класс Массив. Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).
2. Создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

### Листинг выполнения подзадачи 1 (файл Array.java)

```
package task3;

public interface Array {

    void summation(Object o);
    void subtraction(Object o);
    void multiplication(Object o);
}
```

### Листинг выполнения подзадачи 1 (файл Array1Dim.java)

```
package task3;

import java.util.ArrayList;
import java.util.Objects;

public class Array1Dim implements Array{

    private ArrayList<Float> content;

    public Array1Dim() {
    }

    public Array1Dim(ArrayList<Float> content) {
        this.content = content;
    }

    @Override
    public void summation(Object o) {
        if(o.getClass().equals(Array1Dim.class)){
            if(this.content.size() == ((Array1Dim) o).content.size()){
                for (int i = 0; i < content.size(); i++) {
                    this.content.set(i, (this.content.get(i) + ((Array1Dim)
o).content.get(i)));
                }
            } else {
                System.out.println("Arrays length mismatch!");
            }
        } else {
            System.out.println("Class mismatch!");
        }
    }

    @Override
    public void subtraction(Object o) {
        if(o.getClass().equals(Array1Dim.class)){
            if(this.content.size() == ((Array1Dim) o).content.size()){
                for (int i = 0; i < content.size(); i++) {
                    this.content.set(i, (this.content.get(i) - ((Array1Dim)
o).content.get(i)));
                }
            } else {
                System.out.println("Arrays length mismatch!");
            }
        } else {
            System.out.println("Class mismatch!");
        }
    }

    @Override
    public void multiplication(Object o) {
        if(o.getClass().equals(Array1Dim.class)){
            if(this.content.size() == ((Array1Dim) o).content.size()){
                for (int i = 0; i < content.size(); i++) {
                    this.content.set(i, (this.content.get(i) * ((Array1Dim)
o).content.get(i)));
                }
            } else {
                System.out.println("Arrays length mismatch!");
            }
        }
    }
}
```



```

    } else {
        System.out.println("Class mismatch!");
    }
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    Array1Dim array1Dim = (Array1Dim) o;
    return Objects.equals(content, array1Dim.content);
}

@Override
public int hashCode() {
    return Objects.hash(content);
}

@Override
public String toString() {
    return "Array1Dim{" +
        "content=" + content +
        '}';
}

public Array1Dim getClone() {
    return new Array1Dim((ArrayList) (this.content.clone()));
}
}

```

Листинг выполнения подзадачи 1 (файл MainForArrays.java)

```

package task3;

import java.util.ArrayList;
import java.util.Arrays;

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

        Array1Dim ar1 = new Array1Dim(new ArrayList<Float>(Arrays.asList(1.f, 2.f, 3.f)));
        Array1Dim ar2 = new Array1Dim(new ArrayList<Float>(Arrays.asList(4.f, 5.f, 6.f)));

        System.out.println("Array 1 : ");
        System.out.println(ar1);
        System.out.println("Array 2 : ");
        System.out.println(ar2);

        Array1Dim ar_sum = ar1.getClone();
        Array1Dim ar_sub = ar1.getClone();
        Array1Dim ar_mul = ar1.getClone();

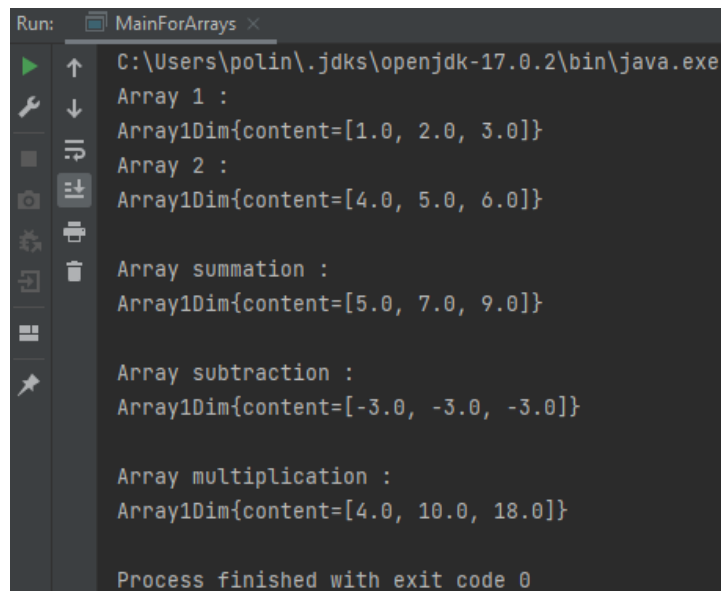
        System.out.println();
        System.out.println("Array summation : ");
        ar_sum.summation(ar2);
        System.out.println(ar_sum);

        System.out.println();
        System.out.println("Array subtraction : ");
        ar_sub.subtraction(ar2);
        System.out.println(ar_sub);

        System.out.println();
        System.out.println("Array multiplication : ");
        ar_mul.multiplication(ar2);
        System.out.println(ar_mul);

    }
}

