Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ **ИНФОРМАТИКА, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ И СИСТЕМЫ УПРАВЛЕНИЯ**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

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

О Т Ч Е Т

по лабораторной работе №3

Название: Классы, наследование, полиморфизм
Дисциплина: Языки программирования для работы с большими данными

Вариант: 2

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

Вариант 1

2. 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.

Листинг 1 – Код программы

```
class Vector {
    private double[] elements;

    public Vector(double[] elements) {
        this.elements = elements;
    }

    public Vector(int dimension) {
        elements = new double[dimension];
    }

    public double modulus() {
        double sumOfSquares = 0.0;
        for (double element : elements) {
            sumOfSquares += element * element;
        }
        return Math.sqrt(sumOfSquares);
    }

    public double scalarProduct(Vector other) {
        double result = 0.0;
        for (int i = 0; i < elements.length; i++) {
            result += elements[i] * other.elements[i];
        }
        return result;
    }

    public Vector add(Vector other) {
        if (elements.length != other.elements.length) {
            throw new IllegalArgumentException("Vectors must have the same dimension");
        }
        double[] result = new double[elements.length];
        for (int i = 0; i < elements.length; i++) {
            result[i] = elements[i] + other.elements[i];
        }
        return new Vector(result);
    }

    public Vector subtract(Vector other) {
        if (elements.length != other.elements.length) {
            throw new IllegalArgumentException("Vectors must have the same dimension");
        }
        double[] result = new double[elements.length];
        for (int i = 0; i < elements.length; i++) {
            result[i] = elements[i] - other.elements[i];
        }
        return new Vector(result);
```

```
    }

    // Method for multiplying a vector by a constant
    public Vector multiply(double constant) {
        double[] result = new double[elements.length];
        for (int i = 0; i < elements.length; i++) {
            result[i] = elements[i] * constant;
        }
        return new Vector(result);
    }

    // Method for determining whether two vectors are collinear or orthogonal
    public String determineRelationship(Vector other) {
        if (elements.length != other.elements.length) {
            throw new IllegalArgumentException("Vectors must have the same dimension");
        }
        double scalarProduct = scalarProduct(other);
        if (scalarProduct == 0) {
            return "The vectors are orthogonal";
        } else if (scalarProduct == modulus() * other.modulus()) {
            return "The vectors are collinear";
        } else {
            return "The vectors are neither collinear nor orthogonal";
        }
    }
    public static void main(String[] args) {
        Vector[] vectors = new Vector[4];
        vectors[0] = new Vector(new double[] {-2.0, -2.0, 3.0});
        vectors[1] = new Vector(new double[] {4.0, 5.0, 6.0});
        vectors[2] = new Vector(new double[] {8.0, 8.0, 12.0});
        vectors[3] = new Vector(new double[] {4.0, 4.0, 6.0});

        Vector sum = vectors[0].add(vectors[1]);
        System.out.println("The sum of vectors[0] and vectors[1] is " + sum);

        Vector difference = vectors[1].subtract(vectors[2]);
        System.out.println("The difference of vectors[1] and vectors[2] is " + difference);

        Vector product = vectors[2].multiply(2.0);
        System.out.println("The product of vectors[2] and 2.0 is " + product);

        System.out.println("The modulus of vectors[0] is " + vectors[0].modulus());
        System.out.println("The scalar product of vectors[0] and vectors[1] is " + vectors[0].scalarProduct(vectors[1]));

        System.out.println("The relationship between vectors[0] and vectors[1] is " +
vectors[0].determineRelationship(vectors[1]));
        System.out.println("The relationship between vectors[1] and vectors[2] is " +
vectors[1].determineRelationship(vectors[2]));
        System.out.println("The relationship between vectors[0] and vectors[2] is " +
vectors[0].determineRelationship(vectors[2]));
        System.out.println("The relationship between vectors[0] and vectors[2] is " +
vectors[2].determineRelationship(vectors[3]));
    }

    // toString method for printing
    @Override
    public String toString() {
        StringBuilder stringBuilder = new StringBuilder("(");
```

```
        for (int i = 0; i < elements.length; i++) {
            stringBuilder.append(elements[i]);
            if (i != elements.length - 1) {
                stringBuilder.append(", ");
            }
        }
        stringBuilder.append(")");
        return stringBuilder.toString();
    }
}
```

```
C:\Program Files\Java\jdk-15.0.1\bin\java.exe "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA 2022
The sum of vectors[0] and vectors[1] is (2.0, 3.0, 9.0)
The difference of vectors[1] and vectors[2] is (-4.0, -3.0, -6.0)
The product of vectors[2] and 2.0 is (16.0, 16.0, 24.0)
The modulus of vectors[0] is 4.123105625617661
The scalar product of vectors[0] and vectors[1] is 0.0
The relationship between vectors[0] and vectors[1] is The vectors are orthogonal
The relationship between vectors[1] and vectors[2] is The vectors are neither collinear nor orthogonal
The relationship between vectors[0] and vectors[2] is The vectors are neither collinear nor orthogonal
The relationship between vectors[0] and vectors[2] is The vectors are collinear

Process finished with exit code 0
```

