|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

МАГИСТЕРСКАЯ ПРОГРАММА **09.04.01/12 Интеллектуальный анализ больших**

**данных в системах поддержки принятия решений.**

**Отчет**

**по лабораторной работе № 3**

**Вариант № 3**

**Название:** Классы, наследование и полиморфизм

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

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-23М |  |  | В.Н. Зыкин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2023

**Цель:** освоить принципы ООП на языке программирования Java.

**Задание 1:** 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.

Код класса VectorR3:

import java.util.Arrays;  
  
public class VectorR3 {  
 private double [] components;  
  
 public VectorR3(double [] components) {  
 if (components.length != 3) {  
 throw new RuntimeException("Incorrect vector size");  
 }  
 this.components = components;  
 }  
  
 // Метод для проверки ортогональности векторов  
 public boolean isOrthogonal(VectorR3 other) {  
 double dotProduct = 0;  
 for (int i = 0; i < components.length; i++) {  
 dotProduct += components[i] \* other.components[i];  
 }  
 return dotProduct == 0;  
 }  
  
 // Метод для проверки пересечения не ортогональных векторов  
 public boolean isIntersecting(VectorR3 other) {  
 return !isOrthogonal(other);  
 }  
  
 // Метод для сравнения векторов  
 public boolean isEqual(VectorR3 other) {  
 return Arrays.equals(components, other.components);  
 }  
  
 // Метод для определения компланарности векторов  
 public boolean isCoplanar(VectorR3 v2, VectorR3 v3) {  
 double[][] matrix = {  
 {components[0], components[1], components[2]},  
 {v2.components[0], v2.components[1], v2.components[2]},  
 {v3.components[0], v3.components[1], v3.components[2]}  
 };  
  
 double determinant = calculateDeterminant(matrix);  
 return determinant == 0;  
 }  
  
 // Метод для определения компланарности векторов  
 public static boolean isCoplanar(VectorR3 v1, VectorR3 v2, VectorR3 v3) {  
 double[][] matrix = {  
 {v1.components[0], v1.components[1], v1.components[2]},  
 {v2.components[0], v2.components[1], v2.components[2]},  
 {v3.components[0], v3.components[1], v3.components[2]}  
 };  
  
 double determinant = calculateDeterminant(matrix);  
 return determinant == 0;  
 }  
  
 private static double calculateDeterminant(double[][] matrix) {  
 return matrix[0][0] \* (matrix[1][1] \* matrix[2][2] - matrix[1][2] \* matrix[2][1])  
 - matrix[0][1] \* (matrix[1][0] \* matrix[2][2] - matrix[1][2] \* matrix[2][0])  
 + matrix[0][2] \* (matrix[1][0] \* matrix[2][1] - matrix[1][1] \* matrix[2][0]);  
 }  
  
 public double[] getComponents() {  
 return components;  
 }  
  
 public void setComponents(double[] components) {  
 this.components = components;  
 }  
}

Код класса Main:

import java.util.\*;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.in);  
 List<VectorR3> vectors = new ArrayList<>();  
 System.out.print("Введите количество векторов: ");  
 int m = in.nextInt();  
 System.out.println("Введите координаты векторов:");  
 for(int i = 0; i < m; i++) {  
 double x = in.nextDouble();  
 double y = in.nextDouble();  
 double z = in.nextDouble();  
 vectors.add(new VectorR3(new double[]{x,y,z}));  
 }  
  
 List<VectorR3[]> coplanar = findCoplanarVectors(vectors);  
 System.out.println("Компланарные группы векторов:");  
 for (VectorR3[] group : coplanar) {  
 System.out.println("{");  
 for(VectorR3 vector : group) {  
 System.out.println(Arrays.toString(vector.getComponents()));  
 }  
 System.out.println("}");  
 }  
  
  
  
