



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

О Т Ч Е Т

по лабораторной работе № 3

Название: Классы, наследование, полиморфизм

Дисциплина: Языки программирования для работы с большими
данными

Студент

ИУ6-23М

(Группа)

(Подпись, дата)

А.А. Аветисян

(И.О. Фамилия)

Преподаватель

(Подпись, дата)

(И.О. Фамилия)

Москва, 2022

Лабораторная работа № 3

Задание:

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.

Ход работы:

Код программы:

```
public class Vector{
    private int n;
    private ArrayList<Double> data;

    public Vector(){
        this.n = 0;
        data = new ArrayList<>();
    }

    public Vector(int n){
        this.n = n;
        data = new ArrayList<>(n);
    }

    public Vector(double... data){
        n = data.length;
        this.data = new ArrayList<>(n);
        for (int i = 0; i < n; i++){
            this.data.add(data[i]);
        }
    }

    public int getN() {
        return n;
    }

    public void setN(int n) {
        this.n = n;
    }

    public ArrayList<Double> getData() {
        return data;
    }

    public void setData(ArrayList<Double> data) {
        this.data = data;
    }

    @Override
    public String toString() {
        return "Vector {" + data + "}";
    }
}
```

```

    }

    public double modulus(){ //модуль
        double mod = 0;
        for (double d : this.data){
            mod += d*d;
        }
        return mod;
    }

    public double scalar(Vector v){ //скалярное произведение
        if (n != v.data.size()){
            System.out.println("Ошибочный размер вектора");
        }
        double sc = 0;
        for (int i = 0; i < n; i++){
            sc += this.data.get(i) * v.data.get(i);
        }
        return sc;
    }

    public Vector sum(Vector v){ // сумма
        if (n != v.data.size()){
            System.out.println("Ошибочный размер вектора");
        }
        Vector v_output = new Vector(n);
        for (int i = 0; i < n; i++){
            v_output.data.add(this.data.get(i) + v.data.get(i));
        }
        return v_output;
    }

    public Vector diff(Vector v){ // разность
        if (n != v.data.size()){
            System.out.println("Ошибочный размер вектора");
        }
        Vector v_output = new Vector(n);
        for (int i = 0; i < n; i++){
            v_output.data.add(this.data.get(i) - v.data.get(i));
        }
        return v_output;
    }

    public Vector multi(int c){ // умножение на константу
        Vector v_output = new Vector(this.data.size());
        for (int i = 0; i < n; i++){
            v_output.data.add(this.data.get(i) * c);
        }
        return v_output;
    }
}

```

```

public static void isCollinear_Orthogonal(Vector v1, Vector v2){
    if (v1.getN() != v2.getN()){
        System.out.println("Ошибка размерности");
    }
    boolean isCollinear = true;
    ArrayList<Double> ratio = new ArrayList<>();
    for (int i = 0; i < v1.getN(); i++){
        if ((v1.getData().get(i) != 0 && v2.getData().get(i) == 0) ||
            (v2.getData().get(i) == 0 && v1.getData().get(i) != 0)){
            isCollinear = false;
            break;
        }
        if (v1.getData().get(i) == 0 && v2.getData().get(i) == 0){
            break;
        }
        ratio.add(v1.getData().get(i)/v2.getData().get(i));
    }
    for (int i = 0; i < v1.getN() - 1; i++){
        if (!Objects.equals(ratio.get(i),ratio.get(i + 1))){
            isCollinear = false;
            break;
        }
    }
    if (isCollinear){
        System.out.println("Вектора коллинеарны");
    } else if (v1.scalar(v2) == 0){
        System.out.println("Вектора ортогональны");
    } else {
        System.out.println("Вектора не коллинеарны и не ортогональны");
    }
}

```

```

public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int number, demention;
    System.out.print("Введите количество векторов: ");
    if (scanner.hasNextInt()) {
        number = scanner.nextInt();
    } else {
        System.out.println("error");
        number = -1;
    }
}

```

```

System.out.print("Введите размерность: ");
if (scanner.hasNextInt()) {
    demention = scanner.nextInt();
} else {
    System.out.println("error");
    demention = -1;
}