```



```
Run: MainForArrays x
C:\Users\polin\.jdk\openjdk-17.0.2\bin\java.exe
Array 1 :
Array1Dim{content=[1.0, 2.0, 3.0]}
Array 2 :
Array1Dim{content=[4.0, 5.0, 6.0]}

Array summation :
Array1Dim{content=[5.0, 7.0, 9.0]}

Array subtraction :
Array1Dim{content=[-3.0, -3.0, -3.0]}

Array multiplication :
Array1Dim{content=[4.0, 10.0, 18.0]}

Process finished with exit code 0
```

Рисунок 6 - Результат выполнения кода решения подзадачи 1

#### Листинг выполнения подзадачи 2 (файл Number.java)

```
package task3;

import java.util.Objects;

public class Number {

    private int number;

    public Number() {
        this.number = 0;
    }

    public Number(int number) {
        this.number = number;
    }

    public Number summation(Number other){
        return new Number(this.number + other.number);
    }

    public Number subtraction(Number other){
        return new Number(this.number - other.number);
    }

    public Number multiplication(Number other){
        return new Number(this.number * other.number);
    }

    public Number division(Number other){
        return new Number(this.number / other.number);
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Number number1 = (Number) o;
        return number == number1.number;
    }

    @Override
    public int hashCode() {
        return Objects.hash(number);
    }

    @Override
```

```

    public String toString() {
        return "Number{" +
            "number=" + number +
            '}';
    }

    public int getNumber() {
        return number;
    }
}

```

## Листинг выполнения подзадачи 2 (файл Fraction.java)

```

package task3;

import java.util.Objects;

public class Fraction {

    private Number numerator;
    private Number denominator;

    public Fraction() {
    }

    public Fraction(Number numerator, Number denominator) {
        this.numerator = numerator;
        this.denominator = denominator;
    }

    public Fraction summation(Fraction other){
        Fraction result = new
Fraction(other.denominator.multiplication(this.numerator).summation(this.denominator.multiplic
ation(other.numerator)), this.denominator.multiplication(other.denominator));
        result.normalize();
        return result;
    }

    public Fraction subtraction(Fraction other){
        Fraction result = new
Fraction(other.denominator.multiplication(this.numerator).subtraction(this.denominator.multiplic
ation(other.numerator)), this.denominator.multiplication(other.denominator));
        result.normalize();
        return result;
    }

    public Fraction multiplication(Fraction other){

        Fraction result = new Fraction(this.numerator.multiplication(other.numerator),
this.denominator.multiplication(other.denominator));
        result.normalize();
        return result;
    }

    public Fraction division(Fraction other){

        Fraction temp = new Fraction(other.denominator, other.numerator);
        Fraction result = this.multiplication(temp);
        result.normalize();
        return result;
    }

    public Number getNumerator() {
        return numerator;
    }

    public void setNumerator(Number numerator) {
        this.numerator = numerator;
    }

    public Number getDenominator() {