 }  
  
 // Метод для определения компланарности векторов  
 public static List<VectorR3[]> findCoplanarVectors(List<VectorR3> vectors) {  
 List<VectorR3[]> coplanarVectors = new ArrayList<>();  
  
 for (int i = 0; i < vectors.size()-2; i++) {  
 for(int j = i + 1; j < vectors.size()-1; j++) {  
 for(int k = j + 1; k < vectors.size(); k++) {  
 if(VectorR3.isCoplanar(vectors.get(i), vectors.get(j), vectors.get(k))) {  
 VectorR3 [] group = {vectors.get(i), vectors.get(j), vectors.get(k)};  
 coplanarVectors.add(group);  
 }  
 }  
 }  
 }  
  
 return coplanarVectors;  
 }  
}

Результат работы программы представлен на рисунке 1.

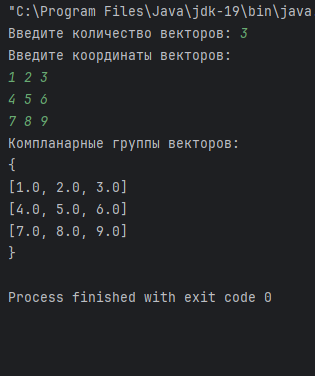


Рисунок 1 – Результат работы программы

**Задание 2:** определить класс Матрица размерности (n x n). Класс должен содержать несколько конструкторов. Реализовать методы для сложения, вычитания, умножения матриц. Объявить массив объектов. Создать методы, вычисляющие первую и вторую нормы матрицы



Определить, какая из матриц имеет наименьшую первую и вторую нормы.

Код класса Matrix:

import java.util.Arrays;  
  
public class Matrix {  
 private int n;  
 private double[][] elements;  
  
 // Конструктор для инициализации матрицы нулями  
 public Matrix(int n) {  
 this.n = n;  
 this.elements = new double[n][n];  
 }  
  
 // Конструктор для инициализации матрицы заданными значениями  
 public Matrix(int n, double[][] elements) {  
 this.n = n;  
 this.elements = Arrays.copyOf(elements, n);  
 }  
  
 // Конструктор для инициализации матрицы другой матрицей  
 public Matrix(Matrix matrix) {  
 this.n = matrix.getN();  
 this.elements = Arrays.copyOf(matrix.getElements(), matrix.getN());  
 }  
  
 // Метод для сложения матриц  
 public Matrix add(Matrix other) {  
 if (this.n != other.n) {  
 throw new IllegalArgumentException("Размерности матриц не совпадают");  
 }  
  
 Matrix result = new Matrix(n);  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 result.elements[i][j] = this.elements[i][j] + other.elements[i][j];  
 }  
 }  
 return result;  
 }  
  
 // Метод для вычитания матриц  
 public Matrix subtract(Matrix other) {  
 if (this.n != other.n) {  
 throw new IllegalArgumentException("Размерности матриц не совпадают");  
 }  
  
 Matrix result = new Matrix(n);  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 result.elements[i][j] = this.elements[i][j] - other.elements[i][j];  
 }  
 }  
 return result;  
 }  
  
 // Метод для умножения матриц  
 public Matrix multiply(Matrix other) {  
 if (this.n != other.n) {  
 throw new IllegalArgumentException("Размерности матриц не совпадают");  
 }  
  
 Matrix result = new Matrix(n);  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 for (int k = 0; k < n; k++) {  
 result.elements[i][j] += this.elements[i][k] \* other.elements[k][j];  
 }  
 }  
 }  
 return result;  
 }  
  
 // Метод для вычисления первой нормы матрицы  
 public double calculateFirstNorm() {  
 double norm = 0;  
 for (int j = 0; j < n; j++) {  
 double columnSum = 0;  
 for (int i = 0; i < n; i++) {  
 columnSum += Math.abs(elements[i][j]);  
 }  
 norm = Math.max(norm, columnSum);  
 }  
 return norm;  
 }  
  