Рисунок 1 – Результат работы программы

3. Определить класс Вектор в R3. 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 2 – Код программы

```
import java.util.Arrays;
 class Vector {
    private double x;
    private double y;
    private double z;

    public Vector(double x, double y, double z) {
        this.x = x;
        this.y = y;
        this.z = z;
    }

    public Vector() {
        this(0, 0, 0);
    }

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

    public double scalarProduct(Vector v) {
        return x * v.x + y * v.y + z * v.z;
    }

    public Vector add(Vector v) {
        return new Vector(x + v.x, y + v.y, z + v.z);
```

```
}

public Vector subtract(Vector v) {
   return new Vector(x - v.x, y - v.y, z - v.z);
}

public Vector multiply(double k) {
   return new Vector(k * x, k * y, k * z);
}

public boolean isCollinear(Vector v) {
   return x / v.x == y / v.y && y / v.y == z / v.z;
}

public boolean isOrthogonal(Vector v) {
   return scalarProduct(v) == 0;
}


 public boolean intersects(Vector v) {
   return !isOrthogonal(v);
}

public boolean equals(Vector v) {
   return x == v.x && y == v.y && z == v.z;
}

public int compareTo(Vector v) {
   double modulus1 = modulus();
   double modulus2 = v.modulus();
   if (modulus1 < modulus2) {
      return -1;
   } else if (modulus1 > modulus2) {
      return 1;
   } else {
      return 0;
   }
}

public String toString() {
   return "(" + x + ", " + y + ", " + z + ")";
}

public static void main(String[] args) {
   int m = 3;
   Vector[] vectors = new Vector[m];
   for (int i = 0; i < m; i++) {
      vectors[i] = new Vector();
   }
   Vector v1 = new Vector(1, 2, 3);
   Vector v2 = new Vector(2, 4, 6);
   Vector v3 = new Vector(2, -1, 0);
   for (int i = 0; i < m; i++) {
      for (int j = i + 1; j < m; j++) {
         for (int k = j + 1; k < m; k++) {
            if (isCoplanar(vectors[i], vectors[j], vectors[k])) {
               System.out.println("Vectors are coplanar.");
            }
```

```
            }
        }
    }

    // Example usage of methods


    // Check orthogonality
    System.out.println("v1 and v2 are orthogonal: " + v1.isOrthogonal(v2));
    System.out.println("v1 and v3 are orthogonal: " + v1.isOrthogonal(v3));

    System.out.println("v1 and v3 are collinear: " + v1.isCollinear(v2));

    // Check intersection
    System.out.println("v1 and v2 intersect: " + v1.intersects(v2));
    System.out.println("v1 and v3 intersect: " + v1.intersects(v3));

    // Compare vectors
    System.out.println("v1 compared to v2: " + v1.compareTo(v2));
    System.out.println("v2 compared to v1: " + v2.compareTo(v1));
    System.out.println("v1 compared to v3: " + v1.compareTo(v3));

  }

  public static boolean isCoplanar(Vector v1, Vector v2, Vector v3) {
    return v1.scalarProduct(v2.crossProduct(v3)) == 0;
  }

  public Vector crossProduct(Vector v) {
    return new Vector(y * v.z - z * v.y, z * v.x - x * v.z, x * v.y - y * v.x);
  }

}
```

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaager
Vectors are coplanar.
v1 and v2 are orthogonal: false
v1 and v3 are orthogonal: true
v1 and v3 are collinear: true
v1 and v2 intersect: true
v1 and v3 intersect: false
v1 compared to v2: -1
v2 compared to v1: 1
v1 compared to v3: 1

Process finished with exit code 0
```

Рисунок 2 – Результат работы программы

Вариант 2

2. Customer: id, Фамилия, Имя, Отчество, Адрес, Номер кредитной карточки, Номер банковского счета. Создать массив объектов. Вывести: а) список

покупателей в алфавитном порядке; b) список покупателей, у которых номер кредитной карточки находится в заданном интервале.

Листинг 3 – Код программы