```

```

        return denominator;
    }

    public void setDenominator(Number denominator) {
        this.denominator = denominator;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Fraction fraction = (Fraction) o;
        return Objects.equals(numerator, fraction.numerator) && Objects.equals(denominator,
fraction.denominator);
    }

    @Override
    public int hashCode() {
        return Objects.hash(numerator, denominator);
    }

    @Override
    public String toString() {
        return "Fraction{" +
            numerator +
            " / " + denominator +
            '}';
    }

    private void normalize(){

        if(this.numerator.getNumber() > this.denominator.getNumber()){
            if(this.numerator.getNumber() % this.denominator.getNumber() == 0){
                this.numerator = this.numerator.division(this.denominator);
                this.denominator = new Number(1);

            } else {
                int i = 2;
                while (i <= (int) Math.sqrt(this.denominator.getNumber())) {
                    if(this.numerator.getNumber() % i == 0 && this.denominator.getNumber() % i
== 0){

                        this.numerator = this.numerator.division(new Number(i));
                        this.denominator = this.denominator.division(new Number(i));
                    } else {
                        i++;
                    }
                }
            }
        } else if(this.numerator.getNumber() < this.denominator.getNumber()) {
            if(this.denominator.getNumber() % this.numerator.getNumber() == 0){
                this.denominator = this.denominator.division(this.numerator);
                this.numerator = new Number(1);
            } else {
                int i = 2;
                while (i <= (int) Math.sqrt(this.numerator.getNumber())) {
                    if(this.numerator.getNumber() % i == 0 && this.denominator.getNumber() % i
== 0){

                        this.numerator = this.numerator.division(new Number(i));
                        this.denominator = this.denominator.division(new Number(i));
                    } else {
                        i++;
                    }
                }
            }
        } else {
            this.numerator = new Number(1);
            this.denominator = new Number(1);
        }
    }
}

```

## Листинг выполнения подзадачи 2 (файл MainForFraction.java)

```
package task3;

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

        Fraction fr1 = new Fraction(new Number(1), new Number(2));
        Fraction fr2 = new Fraction(new Number(1), new Number(4));

        System.out.println("Fraction 1:");
        System.out.println(fr1);
        System.out.println("Fraction 2:");
        System.out.println(fr2);

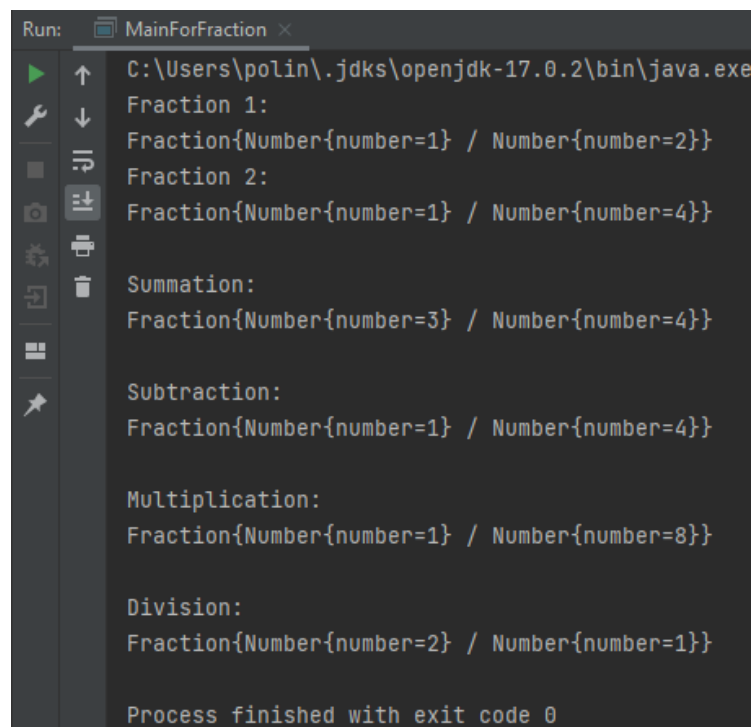
        System.out.println();
        Fraction fr_sum = fr1.summation(fr2);
        System.out.println("Summation: ");
        System.out.println(fr_sum);

        System.out.println();
        Fraction fr_sub = fr1.subtraction(fr2);
        System.out.println("Subtraction: ");
        System.out.println(fr_sub);

        System.out.println();
        Fraction fr_mul = fr1.multiplication(fr2);
        System.out.println("Multiplication: ");
        System.out.println(fr_mul);

        System.out.println();
        Fraction fr_div = fr1.division(fr2);
        System.out.println("Division: ");
        System.out.println(fr_div);

    }
}
```



```
Run: MainForFraction x
C:\Users\polin\.jdk\openjdk-17.0.2\bin\java.exe
Fraction 1:
Fraction{Number{number=1} / Number{number=2}}
Fraction 2:
Fraction{Number{number=1} / Number{number=4}}

Summation:
Fraction{Number{number=3} / Number{number=4}}

Subtraction:
Fraction{Number{number=1} / Number{number=4}}

Multiplication:
Fraction{Number{number=1} / Number{number=8}}

Division:
Fraction{Number{number=2} / Number{number=1}}

Process finished with exit code 0
```

Рисунок 7 - Результат выполнения кода решения подзадачи 2

#### Задание 4:

Построить модель программной системы.