 // Метод для вычисления второй нормы матрицы  
 public double calculateSecondNorm() {  
 double norm = 0;  
 for (int i = 0; i < n; i++) {  
 double rowSum = 0;  
 for (int j = 0; j < n; j++) {  
 rowSum += Math.abs(elements[i][j]);  
 }  
 norm = Math.max(norm, rowSum);  
 }  
 return norm;  
 }  
  
 @Override  
 public String toString() {  
 StringBuilder matrixString = new StringBuilder();  
 for (int i = 0; i < n; i++) {  
 for (int j = 0; j < n; j++) {  
 matrixString.append(elements[i][j]).append(" ");  
 }  
 matrixString.append("\n");  
 }  
 return matrixString.toString();  
 }  
  
 public int getN() {  
 return n;  
 }  
  
 public void setN(int n) {  
 this.n = n;  
 }  
  
 public double[][] getElements() {  
 return elements;  
 }  
  
 public void setElements(double[][] elements) {  
 this.elements = elements;  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
 Matrix matrixA = new Matrix(3, new double[][]{{1, 2, 3}, {4, 5, 6}, {7, 8, 9}});  
 Matrix matrixB = new Matrix(3, new double[][]{{9, 8, 7}, {6, 5, 4}, {3, 2, 1}});  
 Matrix matrixC = new Matrix(3, new double[][]{{3, 5, 3}, {2, 1, 0}, {6, 3, 8}});  
 Matrix matrixD = new Matrix(3, new double[][]{{0, 5, 4}, {4, 7, 1}, {2, 8, 0}});  
  
 Matrix[] matrices = new Matrix[4];  
 matrices[0] = matrixA;  
 matrices[1] = matrixB;  
 matrices[2] = matrixC;  
 matrices[3] = matrixD;  
  
 double minNorm1 = matrices[0].calculateFirstNorm();  
 double minNorm2 = matrices[0].calculateSecondNorm();  
 int minNorm1Ind = 0;  
 int minNorm2Ind = 0;  
  
 for (int i = 1; i < 4; i++) {  
 double norm1 = matrices[i].calculateFirstNorm();  
 double norm2 = matrices[i].calculateSecondNorm();  
 if(norm1 < minNorm1) {  
 minNorm1 = norm1;  
 minNorm1Ind = i;  
 }  
 if(norm2 < minNorm2) {  
 minNorm2 = norm2;  
 minNorm2Ind = i;  
 }  
 }  
 System.out.println("Матрица с минимальной первой нормой:");  
 System.out.println(matrices[minNorm1Ind].toString());  
 System.out.println("Матрица с минимальной второй нормой:");  
 System.out.println(matrices[minNorm2Ind].toString());  
 }  
}

Результат работы программы показан на рисунке 2.

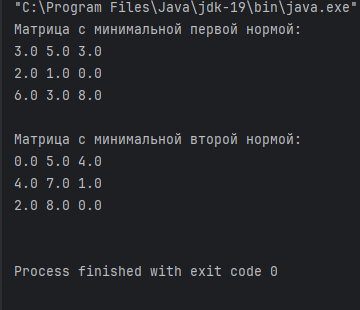


Рисунок 2 – Результат работы программы

**Задание 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.

Код класса Patient:

import java.util.StringJoiner;  
  
public class Patient {  
 private long id;  
 private String firstName;  
 private String secondName;  
 private String patronymic;  
 private String address;  
 private String phone;  
 private long cardNumber;  
 private String diagnose;  
  
 public Patient(long id, String firstName, String secondName, String patronymic, String address, String phone, long cardNumber, String diagnose) {  
 this.id = id;  
 this.firstName = firstName;  
 this.secondName = secondName;  
 this.patronymic = patronymic;  
 this.address = address;  
 this.phone = phone;  
 this.cardNumber = cardNumber;  
 this.diagnose = diagnose;  
 }  
  