```
import java.util.Scanner;
import java.util.Comparator;
import java.util.Arrays;

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      Customer[] cust = new Customer[3];
      cust[0] = new Customer(1, "Anna;  ", "Petrova;  ", "Petrovna;  ", "Moscow;  ", 123, 56521);
      cust[1] = new Customer(2, "Boris;  ", "Artemyev;  ", "Alexandrovich;  ", "Zelenograd;  ", 321, 45545);
      cust[2] = new Customer(3, "Alexandr;  ", "Shirokov;  ", "Arsenyevich;  ", "Khimki;  ", 461, 12357);



      while (true) {
         System.out.println(
               "Выберете пункт меню:\n"
                     + "1. вывод покупателей в алфавитном порядке\n"
                     + "2. вывод покупателей с кредитной картой в интервале\n"
                     + ": "
         );
         int choice = scanner.nextInt();
         if (choice == 0)
            break;
         if (choice < 1 || choice > 3) {
            System.out.println("выбран неправильный пункт меню, повторите ввод.");
            continue;
         }

         switch (choice) {
            case 1:
               Arrays.sort(cust, Comparator.comparing(Customer::getSurname));
               for (int i = 0; i <= 2; i++) {
                  System.out.println(cust[i]);
               }
               break;
            case 2:
               long numto = 0, numfrom = 0;
               System.out.println("Введите диапазон кредитных карт\n");
               System.out.println("От: ");
               int n1 = scanner.nextInt();
               System.out.println("До: ");
               int n2 = scanner.nextInt();
               Customer.customer_check_credit_card(cust,n1,n2);
         }
      }
   }
}
```

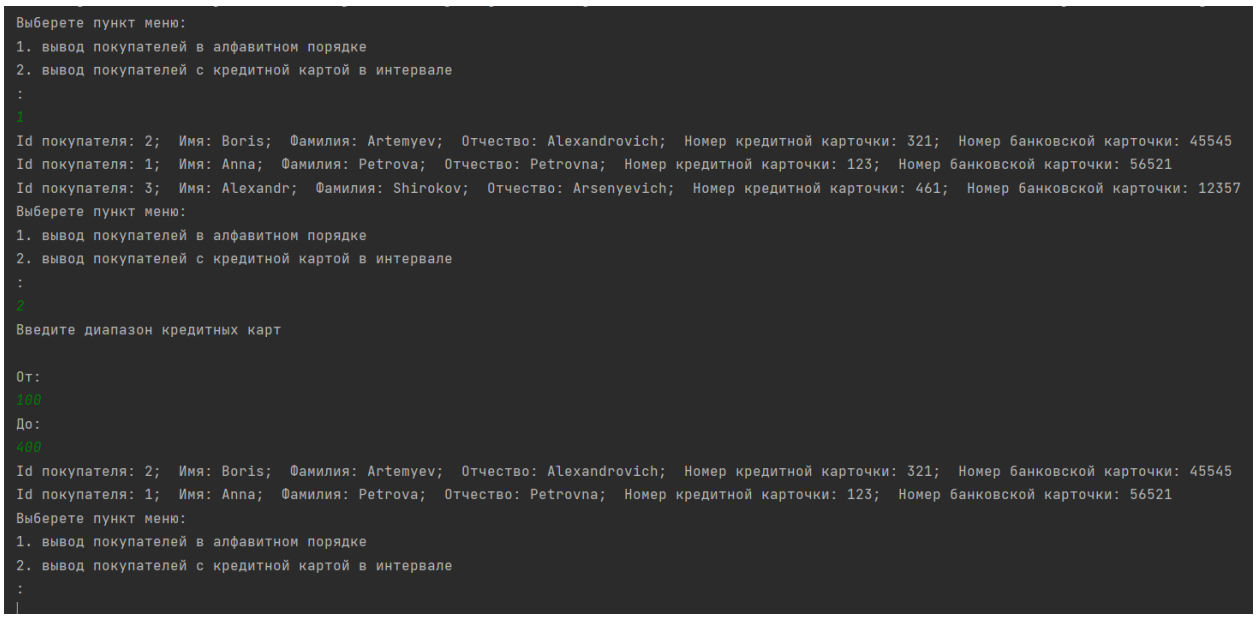

Рисунок 3 – Результат работы программы

3. Patient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Номер медицинской карты, Диагноз. Создать массив объектов. Вывести: a) список пациентов, имеющих данный диагноз; b) список пациентов, номер медицинской карты у которых находится в заданном интервале.

Листинг 4 – Код программы