1. Система Платежи. Клиент имеет Счет в банке и Кредитную Карту (КК). Клиент может оплатить Заказ, сделать платеж на другой Счет, заблокировать КК и аннулировать Счет. Администратор может заблокировать КК за превышение кредита.
2. Система Больница. Пациенту назначается лечащий Врач. Врач может сделать назначение Пациенту (процедуры, лекарства, операции). Медсестра или другой Врач выполняют назначение. Пациент может быть выписан из Больницы по окончании лечения, при нарушении режима или при иных обстоятельствах.

Листинг выполнения подзадачи 1 (файл Client.java)

```
package task4;

import java.util.Objects;

public class Client {

    private String name;
    private int id;

    public Client() {
    }

    public Client(String name, int id) {
        this.name = name;
        this.id = id;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Client client = (Client) o;
        return id == client.id && Objects.equals(name, client.name);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, id);
    }

    public String getName() {
        return name;
    }

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

    @Override
```

```

    public String toString() {
        return "Client{" +
            "name='" + name + '\'' +
            ", id=" + id +
            '}';
    }
}

```

### Листинг выполнения подзадачи 1 (файл Bank.java)

```

package task4;

import java.util.HashMap;

public class Bank {
    private HashMap<Integer, Client> account_client;
    private HashMap<Integer, Integer> account_sum;
    private HashMap<Integer, Integer> account_card;
    private HashMap<Integer, Boolean> card_active;

    public Bank() {
        account_client = new HashMap<>();
        account_sum = new HashMap<>();
        account_card = new HashMap<>();
        card_active = new HashMap<>();
    }

    public void addClient(Integer acc_id, Client client){
        account_client.put(acc_id, client);
        account_sum.put(acc_id, 0);
    }

    public void addCard(Integer acc_id, Integer cardNum){
        account_card.put(acc_id, cardNum);
        card_active.put(cardNum, true);
    }

    public void addMoney(Integer acc_id, Integer amount){
        account_sum.put(acc_id, account_sum.get(acc_id) + amount);
    }

    public void blockCard(Integer cardNum){
        card_active.put(cardNum, false);
    }

    public void payOrder(Integer acc_id, Integer amount){
        account_sum.put(acc_id, account_sum.get(acc_id) - amount);
    }

    public void switchToAccount(Integer acc1_id, Integer acc2_id, Integer amount){
        account_sum.put(acc1_id, account_sum.get(acc1_id) - amount);
        account_sum.put(acc2_id, account_sum.get(acc2_id) + amount);
    }

    public void cancelAccount(Integer acc_id){
        blockCard(account_card.get(acc_id));
        account_client.remove(acc_id);
        account_sum.remove(acc_id);
        account_card.remove(acc_id);
    }

    public void checkCredit(Integer acc_id){
        if(account_sum.get(acc_id) < -10000000){
            blockCard(account_card.get(acc_id));
        }
    }

    @Override
    public String toString() {
        return "Bank{" +
            "account_client=" + account_client +
            "\n    account_sum=" + account_sum +
            "\n    account_card=" + account_card +