 @Override  
 public String toString() {  
 return secondName + " " + firstName + " " + patronymic + ", " + cardNumber;  
 }  
  
 public long getId() {  
 return id;  
 }  
  
 public void setId(long id) {  
 this.id = id;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getSecondName() {  
 return secondName;  
 }  
  
 public void setSecondName(String secondName) {  
 this.secondName = secondName;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public void setPatronymic(String patronymic) {  
 this.patronymic = patronymic;  
 }  
  
 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 getCardNumber() {  
 return cardNumber;  
 }  
  
 public void setCardNumber(long cardNumber) {  
 this.cardNumber = cardNumber;  
 }  
  
 public String getDiagnose() {  
 return diagnose;  
 }  
  
 public void setDiagnose(String diagnose) {  
 this.diagnose = diagnose;  
 }  
}

Код класса Main:

import java.util.ArrayList;  
import java.util.List;  
import java.util.Scanner;  
import java.util.stream.Collectors;  
  
public class Main {  
 public static void main(String[] args) {  
 List<Patient> patients = new ArrayList<>();  
 long id = 0;  
 patients.add(new Patient(id++,  
 "Владислав",  
 "Зыкин",  
 "Николаевич",  
 "Москва",  
 "12345",  
 1,  
 "Усталость"));  
 patients.add(new Patient(id++,  
 "cxvxvc",  
 "fff",  
 "czxcd",  
 "Москва",  
 "12345",  
 2,  
 "Смерть"));  
 patients.add(new Patient(id++,  
 "ывпывп",  
 "смичс",  
 "ывавыа",  
 "Москва",  
 "12345",  
 3,  
 "Усталость"));  
 patients.add(new Patient(id++,  
 "счмисм",  
 "апывп",  
 "вапывап",  
 "Москва",  
 "12345",  
 4,  
 "Смерть"));  
 patients.add(new Patient(id++,  
 "ЫВпвыапр",  
 "пвыапвыа",  
 "вапваыпваып",  
 "Москва",  
 "12345",  
 5,  
 "Усталость"));  
  
 Scanner in = new Scanner(System.in);  
 System.out.println("Введите диагноз:");  
 String diagnose = in.nextLine();  
 System.out.println("Пациенты, обладающие диагнозом \"" + diagnose + "\":");  
 getPatientsWithDiagnose(patients, diagnose).stream().map(Patient::toString).forEach(System.out::println);  
 System.out.println("Задайте интервал номера мед. карты:");  
 int a = in.nextInt();  
 int b = in.nextInt();  
 System.out.println("Пациенты с номером карты в интервале [" + a + ";" + b + "]:");  
 getPatientsInInterval(patients,a,b).stream().map(Patient::toString).forEach(System.out::println);  
 }  
  
 public static List<Patient> getPatientsWithDiagnose(List<Patient> patients, String diagnose) {  
 return patients.stream().filter(p -> p.getDiagnose().equals(diagnose)).collect(Collectors.toList());  
 }  
  
 public static List<Patient> getPatientsInInterval(List<Patient> patients, int a, int b) {  
 return patients.stream().filter(p->p.getCardNumber() >= a && p.getCardNumber() <= b).collect(Collectors.toList());  
 }  
  
}

Результат работы программы представлен на рисунке 3.

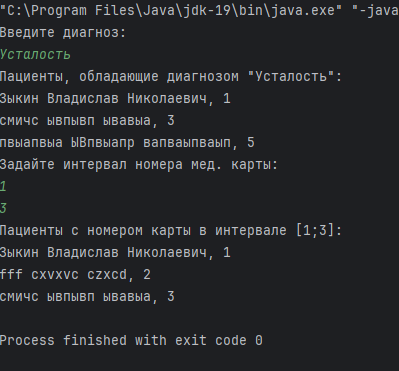
****

Рисунок 3 – Результат работы программы

**Задание 4:** Abiturient: id, Фамилия, Имя, Отчество, Адрес, Телефон, Оценки. Создать массив объектов. Вывести: a) список абитуриентов, имеющих неудовлетворительные оценки; b) список абитуриентов, средний балл у которых выше заданного; c) выбрать заданное число n абитуриентов, имеющих самый высокий средний балл (вывести также полный список абитуриентов, имеющих полупроходной балл).