```

```

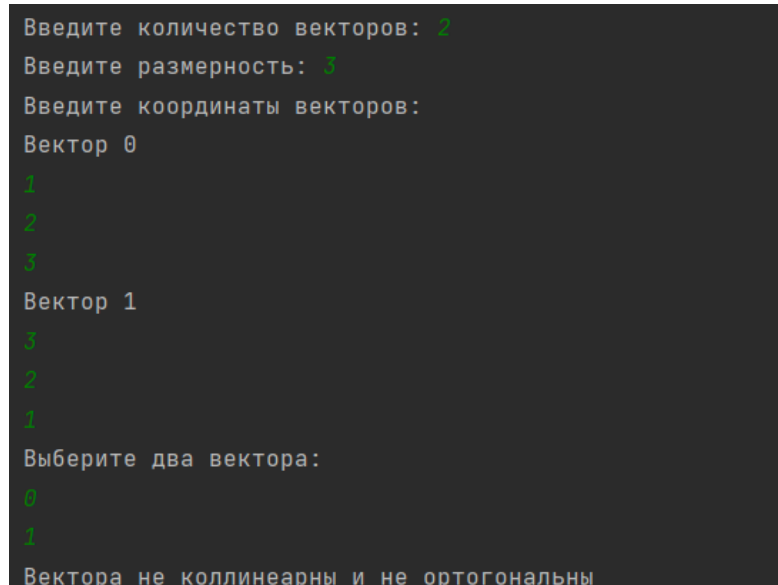
Vector[] vector_array = new Vector[number];
Vector first, second;
double[] coords = new double[demention];
System.out.println("Введите координаты векторов: ");
for (int i = 0; i < number; i++){
    System.out.println("Вектор " + i);
    for (int j = 0; j < 3; j++) {
        if (scanner.hasNextDouble()) {
            coords[j] = scanner.nextDouble();
        }
    }
}

```

```

    } else {
        System.out.println("error");
    }
}
vector_array[i] = new Vector(coords);
}
System.out.println("Выберите два вектора:");
if (scanner.hasNextInt()) {
    first = vector_array[scanner.nextInt()];
} else {
    first = new Vector(dimension);
    System.out.println("error");
}
if (scanner.hasNextInt()) {
    second = vector_array[scanner.nextInt()];
} else {
    second = new Vector(dimension);
    System.out.println("error");
}
isCollinear_Orthogonal(first, second);
}

```



```

Введите количество векторов: 2
Введите размерность: 3
Введите координаты векторов:
Вектор 0
1
2
3
Вектор 1
3
2
1
Выберите два вектора:
0
1
Вектора не коллинеарны и не ортогональны

```

Рисунок 1 – Результат работы программы

Задание:

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.

Ход работы:

Код программы:

```

public class VectorR3 {

    private double x1;

```

```

private double x2;
private double y1;
private double y2;
private double z1;
private double z2;

private double x;
private double y;
private double z;

public VectorR3(){
}

public VectorR3(double x1, double x2, double y1, double y2, double z1, double z2){
    this.x1 = x1;
    this.x2 = x2;
    this.y1 = y1;
    this.y2 = y2;
    this.z1 = z1;
    this.z2 = z2;

    this.x = x2 - x1;
    this.y = y2 - x1;
    this.z = z2 - z1;
}

public double getX() {
    return x;
}

public double getY() {
    return y;
}

public double getZ() {
    return z;
}

public double mod(){
    return Math.sqrt(x*x + y*y + z*z);
}

public double scalar(VectorR3 v){ //скалярное произведение
    return this.x * v.x + this.y * v.y + this.z * v.z;
}

public boolean is_Orthogonal(VectorR3 v){
    return this.scalar(v) == 0;
}

public boolean is_Intersect(VectorR3 v){
    // Система уравнений:
    // this.x1 * t + this.x2 * (1 - t) = v.x1 * s + v.x2 * (1 - s)
    // this.y1 * t + this.y2 * (1 - t) = v.y1 * s + v.y2 * (1 - s)
    // Находим t, s, подставляем в:
    // this.z1 * t + this.z2 * (1 - t) = v.z1 * s + v.z2 * (1 - s)
    // Если получилось равенство, то отрезки (вектора) пересекаются

    double s = ((this.x2 - v.x2) * this.y + (v.y2 - this.y2) * this.x) / (this.x * v.y - this.y * v.x);

```

```

        double t = (s * v.x + this.x2 - v.x2) / this.x;
        double eq1 = this.z1 * t + this.z2 * (1 - t);
        double eq2 = v.z1 * s + v.z2 * (1 - s);

        return eq1 == eq2;
    }

    public void compare(VectorR3 v){
        if (this.mod() > v.mod()){
            System.out.println("Первый больше");
        } else if (v.mod() > this.mod()){
            System.out.println("Второй больше");
        } else {
            System.out.println("Равны");
        }
    }
}

@Override
public String toString() {
    return "VectorR3: " +
        "x1 = " + x1 +
        ", x2 = " + x2 +
        ", y1 = " + y1 +
        ", y2 = " + y2 +
        ", z1 = " + z1 +
        ", z2 = " + z2 +
        '\n';
}

}

public static void isCoplanar(VectorR3 v1, VectorR3 v2, VectorR3 v3){
    double m = v1.getX() * v2.getY() * v3.getZ() + v1.getY() * v2.getZ() * v3.getX()
        + v1.getZ() * v2.getX() * v3.getY() - v1.getZ() * v2.getY() * v3.getX()
        - v1.getX() * v2.getZ() * v3.getY() - v1.getY() * v2.getX() * v3.getZ();
    if (m == 0){
        System.out.println("Вектора компланарны");
    } else {
        System.out.println("Вектора не компланарны");
    }
}

}

public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int number;
    System.out.print("Введите количество векторов: ");
    if (scanner.hasNextInt()) {
        number = scanner.nextInt();
    } else {
        System.out.println("error");
        number = -1;
    }
    VectorR3[] vector_array = new VectorR3[number];
    Random random = new Random();
    double x1, x2, y1, y2, z1, z2;
    System.out.println("Введите координаты векторов: ");
    for (int i = 0; i < number; i++){
        x1 = random.nextDouble();
        x2 = random.nextFloat();