```
import java.util.Scanner;
import java.util.Comparator;
import java.util.Arrays;

public class Main {
   public static void main(String[] args) {
      Scanner scanner = new Scanner(System.in);
      Patient[] pats = new Patient[3];
      pats[0] = new Patient(1, "Anna;  ", "Petrova;  ", "Petrovna;  ", "Moscow;  ", 798612312, 123, "Pain1");
      pats[1] = new Patient(2, "Boris;  ", "Artemyev;  ", "Alexandrovich;  ", "Zelenograd;  ", 798612312, 14,
"Pain2");
      pats[2] = new Patient(3, "Alexandr;  ", "Shirokov;  ", "Arsenyevich;  ", "Khimki;  ", 798612312, 450, "Pain3");



      while (true) {
         System.out.println(
               "Выберете пункт меню:\n"
                     + "1. вывод пациентов с заданным диагнозом\n"
                     + "2. вывод пациентов с мед картой в интервале\n"
                     + ": "
         );
         int choice = scanner.nextInt();
         if (choice == 0)
            break;
         if (choice < 1 || choice > 3) {
            System.out.println("выбран неправильный пункт меню, повторите ввод.");
            continue;
         }
```

```
        switch (choice) {
            case 1:
                System.out.print("Введите диагноз для поиска пациентов: ");
                String diagnosis= scanner.next();
                Patient.check_diagnosis(pats,diagnosis);
                break;
            case 2:
                long numto = 0, numfrom = 0;
                System.out.println("Введите диапазон мед карт\n");
                System.out.println("От: ");
                int n1 = scanner.nextInt();
                System.out.println("До: ");
                int n2 = scanner.nextInt();
                Patient.customer_check_med_card(pats,n1,n2);
        }
    }
  }
}
```

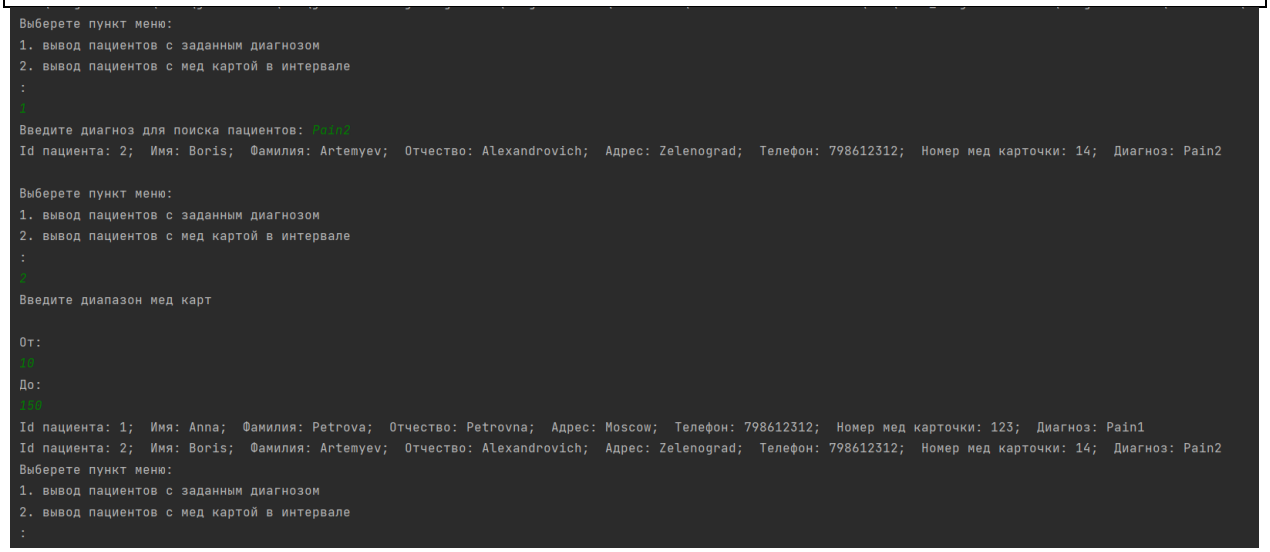

Рисунок 4 – Результат работы программы

Вариант 3

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

2. Создать объект класса Одномерный массив, используя класс Массив. Методы: создать, вывести на консоль, выполнить операции (сложить, вычесть, перемножить).

Листинг 5 – Код программы