```

```

        "\n    card_active=" + card_active +
        '\n';
    }
}

```

## Листинг выполнения подзадачи 1 (файл MainForBank.java)

```

package task4;

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

        Client cl1 = new Client("Ivanov Ivan Ivanovich", 1);
        Client cl2 = new Client("Petrov Petr Petrovich", 2);
        Client cl3 = new Client("Smirnov Igor Igorevich", 3);

        Bank bank = new Bank();

        bank.addClient( 4, cl1);
        bank.addCard(4, 531);
        bank.addMoney(4, 15000);

        bank.addClient( 5, cl2);
        bank.addCard(5, 129);
        bank.addMoney(5, 3000);

        bank.addClient( 6, cl3);
        bank.addCard(6, 773);
        bank.addMoney(6, 100);

        System.out.println("Added clients:");
        System.out.println(bank);

        bank.payOrder(4, 5000);
        bank.switchToAccount(4, 5, 1000);

        System.out.println("Payments are done:");
        System.out.println(bank);

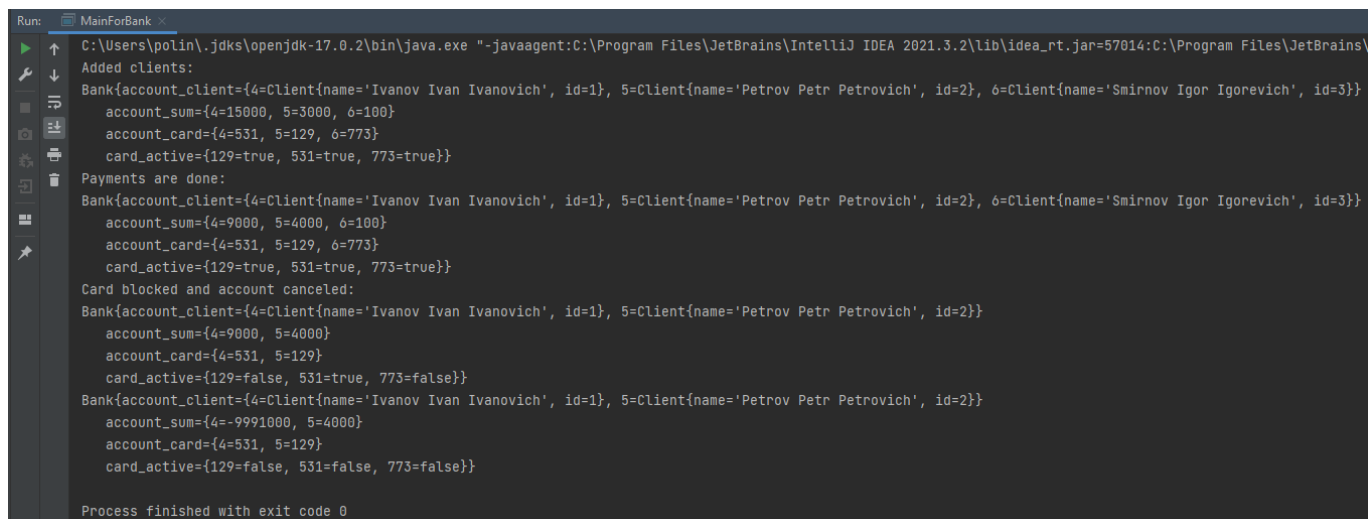
        bank.blockCard(129);
        bank.cancelAccount(6);

        System.out.println("Card blocked and account canceled:");
        System.out.println(bank);

        bank.payOrder(4, 10000000);
        bank.checkCredit(4);

        System.out.println(bank);
    }
}

```



```

Run: MainForBank x
C:\Users\polin\.jdk\openjdk-17.0.2\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.2\lib\idea_rt.jar=57014:C:\Program Files\JetBrains\
Added clients:
Bank{account_client={4=Client{name='Ivanov Ivan Ivanovich', id=1}, 5=Client{name='Petrov Petr Petrovich', id=2}, 6=Client{name='Smirnov Igor Igorevich', id=3}},
  account_sum={4=15000, 5=3000, 6=100},
  account_card={4=531, 5=129, 6=773},
  card_active={129=true, 531=true, 773=true}}
Payments are done:
Bank{account_client={4=Client{name='Ivanov Ivan Ivanovich', id=1}, 5=Client{name='Petrov Petr Petrovich', id=2}, 6=Client{name='Smirnov Igor Igorevich', id=3}},
  account_sum={4=9000, 5=4000, 6=100},
  account_card={4=531, 5=129, 6=773},
  card_active={129=true, 531=true, 773=true}}
Card blocked and account canceled:
Bank{account_client={4=Client{name='Ivanov Ivan Ivanovich', id=1}, 5=Client{name='Petrov Petr Petrovich', id=2}},
  account_sum={4=9000, 5=4000},
  account_card={4=531, 5=129},
  card_active={129=false, 531=true, 773=false}}
Bank{account_client={4=Client{name='Ivanov Ivan Ivanovich', id=1}, 5=Client{name='Petrov Petr Petrovich', id=2}},
  account_sum={4=-9991000, 5=4000},
  account_card={4=531, 5=129},
  card_active={129=false, 531=false, 773=false}}
Process finished with exit code 0