```

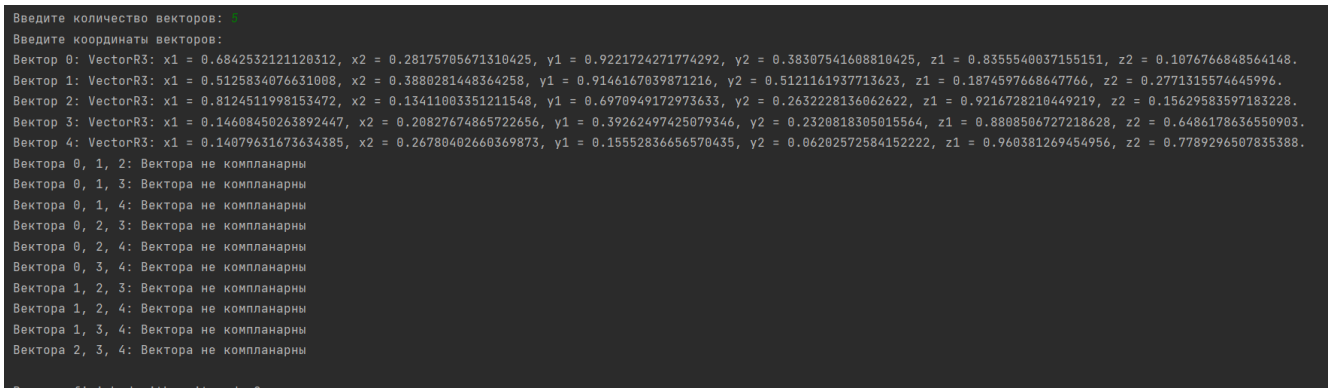
```

        y1 = random.nextFloat();
        y2 = random.nextFloat();
        z1 = random.nextFloat();
        z2 = random.nextFloat();
        vector_array[i] = new VectorR3(x1, x2, y1, y2, z1, z2);
    }

    for (int i = 0; i < number; i++){
        System.out.println("Вектор " + i + ": " + vector_array[i].toString());
    }

    for (int i = 0; i < number - 2; i++){
        for (int j = i + 1; j < number - 1; j++){
            for (int k = j + 1; k < number; k++){
                System.out.print("Вектора " + i + ", " + j + ", " + k + ": ");
                isCoplanar(vector_array[i], vector_array[j], vector_array[k]);
            }
        }
    }
}

```



```

Введите количество векторов: 5
Введите координаты векторов:
Вектор 0: VectorR3: x1 = 0.6842532121120312, x2 = 0.28175705671310425, y1 = 0.9221724271774292, y2 = 0.38307541608810425, z1 = 0.8355540037155151, z2 = 0.1076766848564148.
Вектор 1: VectorR3: x1 = 0.5125834076631008, x2 = 0.3880281448364258, y1 = 0.9146167039871216, y2 = 0.5121161937713623, z1 = 0.1874597668647766, z2 = 0.2771315574645996.
Вектор 2: VectorR3: x1 = 0.8124511998153472, x2 = 0.13411003351211548, y1 = 0.6970949172973633, y2 = 0.2632228136062622, z1 = 0.9216728210449219, z2 = 0.15629583597183228.
Вектор 3: VectorR3: x1 = 0.14608450263892447, x2 = 0.20827674865722656, y1 = 0.39262497425079346, y2 = 0.2320818305015564, z1 = 0.8808506727218628, z2 = 0.6486178636550903.
Вектор 4: VectorR3: x1 = 0.14079631673634385, x2 = 0.26780402660369873, y1 = 0.15552836656570435, y2 = 0.06202572584152222, z1 = 0.960381269454956, z2 = 0.7789296507835388.
Вектора 0, 1, 2: Вектора не компланарны
Вектора 0, 1, 3: Вектора не компланарны
Вектора 0, 1, 4: Вектора не компланарны
Вектора 0, 2, 3: Вектора не компланарны
Вектора 0, 2, 4: Вектора не компланарны
Вектора 0, 3, 4: Вектора не компланарны
Вектора 1, 2, 3: Вектора не компланарны
Вектора 1, 2, 4: Вектора не компланарны
Вектора 1, 3, 4: Вектора не компланарны
Вектора 2, 3, 4: Вектора не компланарны

```

Рисунок 2 – Результат работы программы

Задание:

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.

Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: а) список покупателей в алфавитном порядке; б) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

Ход работы:

Код программы:

```
public class Customer {
```



```

private int id;
private String surname;
private String name;
private String lastname;
private long credit_card;
private long bank_acc;

public Customer(){
}

public Customer(int id, String surname, String name, String lastname, long credit_card, long bank_acc){
    this.id = id;
    this.surname = surname;
    this.name = name;
    this.lastname = lastname;
    this.credit_card = credit_card;
    this.bank_acc = bank_acc;
}

public int getId() {
    return id;
}

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

public String getSurname() {
    return surname;
}

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

public String getName() {
    return name;
}

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

public String getLastname() {
    return lastname;
}

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

public long getCredit_card() {
    return credit_card;
}

public void setCredit_card(long credit_card) {
    this.credit_card = credit_card;
}

public long getBank_acc() {

```

```

        return bank_acc;
    }

    public void setBank_acc(long bank_acc) {
        this.bank_acc = bank_acc;
    }

    @Override
    public String toString() {
        return "Customer{" +
            "id=" + id +
            ", surname=" + surname + "\" +
            ", name=" + name + "\" +
            ", lastname=" + lastname + "\" +
            ", credit_card=" + credit_card +
            ", bank_acc=" + bank_acc +
            '>';
    }
}

public static void sorting(ArrayList<Customer> customers){
    System.out.println("Сортировка:");
    Comparator<Customer> CustomersComparator
        = Comparator.comparing(Customer::getSurname);
    customers.sort(CustomersComparator);
    for (Customer customer:customers){
        System.out.println(customer);
    }
}

public static void credit_select(ArrayList<Customer> customers, long credit_start, long credit_end){
    System.out.println("Карты из диапазона:");
    for (Customer customer:customers){
        if ((customer.getCredit_card() > credit_start) && (customer.getCredit_card() < credit_end)){
            System.out.println(customer);
        }
    }
}

public static void main(String[] args){
    ArrayList<Customer> customers = new ArrayList<>();
    Scanner scanner = new Scanner(System.in);
    int number;
    System.out.print("Введите количество покупателей: ");
    if (scanner.hasNextInt()) {
        number = scanner.nextInt();
    } else {
        System.out.println("error");
        number = -1;
    }

    for (int i = 0; i < number; i++){
        Customer customer = new Customer();
        customer.setId(i + 1);
        System.out.println("Покупатель " + i + ":");
        System.out.print("Введите фамилию: ");
        if (scanner.hasNext()) {
            customer.setSurname(scanner.next());
        } else {

```

```

        System.out.println("error");
    }
    System.out.print("Введите имя: ");
    if (scanner.hasNext()) {
        customer.setName(scanner.next());
    } else {
        System.out.println("error");
    }
    System.out.print("Введите отчество: ");
    if (scanner.hasNext()) {
        customer.setLastname(scanner.next());
    } else {
        System.out.println("error");
    }
    System.out.print("Введите номер карты: ");
    if (scanner.hasNextLong()) {
        customer.setCredit_card(scanner.nextLong());
    } else {
        System.out.println("error");
    }
    System.out.print("Введите номер счета: ");
    if (scanner.hasNextLong()) {
        customer.setBank_acc(scanner.nextLong());
    } else {
        System.out.println("error");
    }
    customers.add(customer);
}
sorting(customers);
credit_select(customers, 0, 110000000 );

}

```

```

Введите количество покупателей: 3
Покупатель 0:
Введите фамилию: Аветисян
Введите имя: Арсений
Введите отчество: Андреевич
Введите номер карты: 12345
Введите номер счета: 111111111
Покупатель 1:
Введите фамилию: Петров
Введите имя: Василий
Введите отчество: Петрович
Введите номер карты: 111111111
Введите номер счета: 111111111
Покупатель 2:
Введите фамилию: Аннушкин
Введите имя: Антон
Введите отчество: Антонович
Введите номер карты: 1111111
Введите номер счета: 111111111
Сортировка:
Customer{id=1, surname='Аветисян', name='Арсений', lastname='Андреевич', credit_card=12345, bank_acc=111111111}
Customer{id=3, surname='Аннушкин', name='Антон', lastname='Антонович', credit_card=1111111, bank_acc=111111111}
Customer{id=2, surname='Петров', name='Василий', lastname='Петрович', credit_card=111111111, bank_acc=111111111}
Карты из диапазона:
Customer{id=1, surname='Аветисян', name='Арсений', lastname='Андреевич', credit_card=12345, bank_acc=111111111}
Customer{id=3, surname='Аннушкин', name='Антон', lastname='Антонович', credit_card=1111111, bank_acc=111111111}

```

Рисунок 3 – Результат работы программы

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.