```
import java.util.Arrays;

class OneDimensionalArray {
   private int[] array;

   public OneDimensionalArray(int[] array) {
      this.array = array;
   }
```

```
public OneDimensionalArray(int length) {
    this.array = new int[length];
}

public void fillRandom() {
    for (int i = 0; i < this.array.length; i++) {
        this.array[i] = (int) (Math.random() * 100);
    }
}

public void output() {
    System.out.println(Arrays.toString(this.array));
}

public static OneDimensionalArray add(OneDimensionalArray a1, OneDimensionalArray a2) {
    OneDimensionalArray result = new OneDimensionalArray(a1.array.length);
    for (int i = 0; i < a1.array.length; i++) {
        result.array[i] = a1.array[i] + a2.array[i];
    }
    return result;
}

public static OneDimensionalArray subtract(OneDimensionalArray a1, OneDimensionalArray a2) {
    OneDimensionalArray result = new OneDimensionalArray(a1.array.length);
    for (int i = 0; i < a1.array.length; i++) {
        result.array[i] = a1.array[i] - a2.array[i];
    }
    return result;
}

public OneDimensionalArray multiply(int constant) {
    OneDimensionalArray result = new OneDimensionalArray(this.array.length);
    for (int i = 0; i < this.array.length; i++) {
        result.array[i] = this.array[i] * constant;
    }
    return result;
}

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

// Override the hashCode() method
@Override
public int hashCode() {
    return Arrays.hashCode(array);
}

// Override the toString() method
@Override
public String toString() {
    return "OneDimensionalArray{" +
            "array=" + Arrays.toString(array) +
```

```
            '}';
    }

    // Example usage
    public static void main(String[] args) {
        OneDimensionalArray a1 = new OneDimensionalArray(new int[]{1, 2, 3});
        OneDimensionalArray a2 = new OneDimensionalArray(new int[]{4, 5, 6});
        OneDimensionalArray sum = add(a1, a2);
        OneDimensionalArray difference = subtract(a1, a2);
        OneDimensionalArray product = a1.multiply(2);
        System.out.println(sum); // Output: OneDimensionalArray{array=[5, 7, 9]}
        System.out.println(difference); // Output: OneDimensionalArray{array=[-3, -3, -3]}
        System.out.println(product); // Output: OneDimensionalArray{array=[2, 4, 6]}
    }
}
```

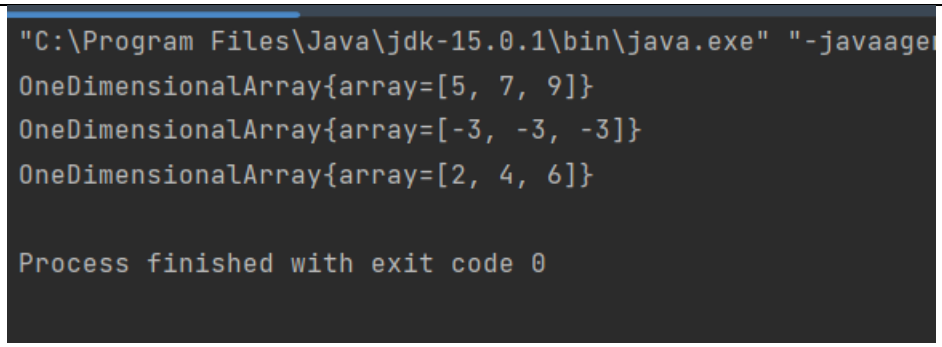

Рисунок 5 – Результат работы программы

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

Листинг 6 – Код программы

```
import java.util.Objects;

class Number {
    private int numerator;
    private int denominator;

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

    public int getNumerator() {
        return numerator;
    }

    public int getDenominator() {
        return denominator;
    }

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

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

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

class SimpleFraction extends Number {
    public SimpleFraction(int numerator, int denominator) {
        super(numerator, denominator);
    }

    public void display() {
        System.out.println("Fraction: " + this.getNumerator() + "/" + this.getDenominator());
    }

    public SimpleFraction addition(SimpleFraction fraction) {
        int numerator = this.getNumerator() * fraction.getDenominator() + fraction.getNumerator() *
this.getDenominator();
        int denominator = this.getDenominator() * fraction.getDenominator();
        return new SimpleFraction(numerator, denominator);
    }

    public SimpleFraction subtraction(SimpleFraction fraction) {
        int numerator = this.getNumerator() * fraction.getDenominator() - fraction.getNumerator() *
this.getDenominator();
        int denominator = this.getDenominator() * fraction.getDenominator();
        return new SimpleFraction(numerator, denominator);
    }

    public SimpleFraction multiplication(SimpleFraction fraction) {
        int numerator = this.getNumerator() * fraction.getNumerator();
        int denominator = this.getDenominator() * fraction.getDenominator();
        return new SimpleFraction(numerator, denominator);
    }

    public SimpleFraction division(SimpleFraction fraction) {
        int numerator = this.getNumerator() * fraction.getDenominator();
        int denominator = this.getDenominator() * fraction.getNumerator();
        return new SimpleFraction(numerator, denominator);
    }