```

Рисунок 8 - Результат выполнения кода решения подзадачи 1



## Листинг выполнения подзадачи 2 (файл Patient.java)

```
package task4;

import java.util.Objects;

public class Patient {
    private String name;
    private int id;
    private boolean inHospital;
    private String reason;

    public Patient() {
    }

    public Patient(String name, int id ) {
        this.name = name;
        this.id = id;
        this.inHospital = true;
        this.reason = "";
    }

    public String getName() {
        return name;
    }

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

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public boolean isInHospital() {
        return inHospital;
    }

    public void setInHospital(boolean inHospital) {
        this.inHospital = inHospital;
    }

    public String getReason() {
        return reason;
    }

    public void setReason(String reason) {
        this.reason = reason;
    }

    public void dismiss(String reason){
        this.inHospital = false;
        this.reason = reason;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Patient patient = (Patient) o;
        return id == patient.id && inHospital == patient.inHospital && Objects.equals(name,
patient.name) && Objects.equals(reason, patient.reason);
    }

    @Override
    public int hashCode() {
        return Objects.hash(name, id, inHospital, reason);
    }
}
```

```

@Override
public String toString() {
    return "Patient{" +
        "name='" + name + '\'' +
        ", id=" + id +
        ", inHospital=" + inHospital +
        ", reason='" + reason + '\'' +
        '}';
}
}

```

## Листинг выполнения подзадачи 2 (файл Staff.java)

```

package task4;

import java.util.Objects;

public class Staff {
    private int id;
    private String name;
    private String position;

    public Staff() {
    }

    public Staff(int id, String name, String position) {
        this.id = id;
        this.name = name;
        this.position = position;
    }

    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getName() {
        return name;
    }

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

    public String getPosition() {
        return position;
    }

    public void setPosition(String position) {
        this.position = position;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Staff staff = (Staff) o;
        return id == staff.id && Objects.equals(name, staff.name) && Objects.equals(position,
staff.position);
    }

    @Override
    public int hashCode() {
        return Objects.hash(id, name, position);
    }

    @Override
    public String toString() {
        return "Staff{" +
            "id=" + id +

```

```

        ", name='" + name + '\'' +
        ", position='" + position + '\'' +
        '}' ;
    }
}

```

## Листинг выполнения подзадачи 2 (файл Assignment.java)

```

package task4;

import java.util.ArrayList;
import java.util.Objects;

public class Assignment {
    private ArrayList<String> procedures;
    private ArrayList<String> drugs;
    private ArrayList<String> surgeries;
    private boolean done;

    public Assignment() {
        this.procedures = new ArrayList<>();
        this.drugs = new ArrayList<>();
        this.surgeries = new ArrayList<>();
        this.done = false;
    }

    public void addProcedure(String proc){
        this.procedures.add(proc);
    }

    public void addDrug(String drug){
        this.drugs.add(drug);
    }

    public void addSurgery(String surg){
        this.surgeries.add(surg);
    }

    public void complete(){
        this.done = true;
    }

    @Override
    public String toString() {
        return "Assignment{" +
            "procedures=" + procedures +
            ", drugs=" + drugs +
            ", surgeries=" + surgeries +
            ", done=" + done +
            '}';
    }

    @Override
    public boolean equals(Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        Assignment that = (Assignment) o;
        return done == that.done && Objects.equals(procedures, that.procedures) &&
Objects.equals(drugs, that.drugs) && Objects.equals(surgeries, that.surgeries);
    }

    @Override
    public int hashCode() {
        return Objects.hash(procedures, drugs, surgeries, done);
    }
}