Ход работы:

Код программы:

```
public class Patient {
    private int id;
    private String surname;
    private String name;
    private String lastname;
    private String address;
    private String phone;
    private long card_number;
    private String diagnosis;

    public Patient(){
    }

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

    public int getId() {
        return id;
    }

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

    public String getSurname() {
        return surname;
    }

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

    public String getName() {
        return name;
    }

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

```

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 long getCard_number() {
    return card_number;
}

public void setCard_number(long card_number) {
    this.card_number = card_number;
}

public String getDiagnosis() {
    return diagnosis;
}

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

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

```

```

public static void same_diagnosis(ArrayList<Patient> patients, String diagnosis){
    System.out.println("Пациенты с диагнозом: " + diagnosis);
    for (Patient patient:patients){
        if (patient.getDiagnosis().equals(diagnosis)){
            System.out.println(patient);
        }
    }
}

public static void card_select(ArrayList<Patient> patients, long card_start, long card_end){
    System.out.println("Карты из диапазона:");
    for (Patient patient:patients){
        if ((patient.getCard_number() > card_start) && (patient.getCard_number() < card_end)){
            System.out.println(patient);
        }
    }
}

public static void main(String[] args){
    ArrayList<Patient> patients = new ArrayList<>();
    Scanner scanner = new Scanner(System.in);
    int number;
    System.out.print("Введите количество пациентов: ");
    if (scanner.hasNextInt()) {
        number = scanner.nextInt();
    } else {
        System.out.println("error");
        number = -1;
    }

    for (int i = 0; i < number; i++){
        Patient patient = new Patient();
        patient.setId(i + 1);
        System.out.println("Покупатель " + i + ":" );
        System.out.print("Введите фамилию: ");
        if (scanner.hasNext()) {
            patient.setSurname(scanner.next());
        } else {
            System.out.println("error");
        }
        System.out.print("Введите имя: ");
        if (scanner.hasNext()) {
            patient.setName(scanner.next());
        } else {
            System.out.println("error");
        }
        System.out.print("Введите отчество: ");
        if (scanner.hasNext()) {
            patient.setLastname(scanner.next());
        } else {
            System.out.println("error");
        }
        System.out.print("Введите адрес: ");
        if (scanner.hasNextLine()) {
            patient.setAddress(scanner.next());
        } else {
            System.out.println("error");
        }
    }
}

```

```

scanner.next();
System.out.print("Введите телефон: ");
if (scanner.hasNext()) {
    patient.setPhone(scanner.next());
} else {
    System.out.println("error");
}
System.out.print("Введите номер карты: ");
if (scanner.hasNextLong()) {
    patient.setCard_number(scanner.nextLong());
} else {
    System.out.println("error");
}
System.out.print("Введите диагноз: ");
if (scanner.hasNext()) {
    patient.setDiagnosis(scanner.next());
} else {
    System.out.println("error");
}

patients.add(patient);
}
same_diagnosis(patients, "ОРВИ");
card_select(patients, 0,110000000 );
}

```

```

Введите количество пациентов: 2
Покупатель 0:
Введите фамилию: Аветисян
Введите имя: Арсений
Введите отчество: Андреевич
Введите адрес: Гончарная ул.
Введите телефон: 89651577685
Введите номер карты: 12345
Введите диагноз: ОРЗ
Покупатель 1:
Введите фамилию: Петров
Введите имя: Петр
Введите отчество: Петрович
Введите адрес: Петрова ул.
Введите телефон: 89766666666
Введите номер карты: 1111111111
Введите диагноз: ОРВИ
Пациенты с диагнозом: ОРВИ
Patient{id=2, surname='Петров', name='Петр', lastname='Петрович', address='Петрова', phone='89766666666', card_number=1111111111, diagnosis='ОРВИ'}
Карты из диапазона:
Patient{id=1, surname='Аветисян', name='Арсений', lastname='Андреевич', address='Гончарная', phone='89651577685', card_number=12345, diagnosis='ОРЗ'}

```

Рисунок 4 – Результат работы программы

Задание:

Создать приложение, удовлетворяющее требованиям, приведенным в задании. Аргументировать принадлежность классу каждого создаваемого метода и корректно переопределить для каждого класса методы equals(), hashCode(), toString(). Создать объект класса Одномерный массив, используя класс Массив. Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).

Ход работы:

Код программы:

```
public abstract class Array {

    public abstract void print();
    public abstract Object sum(Object obj);
    public abstract Object diff(Object obj);
    public abstract Object multi(Object obj);

}

public class One_Dim_Array extends Array{

    private ArrayList<Integer> data;

    public One_Dim_Array(){

    }

    public One_Dim_Array(int n){
        this.data = new ArrayList<>(n);
    }

    public One_Dim_Array(ArrayList<Integer> data){
        this.data = data;
    }

    @Override
    public void print() {
        System.out.println("Массив: " + this);
    }

    @Override
    public One_Dim_Array sum(Object obj) {
        One_Dim_Array array = new One_Dim_Array(this.data.size());
        if (obj.getClass().equals(this.getClass())){
            if (this.data.size() != ((One_Dim_Array) obj).data.size()){
                System.out.println("ошибка, несоответствие размеров");
            } else {
                for (int i = 0; i < this.data.size(); i++) {
                    array.data.add(this.data.get(i) + ((One_Dim_Array) obj).data.get(i));
                }
            }
        } else {
            System.out.println("ошибка, несоответствие классов");
        }
        return array;
    }