    @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;
        SimpleFraction that = (SimpleFraction) o;
        return this.getNumerator() == that.getNumerator() && this.getDenominator() == that.getDenominator();
    }

    @Override
    public int hashCode() {
```

```
        return Objects.hash(super.hashCode());
    }

    @Override
    public String toString() {
        return super.toString();
    }
}

public class Main {
    public static void main(String[] args) {
        SimpleFraction fraction1 = new SimpleFraction(2, 3);
        SimpleFraction fraction2 = new SimpleFraction(3, 4);

        System.out.println("Fraction 1: " + fraction1.getNumerator() + "/" + fraction1.getDenominator());
        System.out.println("Fraction 2: " + fraction2.getNumerator() + "/" + fraction2.getDenominator());

        SimpleFraction resultAdd = fraction1.addition(fraction2);
        System.out.println("Add: " + resultAdd.getNumerator() + "/" + resultAdd.getDenominator());
        SimpleFraction resultSub = fraction1.subtraction(fraction2);
        System.out.println("Sub: " + resultSub.getNumerator() + "/" + resultSub.getDenominator());
        SimpleFraction resultMul = fraction1.multiplication(fraction2);
        System.out.println("Multip: " + resultMul.getNumerator() + "/" + resultMul.getDenominator());
        SimpleFraction resultDiv = fraction1.division(fraction2);
        System.out.println("Div: " + resultDiv.getNumerator() + "/" + resultDiv.getDenominator());


        System.out.println("Equals: " + resultAdd.equals(resultSub));
        System.out.println("HashCode: " + resultAdd.hashCode());
        System.out.println("ToString: " + resultAdd.toString());
    }
}
```

```
"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-java
Fraction 1: 2/3
Fraction 2: 3/4
Add: 17/12
Sub: -1/12
Multip: 6/12
Div: 8/9
Equals: false
HashCode: 1531
ToString: 17/12

Process finished with exit code 0
```

Рисунок 6 – Результат работы программы

Вариант 4

2. 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.

Листинг 7 – Код программы

```
class BankAccount {
    private String accountNumber;
    private double balance;

    public BankAccount(String accountNumber, double balance) {
        this.accountNumber = accountNumber;
        this.balance = balance;
    }

    public String getAccountNumber() {
        return accountNumber;
    }

    public double getBalance() {
        return balance;
    }

    public void deposit(double amount) {
        balance += amount;
    }

    public void withdraw(double amount) {
        balance -= amount;
    }
}

class CreditCard {
    private String cardNumber;
    private double creditLimit;
    private double balance;
    private boolean isBlocked;

    public CreditCard(String cardNumber, double creditLimit) {
        this.cardNumber = cardNumber;
        this.creditLimit = creditLimit;
        this.balance = 0;
        this.isBlocked = false;
    }

    public String getCardNumber() {
        return cardNumber;
    }

    public double getCreditLimit() {
        return creditLimit;
    }

    public double getBalance() {
        return balance;
    }

    public boolean isBlocked() {
        return isBlocked;
    }

    public void makePurchase(double amount) {
        if (!isBlocked && balance + amount <= creditLimit) {
```

```
            balance += amount;
        }
    }

    public void makePayment(double amount) {
        if (!isBlocked) {
            balance -= amount;
        }
    }

    public void blockCard() {
        isBlocked = true;
    }

    public void unblockCard() {
        isBlocked = false;
    }
}

class Client {
    private String name;
    private BankAccount bankAccount;
    private CreditCard creditCard;

    public Client(String name, BankAccount bankAccount, CreditCard creditCard) {
        this.name = name;
        this.bankAccount = bankAccount;
        this.creditCard = creditCard;
    }

    public String getName() {
        return name;
    }

    public BankAccount getBankAccount() {
        return bankAccount;
    }

    public CreditCard getCreditCard() {
        return creditCard;
    }

    public void payOrder(double amount) {
        if (bankAccount.getBalance() >= amount) {
            bankAccount.withdraw(amount);
        } else if (creditCard.getBalance() + bankAccount.getBalance() >= amount) {
            double remainingAmount = amount - bankAccount.getBalance();
            bankAccount.withdraw(bankAccount.getBalance());
            creditCard.makePurchase(remainingAmount);
        }
    }

    public void makePaymentToAnotherAccount(BankAccount recipient, double amount) {
        if (bankAccount.getBalance() >= amount) {
            bankAccount.withdraw(amount);
            recipient.deposit(amount);
        }
    }
```