```

## Листинг выполнения подзадачи 2 (файл Hospital.java)

```
package task4;

import java.util.ArrayList;
import java.util.HashMap;

public class Hospital {
    private HashMap<Integer, Integer> client_doctor;
    private HashMap<Integer, Assignment> client_assign;
    private HashMap<Assignment, Integer> assign_staff;
    private ArrayList<Staff> staffArray;
    private ArrayList<Patient> patientArray;

    public Hospital() {
        this.client_doctor = new HashMap<>();
        this.client_assign = new HashMap<>();
        this.assign_staff = new HashMap<>();
        this.staffArray = new ArrayList<>();
        this.patientArray = new ArrayList<>();
    }

    public void addPatient(Patient patient){
        this.patientArray.add(patient);
    }

    public void addStaff(Staff staff){
        this.staffArray.add(staff);
    }

    public void setDoctor(int pat_id, int doc_id){
        this.client_doctor.put(pat_id, doc_id);
    }

    public void setAssignment(int pat_id, Assignment assignment){
        this.client_assign.put(pat_id, assignment);
    }

    public void completeAssignment(int pat_id, int staff_id){
        this.assign_staff.put(this.client_assign.get(pat_id), staff_id);
        this.client_assign.get(pat_id).complete();
    }

    public void dismissPatient(Patient patient, String reason){
        int pat_id = patient.getId();
        patient.dismiss(reason);
        this.client_doctor.remove(pat_id);
    }

    @Override
    public String toString() {
        return "Hospital{" + "\n" +
            "  ---client_doctor=" + client_doctor + ",\n" +
            "  ---client_assign=" + client_assign + ",\n" +
            "  ---assign_staff=" + assign_staff + ",\n" +
            "  ---staffArray=" + staffArray + ",\n" +
            "  ---patientArray=" + patientArray +
            '}' ;
    }
}
```

## Листинг выполнения подзадачи 2 (файл MainForHospital.java)

```
package task4;

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

        Hospital hospital = new Hospital();

        Patient pat1 = new Patient("Ivanov Petr Semenovich", 1);
        Patient pat2 = new Patient("Smirnov Ivan Andreevich", 2);
        Patient pat3 = new Patient("Semakov Andrey Ivanovich", 3);
```

```

Staff doc1 = new Staff(1, "Troshin Mikhail Dmitrievich", "Doctor");
Staff doc2 = new Staff(2, "Mikhailova Maria Maximovna", "Doctor");
Staff nurse = new Staff(3, "Zimina Natalia Andreevna", "Nurse");

hospital.addPatient(pat1);
hospital.addPatient(pat2);
hospital.addPatient(pat3);

hospital.addStaff(doc1);
hospital.addStaff(doc2);
hospital.addStaff(nurse);

hospital.setDoctor(pat1.getId(), doc1.getId());
hospital.setDoctor(pat2.getId(), doc2.getId());
hospital.setDoctor(pat3.getId(), doc2.getId());

System.out.println();
System.out.println("Doctors and patients: ");
System.out.println(hospital);

Assignment a1 = new Assignment();
a1.addDrug("Nurofen");
a1.addDrug("Sinupret");
a1.addDrug("Coldrex");
a1.addDrug("Tantum Verde");
a1.addProcedure("Inhalation");
hospital.setAssignment(pat1.getId(), a1);

Assignment a2 = new Assignment();
a2.addSurgery("Laser birthmark removal");
a2.addDrug("Panthenol");
hospital.setAssignment(pat2.getId(), a2);

Assignment a3 = new Assignment();
a3.addProcedure("Back massage");
a3.addProcedure("Swimming pool");
a3.addDrug("Vitamin D");
hospital.setAssignment(pat3.getId(), a3);

System.out.println();
System.out.println("Patients with assignments: ");
System.out.println(hospital);

hospital.completeAssignment(pat1.getId(), doc2.getId());
hospital.completeAssignment(pat2.getId(), nurse.getId());
hospital.completeAssignment(pat3.getId(), nurse.getId());

System.out.println();
System.out.println("Patients with done assignments: ");
System.out.println(hospital);

hospital.dismissPatient(pat1, "End of treatment");
hospital.dismissPatient(pat2, "End of treatment");
hospital.dismissPatient(pat3, "Transferred to another department for treatment");

System.out.println();
System.out.println("Patients are dismissed: ");
System.out.println(hospital);
}