    @Override
    public One_Dim_Array multi(Object obj) {
        One_Dim_Array array = new One_Dim_Array(this.data.size());
        if (obj.getClass().equals(this.getClass())){
            if (this.data.size() != ((One_Dim_Array) obj).data.size()){
                System.out.println("ошибка, несоответствие размеров");
            } else {
                for (int i = 0; i < this.data.size(); i++) {
                    array.data.add(this.data.get(i)*((One_Dim_Array) obj).data.get(i));
                }
            }
        }
    }
}
```



```

        }
    }
    } else {
        System.out.println("ошибка, несоответствие классов");
    }
    return array;
}

@Override
public One_Dim_Array diff(Object obj) {
    One_Dim_Array array = new One_Dim_Array(this.data.size());
    if (obj.getClass().equals(this.getClass())){
        if (this.data.size() != ((One_Dim_Array) obj).data.size()){
            System.out.println("ошибка, несоответствие размеров");
        } else {
            for (int i = 0; i < this.data.size(); i++) {
                array.data.add(this.data.get(i) - ((One_Dim_Array) obj).data.get(i));
            }
        }
    } else {
        System.out.println("ошибка, несоответствие классов");
    }
    return array;
}

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

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

@Override
public String toString() {
    return "One_Dim_Array{" +
        "data=" + data +
        '}';
}

}

public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int number;
    System.out.print("Введите количество элементов: ");
    if (scanner.hasNextInt()) {
        number = scanner.nextInt();
    } else {
        System.out.println("error");
        number = -1;
    }

    ArrayList<Integer> input_data = new ArrayList<>(number);
    System.out.print("Введите элементы: ");
    for (int i = 0; i < number; i++){

```

```

        if (scanner.hasNextInt()) {
            input_data.add(scanner.nextInt());
        } else {
            System.out.println("error");
        }
    }
}
One_Dim_Array array = new One_Dim_Array(input_data);

ArrayList<Integer> input_data_second = new ArrayList<>(number);
System.out.print("Введите второй массив: ");
for (int i = 0; i < number; i++){
    if (scanner.hasNextInt()) {
        input_data_second.add(scanner.nextInt());
    } else {
        System.out.println("error");
    }
}
One_Dim_Array second_array = new One_Dim_Array(input_data_second);
System.out.println(array + " " + second_array);

System.out.println("Сумма: " + array.sum(second_array));
System.out.println("Разность: " + array.diff(second_array));
System.out.println("Произведение: " + array.multi(second_array));
}

```

```

Введите количество элементов: 3
Введите элементы: 1
2
3
Введите второй массив: 4
5
6
One_Dim_Array{data=[1, 2, 3]} One_Dim_Array{data=[4, 5, 6]}
Сумма: One_Dim_Array{data=[5, 7, 9]}
Разность: One_Dim_Array{data=[-3, -3, -3]}
Произведение: One_Dim_Array{data=[4, 10, 18]}

```

Рисунок 5 – Результат работы программы

Задание:

Создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

Ход работы:

Код программы:

```
public class Number {

    private final int number;

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

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

    public int getNumber() {
        return number;
    }

    public void print(){
        System.out.println(this);
    }

    public Object sum(Object obj){
        if (obj.getClass().equals(this.getClass())){
            Number number = new Number(((Number) obj).number + this.number);
        } else {
            System.out.println("ошибка, несоответствие классов");
        }
        return number;
    }

    public Object diff(Object obj){
        if (obj.getClass().equals(this.getClass())){
            Number number = new Number(this.number - ((Number) obj).number);
        } else {
            System.out.println("ошибка, несоответствие классов");
        }
        return number;
    }

    public Object multi(Object obj){
        if (obj.getClass().equals(this.getClass())){
            Number number = new Number(((Number) obj).number * this.number);
        } else {
            System.out.println("ошибка, несоответствие классов");
        }
        return number;
    }

    public Object div(Object obj){
        if (obj.getClass().equals(this.getClass())){
            Number number = new Number(this.number/((Number) obj).number);
        } else {
```

```

        System.out.println("ошибка, несоответствие классов");
    }
    return 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 class Simple_Fraction extends Number{

    private int numerator;
    private int denominator;

    public Simple_Fraction(){
    }

    public Simple_Fraction(int numerator, int denominator){
        this.numerator = numerator;
        this.denominator = denominator;
    }

    @Override
    public void print() {
        System.out.println(this);
    }

    @Override
    public Object sum(Object obj) {
        Simple_Fraction simple_fraction;
        if (obj.getClass().equals(this.getClass())) {
            simple_fraction = new Simple_Fraction(((Simple_Fraction) obj).numerator * this.denominator
                + this.numerator * ((Simple_Fraction) obj).denominator,
                ((Simple_Fraction) obj).denominator * this.denominator);
        } else {
            System.out.println("ошибка, несоответствие классов");
            simple_fraction = new Simple_Fraction();
        }
        return simple_fraction;
    }

    @Override
    public Object diff(Object obj) {

```

```

Simple_Fraction simple_fraction;
if (obj.getClass().equals(this.getClass())) {
    simple_fraction = new Simple_Fraction(this.numerator * ((Simple_Fraction) obj).denominator
        - ((Simple_Fraction) obj).numerator * this.denominator,
        ((Simple_Fraction) obj).denominator * this.denominator);
} else {
    System.out.println("ошибка, несоответствие классов");
    simple_fraction = new Simple_Fraction();
}
return simple_fraction;
}