Код класса Abiturient:

import java.util.Arrays;  
import java.util.Comparator;  
  
public class Abiturient {  
 private int id;  
 private String lastName;  
 private String firstName;  
 private String patronymic;  
 private String address;  
 private String phoneNumber;  
 private int[] grades;  
  
 public Abiturient(int id, String lastName, String firstName, String patronymic,  
 String address, String phoneNumber, int[] grades) {  
 this.id = id;  
 this.lastName = lastName;  
 this.firstName = firstName;  
 this.patronymic = patronymic;  
 this.address = address;  
 this.phoneNumber = phoneNumber;  
 this.grades = grades;  
 }  
  
 @Override  
 public String toString() {  
 return lastName + " " + firstName + " " + patronymic + ", средний балл: " +getAvg();  
 }  
  
 public double getAvg() {  
 return Arrays.stream(grades).average().orElse(0);  
 }  
  
 public int getId() {  
 return id;  
 }  
  
 public void setId(int id) {  
 this.id = id;  
 }  
  
 public String getLastName() {  
 return lastName;  
 }  
  
 public void setLastName(String lastName) {  
 this.lastName = lastName;  
 }  
  
 public String getFirstName() {  
 return firstName;  
 }  
  
 public void setFirstName(String firstName) {  
 this.firstName = firstName;  
 }  
  
 public String getPatronymic() {  
 return patronymic;  
 }  
  
 public void setPatronymic(String patronymic) {  
 this.patronymic = patronymic;  
 }  
  
 public String getAddress() {  
 return address;  
 }  
  
 public void setAddress(String address) {  
 this.address = address;  
 }  
  
 public String getPhoneNumber() {  
 return phoneNumber;  
 }  
  
 public void setPhoneNumber(String phoneNumber) {  
 this.phoneNumber = phoneNumber;  
 }  
  
 public int[] getGrades() {  
 return grades;  
 }  
  
 public void setGrades(int[] grades) {  
 this.grades = grades;  
 }  
}

Код класса Main:

import java.util.\*;  
import java.util.stream.Collectors;  
  
public class Main {  
 public static void main(String[] args) {  
 Abiturient ab1 = new Abiturient(0,"A","B","C","Москва", "2", new int[]{3,3,4,5,5,5});  
 Abiturient ab2 = new Abiturient(1,"F","sadfasdf","sadfasdf","sdfasff", "2", new int[]{4,3,5,5,5,5});  
 Abiturient ab3 = new Abiturient(2,"sdfasf","zxxvc","SAAD","VCXVXC", "SAS", new int[]{5,5,5,5,4,4});  
 Abiturient ab4 = new Abiturient(3,"F","sadfasdf","sadfasdf","sdfasff", "2", new int[]{3,3,2,4,4,4});  
 Abiturient ab5 = new Abiturient(4,"F","sadfasdf","sadfasdf","sdfasff", "2", new int[]{4,3,3,2,4});  
  
 List<Abiturient> abits = Arrays.asList(ab1,ab2,ab3,ab4,ab5);  
  
 System.out.println("Введите проходной:");  
 Scanner in = new Scanner(System.in);  
 double pr = in.nextDouble();  
 System.out.println("Абитуриенты, у которых есть плохие оценки:");  
 findBad(abits).stream().map(Abiturient::toString).forEach(System.out::println);  
 System.out.println("Абитуриенты со средним выше " + pr + ":");  
 findHigherPr(abits, pr).stream().map(Abiturient::toString).forEach(System.out::println);  
 System.out.println("Введите количество:");  
 int count = in.nextInt();  
 System.out.println("Абитуриенты в топ " + count + ":");  
 findTopN(abits, count).stream().map(Abiturient::toString).forEach(System.out::println);  
 System.out.println("Список абитуриентов с полупроходным:");  
 findPoluprohod(abits,count,pr).stream().map(Abiturient::toString).forEach(System.out::println);  
  