```

```
Run MainForHospital
C:\Users\polin\jdk\openjdk-17.0.2\bin\java.exe -javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.2\lib\idea_rt.jar=57286:C:\Program Files\JetBrains\IntelliJ IDEA 2021.3.2\bin" -Dfile.encoding=UTF-8 -classpath C:\Users\polin\jdk\openjdk-17.0.2\bin\java.exe

Doctors and patients:
Hospital{
  ---client_doctor={1=1, 2=2, 3=2},
  ---client_assign={},
  ---assign_staff={},
  ---staffArray={Staff{id=1, name='Troshin Mikhail Dmitrievich', position='Doctor'}, Staff{id=2, name='Mikhailova Maria Maximovna', position='Doctor'}, Staff{id=3, name='Zimina Natalia Andreevna', position='Nurse'}},
  ---patientArray={Patient{name='Ivanov Petr Semenovich', id=1, inHospital=true, reason=''}, Patient{name='Smirnov Ivan Andreevich', id=2, inHospital=true, reason=''}, Patient{name='Semakov Andrey Ivanovich', id=3, inHospital=true, reason=''}}

Patients with assignments:
Hospital{
  ---client_doctor={1=1, 2=2, 3=2},
  ---client_assign={1=Assignment{procedures=[Inhalation], drugs=[Nurofen, Sinupret, Coldrex, Tantum Verde], surgeries=[], done=false}, 2=Assignment{procedures=[], drugs=[Panthenol], surgeries=[Laser birthmark removal], done=false},
  ---assign_staff={},
  ---staffArray={Staff{id=1, name='Troshin Mikhail Dmitrievich', position='Doctor'}, Staff{id=2, name='Mikhailova Maria Maximovna', position='Doctor'}, Staff{id=3, name='Zimina Natalia Andreevna', position='Nurse'}},
  ---patientArray={Patient{name='Ivanov Petr Semenovich', id=1, inHospital=true, reason=''}, Patient{name='Smirnov Ivan Andreevich', id=2, inHospital=true, reason=''}, Patient{name='Semakov Andrey Ivanovich', id=3, inHospital=true, reason=''}}

Patients with done assignments:
Hospital{
  ---client_doctor={1=1, 2=2, 3=2},
  ---client_assign={1=Assignment{procedures=[Inhalation], drugs=[Nurofen, Sinupret, Coldrex, Tantum Verde], surgeries=[], done=true}, 2=Assignment{procedures=[], drugs=[Panthenol], surgeries=[Laser birthmark removal], done=true},
  ---assign_staff={Assignment{procedures=[], drugs=[Panthenol], surgeries=[Laser birthmark removal], done=true}=3, Assignment{procedures=[Back massage, Swimming pool], drugs=[Vitamin D], surgeries=[], done=true}=3, Assignment{procedures=[Laser birthmark removal], done=true}=3},
  ---staffArray={Staff{id=1, name='Troshin Mikhail Dmitrievich', position='Doctor'}, Staff{id=2, name='Mikhailova Maria Maximovna', position='Doctor'}, Staff{id=3, name='Zimina Natalia Andreevna', position='Nurse'}},
  ---patientArray={Patient{name='Ivanov Petr Semenovich', id=1, inHospital=true, reason=''}, Patient{name='Smirnov Ivan Andreevich', id=2, inHospital=true, reason=''}, Patient{name='Semakov Andrey Ivanovich', id=3, inHospital=true, reason=''}}

Patients are dismissed:
Hospital{
  ---client_doctor={},
  ---client_assign={},
  ---client_assign={1=Assignment{procedures=[Inhalation], drugs=[Nurofen, Sinupret, Coldrex, Tantum Verde], surgeries=[], done=true}, 2=Assignment{procedures=[], drugs=[Panthenol], surgeries=[Laser birthmark removal], done=true},
  ---assign_staff={Assignment{procedures=[], drugs=[Panthenol], surgeries=[Laser birthmark removal], done=true}=3, Assignment{procedures=[Back massage, Swimming pool], drugs=[Vitamin D], surgeries=[], done=true}=3, Assignment{procedures=[Laser birthmark removal], done=true}=3},
  ---staffArray={Staff{id=1, name='Troshin Mikhail Dmitrievich', position='Doctor'}, Staff{id=2, name='Mikhailova Maria Maximovna', position='Doctor'}, Staff{id=3, name='Zimina Natalia Andreevna', position='Nurse'}},
  ---patientArray={Patient{name='Ivanov Petr Semenovich', id=1, inHospital=false, reason='End of treatment'}, Patient{name='Smirnov Ivan Andreevich', id=2, inHospital=false, reason='End of treatment'}, Patient{name='Semakov Andrey Ivanovich', id=3, inHospital=false, reason='End of treatment'}}

Process finished with exit code 0
```

Рисунок 9 - Результат выполнения кода решения подзадачи 2

## Ссылка на программное решение:

Программное решение представлено в репозитории распределённой системы управления версиями Git:

<https://github.com/Owlfeather/JavaMagisterCourse/tree/main/Lab3/src>

## Вывод:

При выполнении лабораторной работы были получены навыки работы с классами Java, были исследованы механизмы наследования и полиморфизма языка программирования Java.