@Override
public Object multi(Object obj) {
    Simple_Fraction simple_fraction;
    if (obj.getClass().equals(this.getClass())) {
        simple_fraction = new Simple_Fraction(this.numerator * ((Simple_Fraction) obj).numerator,
            ((Simple_Fraction) obj).denominator * this.denominator);
    } else {
        System.out.println("ошибка, несоответствие классов");
        simple_fraction = new Simple_Fraction();
    }
    return simple_fraction;
}

@Override
public Object div(Object obj) {
    Simple_Fraction simple_fraction;
    if (obj.getClass().equals(this.getClass())) {
        simple_fraction = new Simple_Fraction(this.numerator * ((Simple_Fraction) obj).denominator,
            ((Simple_Fraction) obj).numerator * this.denominator);
    } else {
        System.out.println("ошибка, несоответствие классов");
        simple_fraction = new Simple_Fraction();
    }
    return simple_fraction;
}

@Override
public boolean equals(Object o) {
    if (this == o) return true;
    if (o == null || getClass() != o.getClass()) return false;
    if (!super.equals(o)) return false;
    Simple_Fraction that = (Simple_Fraction) o;
    return numerator == that.numerator && denominator == that.denominator;
}

@Override
public String toString() {
    return "Simple_Fraction{" +
        "numerator=" + numerator +
        ", denominator=" + denominator +
        '}';
}

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

```

```

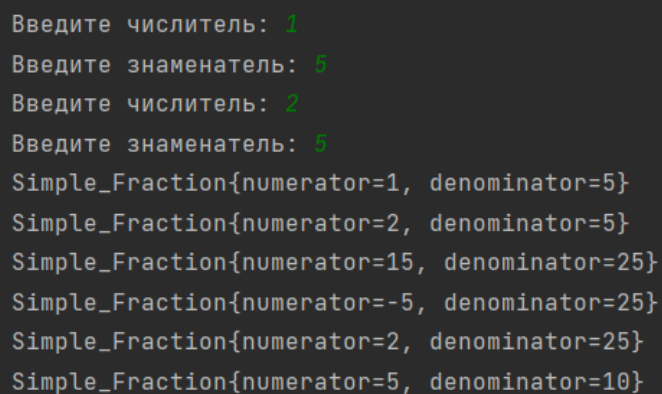
public static void main(String[] args){
    Scanner scanner = new Scanner(System.in);
    int numerator, denominator;
    System.out.print("Введите числитель: ");
    if (scanner.hasNextInt()) {
        numerator = scanner.nextInt();
    } else {
        System.out.println("error");
        numerator = -1;
    }
    System.out.print("Введите знаменатель: ");
    if (scanner.hasNextInt()) {
        denominator = scanner.nextInt();
    } else {
        System.out.println("error");
        denominator = -1;
    }
    Simple_Fraction first_fraction = new Simple_Fraction(numerator, denominator);

    System.out.print("Введите числитель: ");
    if (scanner.hasNextInt()) {
        numerator = scanner.nextInt();
    } else {
        System.out.println("error");
        numerator = -1;
    }
    System.out.print("Введите знаменатель: ");
    if (scanner.hasNextInt()) {
        denominator = scanner.nextInt();
    } else {
        System.out.println("error");
        denominator = -1;
    }
    Simple_Fraction second_fraction = new Simple_Fraction(numerator, denominator);

    first_fraction.print();
    second_fraction.print();

    ((Simple_Fraction) first_fraction.sum(second_fraction)).print();
    ((Simple_Fraction) first_fraction.diff(second_fraction)).print();
    ((Simple_Fraction) first_fraction.multi(second_fraction)).print();
    ((Simple_Fraction) first_fraction.div(second_fraction)).print();
}

```



```

Введите числитель: 1
Введите знаменатель: 5
Введите числитель: 2
Введите знаменатель: 5
Simple_Fraction{numerator=1, denominator=5}
Simple_Fraction{numerator=2, denominator=5}
Simple_Fraction{numerator=15, denominator=25}
Simple_Fraction{numerator=-5, denominator=25}
Simple_Fraction{numerator=2, denominator=25}
Simple_Fraction{numerator=5, denominator=10}

```

Рисунок 6 – Результат работы программы

Задание:

Построить модель программной системы. 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.

Ход работы:

Класс Client позволяет получить контактные данные клиента. Класс Account содержит информацию о счете и карте клиента, позволяет сделать платеж и заблокировать счет. Класс банк позволяет добавить множество аккаунтов, а также блокировать счет за превышение кредита. Отдельно создан класс Order для учета заказов, у клиента есть функция оплатить заказ.

Код программы:

```
package com.company.Lab3;

public class Client {

    private int client_id;
    private String name;

    public Client(){
    }

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

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

    public String getName() {
        return name;
    }

    public int getClient_Id() {
        return client_id;
    }

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

    @Override
    public String toString() {
        return "Client{" +
            "id=" + client_id +
```

```

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

```

```

package com.company.Lab3;

```

```

public class Account {
    private Client client;
    private int account_id;
    private int credit_balance;
    private int credit_limit;
    public boolean account_status;

    public Account(Client client, int credit_limit){
        this.client = client;
        this.account_id = client.getClient_Id();
        this.credit_limit = credit_limit;
        this.credit_balance = 0;
        this.account_status = true;
    }

    public int getCredit_balance() {
        return credit_balance;
    }

    public int getCredit_limit() {
        return credit_limit;
    }

    public int getAccount_id() {
        return account_id;
    }

    public Client getClient() {
        return client;
    }

    public void disableAccount(){
        this.account_status = false;
    }

    public boolean checkAccount(){
        if (!this.account_status){
            System.out.println("Аккаунт аннулирован");
            return false;
        } else {
            return true;
        }
    }

    public void change(int sum){
        if (checkAccount()){
            this.credit_balance += sum;
        }
    }
}

```



```

public void send(Account account, int sum){
    if (checkAccount()){
        this.change(-sum);
        account.change(sum);
    }
}

public void payOrder(Order order){
    if (order.getClient_ID() == this.client.getClient_Id()){
        this.change(-order.getSum());
        order.setStatus();
    } else {
        System.out.println("Заказ не принадлежит клиенту");
    }
}

@Override
public String toString() {
    return "Account{" +
        "client=" + client +
        ", account_id=" + account_id +
        ", credit_balance=" + credit_balance +
        ", account_status=" + account_status +
        '}';
}
}

```

```
import java.util.HashMap;
```

```

public class Bank {

    private HashMap<Integer, Account> accountHashMap;

    public Bank(){
        this.accountHashMap = new HashMap<>();
    }

    public void add_Account(int id, Account account) {
        accountHashMap.put(id, account);
    }

    public void blockCredit(int id) {
        if (accountHashMap.get(id).getCredit_balance() < accountHashMap.get(id).getCredit_limit()){
            accountHashMap.get(id).disableAccount();
        } else {
            System.out.println("Нет прав закрыть аккаунт");
        }
    }
}

```

```
package com.company.Lab3;

public class Order {
    private Client client;
    private int order_id;
    private int sum;
    private boolean status;

    public Order(Client client, int id, int sum){
        this.client = client;
        this.order_id = id;
        this.sum = sum;
        this.status = true;
    }

    public int getClient_ID() {
        return client.getClient_Id();
    }

    public int getSum() {
        return sum;
    }

    public void setStatus() {
        this.status = true;
    }
}
```

Задание:

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.

Ход работы:

Созданы классы Patient, Staff, Doctor. Класс клиента хранит личные данные, диагноз, рекомендации врача и статус пациента. Доктор имеет несколько пациентов, может ставить диагноз и выписывать пациентов. Класс персонала наследуется от класса доктора, также имеет несколько клиентов и может исполнять назначения врача.

Код программы:

```
public class Patient_1 {
    private String name;
    private String diagnosis;
    private String rec;
    private String status;

    public Patient_1(String name){
        this.name = name;
        this.diagnosis = "-";
        this.rec = "-";
        this.status = "Поступил в больницу";
    }

    public String getDiagnosis() {
        return diagnosis;
    }

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

    public String getRec() {
        return rec;
    }

    public void setRec(String rec) {
        this.rec = rec;
    }

    public String getStatus() {
        return status;
    }

    public void setStatus(String status) {
```

```

        this.status = status;
    }
}

public class Doctor {
    private String name;
    private HashMap<Integer, Patient_1> patientHashMap;
    public Doctor(String name){
        this.name = name;
        patientHashMap = new HashMap<>();
    }
    public void addPatient(int id, Patient_1 patient){
        patientHashMap.put(id, patient);
    }
    public HashMap<Integer, Patient_1> getPatientHashMap() {
        return patientHashMap;
    }
    public void setDiagnosis(Patient_1 patient, String diagnosis, String rec) {
        if (patient.getDiagnosis() == "-"){
            System.out.println("Диагноз уже поставлен");
        } else {
            patient.setDiagnosis(diagnosis);
            patient.setRec(rec);
            patient.setStatus("Диагноз поставлен, назначено лечение");
        }
    }
    public void write_out(int id) {
        this.patientHashMap.get(id).setRec("Выписан");
    }
}

import java.util.Objects;
public class Staff extends Doctor{
    public Staff(String name) {
        super(name);
    }
    @Override
    public void addPatient(int id, Patient_1 patient) {
        super.addPatient(id, patient);
    }
    public boolean checkRec(int id){
        return !Objects.equals(super.getPatientHashMap().get(id).getRec(), "-");
    }
    public void doRec(int id){
        super.getPatientHashMap().get(id).setRec("Вылечен");
    }
}

```

Вывод: лабораторная работа выполнена в соответствии с заданием и вариантом.