 }  
  
 public static List<Abiturient> findBad(List<Abiturient> abiturients) {  
 return abiturients.stream().filter(a -> Arrays.stream(a.getGrades()).anyMatch(g-> g == 2)).collect(Collectors.toList());  
 }  
  
 public static List<Abiturient> findHigherPr(List<Abiturient> abiturients, double pr) {  
 return abiturients.stream().filter(a -> a.getAvg() > pr).collect(Collectors.toList());  
 }  
  
 public static List<Abiturient> findTopN(List<Abiturient> abiturients, int n) {  
 return abiturients.stream()  
 .sorted(Comparator.comparingDouble(Abiturient::getAvg).reversed())  
 .limit(n)  
 .collect(Collectors.toList());  
 }  
  
 //Пусть студенты с полупроходным - те, кто не попали в топ n, но имеют выше проходного  
 public static List<Abiturient> findPoluprohod(List<Abiturient> abiturients, int n, double pr) {  
 return abiturients.stream()  
 .sorted(Comparator.comparingDouble(Abiturient::getAvg))  
 .limit(abiturients.size() - n)  
 .filter(a->a.getAvg() >= pr)  
 .collect(Collectors.toList());  
 }  
  
}

Результат работы программы показан на рисунке 4.

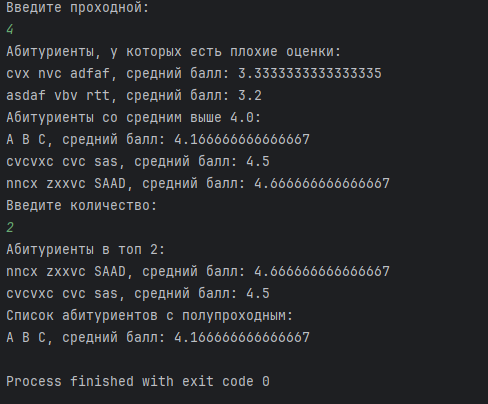


Рисунок 4 – Результат работы программы

**Задание 5:** создать объект класса Простая дробь, используя класс Число. Методы: вывод на экран, сложение, вычитание, умножение, деление.

Код Chislo:

import java.util.Objects;  
  
public class Chislo {  
 private int value;  
  
 public Chislo(int value) {  
 this.value = value;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (!(o instanceof Chislo)) return false;  
 return ((Chislo) o).getValue() == value;  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(value);  
 }  
  
 @Override  
 public String toString() {  
 return String.valueOf(value);  
 }  
  
  
 public int getValue() {  
 return value;  
 }  
  
 public void setValue(int value) {  
 this.value = value;  
 }  
}

Код Drob:

import java.util.Objects;  
  
public class Drob {  
 private Chislo chisl;  
 private Chislo znam;  
  
 public Drob(Chislo chisl, Chislo znam) {  
 this.chisl = chisl;  
 this.znam = znam;  
 }  
  
 public Drob add(Drob other) {  
 int newChisl = this.chisl.getValue() \* other.znam.getValue() + other.chisl.getValue() \* this.znam.getValue();  
 int newZn = this.znam.getValue() \* other.znam.getValue();  
 return new Drob(new Chislo(newChisl), new Chislo(newZn));  
 }  
  
 public Drob sub(Drob other) {  
 int newChisl = this.chisl.getValue() \* other.znam.getValue() - other.chisl.getValue() \* this.znam.getValue();  
 int newZn = this.znam.getValue() \* other.znam.getValue();  
 return new Drob(new Chislo(newChisl), new Chislo(newZn));  
 }  
  