```
    public void blockCreditCardByAdministrator() {
        if (creditCard.getBalance() > creditCard.getCreditLimit()) {
            creditCard.blockCard();
        }
    }

    public void cancelBankAccount() {
        bankAccount = null;
    }

    public void cancelCreditCard() {
        creditCard = null;
    }
}

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

        BankAccount bankAccount1 = new BankAccount("12345678", 5000);
        CreditCard creditCard1 = new CreditCard("11111111", 5000);
        Client client1 = new Client("John", bankAccount1, creditCard1);

        BankAccount bankAccount2 = new BankAccount("87654321", 3000);
        CreditCard creditCard2 = new CreditCard("22222222", 3000);
        Client client2 = new Client("Mary", bankAccount2, creditCard2);

        // Paying for an order
        client1.payOrder(2000);
        System.out.println("Client 1 bank account balance: " + client1.getBankAccount().getBalance());
        System.out.println("Client 1 credit card balance: " + client1.getCreditCard().getBalance());
        System.out.println();
        // Making a payment to another account
        BankAccount recipient = new BankAccount("11112222", 0);
        client2.makePaymentToAnotherAccount(recipient, 1000);
        System.out.println("Client 2 bank account balance: " + client2.getBankAccount().getBalance());
        System.out.println("Recipient bank account balance: " + recipient.getBalance());
        System.out.println();

        // Blocking a credit card
        client1.getCreditCard().makePurchase(4000);
        client1.blockCreditCardByAdministrator();
        System.out.println("Client 1 credit card status: " + client1.getCreditCard().isBlocked());
        System.out.println();

        // Cancelling a bank account and credit card
        client2.cancelBankAccount();
        client2.cancelCreditCard();
        System.out.println("Client 2 bank account: " + client2.getBankAccount());
        System.out.println("Client 2 credit card: " + client2.getCreditCard());
    }
}
```

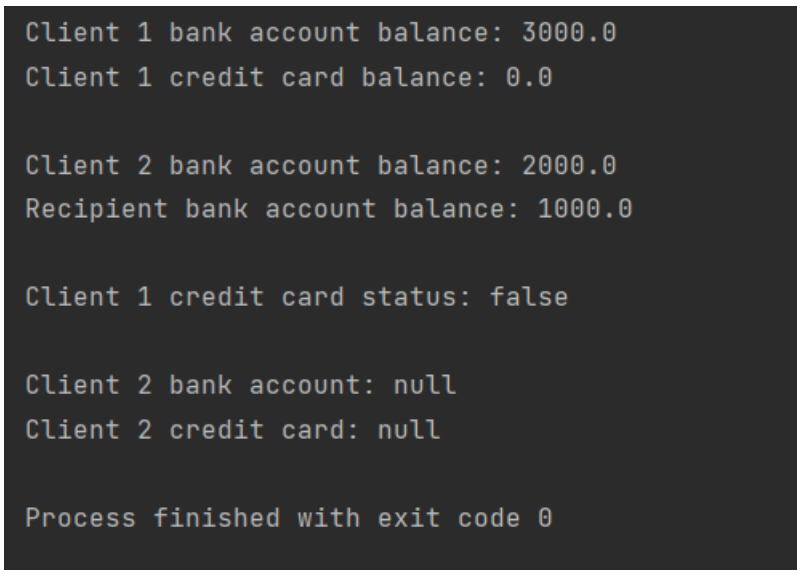

Рисунок 7 – Результат работы программы

3. 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 8 – Код программы

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

class Patient {
    private String name;
    private Doctor attendingDoctor;
    private boolean isAdmitted;

    public Patient(String name, Doctor attendingDoctor) {
        this.name = name;
        this.attendingDoctor = attendingDoctor;
        this.isAdmitted = false;
    }

    public String getName() {
        return name;
    }

    public Doctor getAttendingDoctor() {
        return attendingDoctor;
    }

    public boolean isAdmitted() {
        return isAdmitted;
    }

    public void admit() {
        isAdmitted = true;
    }
```

```
    public void discharge() {
        isAdmitted = false;
    }
}

class Doctor {
    private String name;

    public Doctor(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void makeAppointment(Patient patient, String procedure) {
        // Perform some checks before making the appointment
        if (!patient.isAdmitted()) {
            System.out.println("Patient is not admitted to the hospital.");
            return;
        }

        if (!patient.getAttendingDoctor().equals(this)) {
            System.out.println("This doctor is not the patient's attending doctor.");
            return;
        }

        // Make the appointment
        System.out.println("Appointment made for " + patient.getName() + " for " + procedure);
    }
}

class Nurse {
    private String name;

    public Nurse(String name) {
        this.name = name;
    }

    public String getName() {
        return name;
    }

    public void performAppointment(Patient patient, String procedure) {
        // Perform some checks before performing the appointment
        if (!patient.isAdmitted()) {
            System.out.println("Patient is not admitted to the hospital.");
            return;
        }

        // Perform the appointment
        System.out.println("Performed " + procedure + " on " + patient.getName());
    }
}

class Hospital {
    private List<Patient> patients;
```

```
    private List<Doctor> doctors;
    private List<Nurse> nurses;

    public Hospital() {
        this.patients = new ArrayList<>();
        this.doctors = new ArrayList<>();
        this.nurses = new ArrayList<>();
    }

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

    public void addDoctor(Doctor doctor) {
        doctors.add(doctor);
    }

    public void addNurse(Nurse nurse) {
        nurses.add(nurse);
    }

    public void dischargePatient(Patient patient) {
        // Perform some checks before discharging the patient
        if (!patient.isAdmitted()) {
            System.out.println("Patient is not admitted to the hospital.");
            return;
        }

        // Discharge the patient
        patient.discharge();
        System.out.println("Discharged " + patient.getName() + " from the hospital.");
    }
}

public class HospitalSystem {
    public static void main(String[] args) {
        // Create hospital
        Hospital hospital = new Hospital();

        // Create doctors
        Doctor doctor1 = new Doctor("Mary Doc");
        Doctor doctor2 = new Doctor("Bob Doc");

        // Add doctors to hospital
        hospital.addDoctor(doctor1);
        hospital.addDoctor(doctor2);

        // Create nurses
        Nurse nurse1 = new Nurse("Mary Nurse");
        Nurse nurse2 = new Nurse("KAte Nurse");

        // Add nurses to hospital
        hospital.addNurse(nurse1);
        hospital.addNurse(nurse2);

        // Create patients
        Patient patient1 = new Patient("Bill Patient", doctor1);
        Patient patient2 = new Patient("Olov Patient", doctor2);
```

```
        // Add patients to hospital
        hospital.addPatient(patient1);
        hospital.addPatient(patient2);

        // Admit patients to hospital
        patient1.admit();
        patient2.admit();

        // Make appointments
        doctor1.makeAppointment(patient1, "checkup");
        doctor2.makeAppointment(patient2, "surgery");

        // Perform appointments
        nurse1.performAppointment(patient1, "blood test");
        nurse2.performAppointment(patient2, "anesthesia");

        // Discharge patients
        hospital.dischargePatient(patient1);
        hospital.dischargePatient(patient2);
    }
}
```

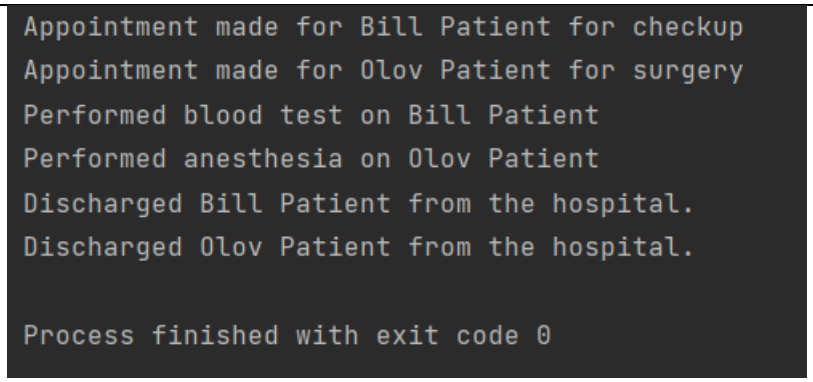

Рисунок 8 – Результат работы программы

Вывод: приобретен навык работы с внутренними классами и интерфейсами.
Ссылка на репозиторий с программами: https://github.com/nargi3/BigData