 public Drob mult(Drob other) {  
 int newChisl = this.chisl.getValue() \* other.chisl.getValue();  
 int newZn = this.znam.getValue() \* other.znam.getValue();  
 return new Drob(new Chislo(newChisl), new Chislo(newZn));  
 }  
  
 public Drob div(Drob other) {  
 if (other.chisl.getValue() == 0) {  
 throw new ArithmeticException("Деление на ноль невозможно");  
 }  
 int newChisl = this.chisl.getValue() \* other.znam.getValue();  
 int newZn = this.znam.getValue() \* other.chisl.getValue();  
 return new Drob(new Chislo(newChisl), new Chislo(newZn));  
 }  
  
 public Chislo getChisl() {  
 return chisl;  
 }  
  
 public void setChisl(Chislo chisl) {  
 this.chisl = chisl;  
 }  
  
 public Chislo getZnam() {  
 return znam;  
 }  
  
 public void setZnam(Chislo znam) {  
 this.znam = znam;  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 Drob drob = (Drob) o;  
 return Objects.equals(chisl, drob.chisl) && Objects.equals(znam, drob.znam);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(chisl, znam);  
 }  
  
 @Override  
 public String toString() {  
 return chisl.toString() + "/" + znam.toString();  
 }  
}

Код Main:

import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
 Scanner in = new Scanner(System.in);  
 System.out.println("Введите дробь 1:");  
 Drob drob1 = new Drob(new Chislo(in.nextInt()), new Chislo(in.nextInt()));  
 System.out.println("Введите дробь 2:");  
 Drob drob2 = new Drob(new Chislo(in.nextInt()), new Chislo(in.nextInt()));  
  
 System.out.println("Сложение:");  
 System.out.println(drob1.add(drob2));  
 System.out.println("Вычитание");  
 System.out.println(drob1.sub(drob2));  
 System.out.println("Умножение:");  
 System.out.println(drob1.mult(drob2));  
 System.out.println("Деление:");  
 System.out.println(drob1.div(drob2));  
 }  
}

Результаты работы программы представлены на рисунке 5:

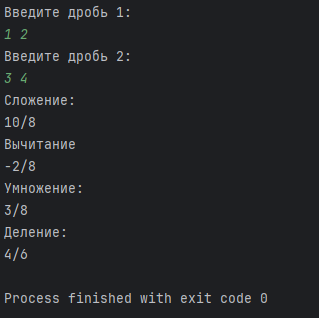


Рисунок 5 – Результаты работы программы

**Задание 6:** создать объект класса Дом, используя классы Окно, Дверь. Методы: закрыть на ключ, вывести на консоль количество окон, дверей.

Код класса Window:

public class Window {  
 public Window() {  
 }  
}

Код класса Door:

public class Door {  
 public Door() {  
 }  
}

Код класса House:

import java.util.ArrayList;  
import java.util.List;  
import java.util.Objects;  
  
class House {  
 private List<Window> windows;  
 private List<Door> doors;  
 private boolean locked;  
  
 public House() {  
 this.windows = new ArrayList<>();  
 this.doors = new ArrayList<>();  
 this.locked = false;  
 }  
 public House(ArrayList<Window> windows, ArrayList<Door> doors) {  
 this.windows = windows;  
 this.doors = doors;  
 this.locked = false;  
 }  
  
 public void addWindow(Window window) {  
 windows.add(window);  
 }  
  
 public void addDoor(Door door) {  
 doors.add(door);  
 }  
  
 public void lock() {  
 locked = true;  
 System.out.println("Дом был закрыт.");  
 }  
  
 public void open() {  
 locked = false;  
 System.out.println("Дом был открыт.");  
 }  
  
 public boolean isLocked() {  
 return this.locked;  
 }  
  
 public void printWindowsCount() {  
 System.out.println("Количество окон: " + windows.size());  
 }  
  
 public void printDoorsCount() {  
 System.out.println("Количество дверей: " + doors.size());  
 }  
  
 @Override  
 public boolean equals(Object o) {  
 if (this == o) return true;  
 if (o == null || getClass() != o.getClass()) return false;  
 House house = (House) o;  
 return locked == house.locked && Objects.equals(windows, house.windows) && Objects.equals(doors, house.doors);  
 }  
  
 @Override  
 public int hashCode() {  
 return Objects.hash(windows.size(), doors.size(), locked);  
 }  
  
 @Override  
 public String toString() {  
 return "Количество окон: " + windows.size()  
 + ", количество дверей: "  
 + doors.size() + ", "  
 + (locked? "закрыт." : "открыт.");  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
 House house = new House();  
 house.addDoor(new Door());  
 house.addDoor(new Door());  
 house.addWindow(new Window());  
 house.open();  
 house.lock();  
 house.printDoorsCount();  
 house.printWindowsCount();  
  
 }  
}

Результат работы программы представлен на рисунке 6:

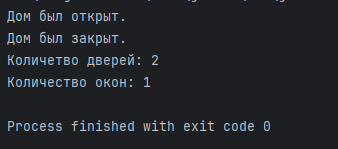


Рисунок 6 – Результат работы программы

**Задание 7:** 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.

Код класса Patient:

public class Main {  
 public static void main(String[] args) {  
 Hospital hospital = new Hospital();  
 Doctor doctor = new Doctor("Dr. Smith");  
 Patient patient = new Patient("John Doe");  
 patient.setAttendingDoctor(doctor);  
 hospital.admitPatient(patient);  
 Prescription procedurePrescription = new ProcedurePrescription(doctor,"X-ray");  
 patient.receivePrescription(procedurePrescription);  
  
 Prescription medicationPrescription = new MedicationPrescription(doctor,"Painkiller");  
 patient.receivePrescription(medicationPrescription);  
 patient.discharge();  
 }  
}

Код класса Doctor:

public class Main {  
 public static void main(String[] args) {  
 Hospital hospital = new Hospital();  
 Doctor doctor = new Doctor("Dr. Smith");  
 Patient patient = new Patient("John Doe");  
 patient.setAttendingDoctor(doctor);  
 hospital.admitPatient(patient);  
 Prescription procedurePrescription = new ProcedurePrescription(doctor,"X-ray");  
 patient.receivePrescription(procedurePrescription);  
  
 Prescription medicationPrescription = new MedicationPrescription(doctor,"Painkiller");  
 patient.receivePrescription(medicationPrescription);  
 patient.discharge();  
 }  
}

Код класса Hospital:

import java.util.ArrayList;  
import java.util.List;  
  
/\*\*  
 \* Больница  
 \*/  
class Hospital {  
 private List<Patient> patients;  
  
 public Hospital() {  
 this.patients = new ArrayList<>();  
 }  
  
 public void admitPatient(Patient patient) {  
 patients.add(patient);  
 System.out.println(patient + " поступил в больницу..");  
 }  
  
 public List<Patient> getPatients() {  
 return patients;  
 }  
  
 public void setPatients(List<Patient> patients) {  
 this.patients = patients;  
 }  
}

Код класса Prescription:

/\*\*  
 \* Назначение  
 \*/  
abstract class Prescription {  
 protected Doctor who;  
 abstract void execute(Patient patient);  
  
 public Doctor getWho() {  
 return who;  
 }  
  
 public void setWho(Doctor who) {  
 this.who = who;  
 }  
}

Код класса MedicationPrescription:

/\*\*  
 \* Назначение лекарства  
 \*/  
class MedicationPrescription extends Prescription {  
 private String medication;  
  
 public MedicationPrescription(Doctor who, String medication) {  
 this.who = who;  
 this.medication = medication;  
 }  
  
 @Override  
 void execute(Patient patient) {  
 System.out.println("Врач " + who + " вводит лекарство: " + medication + " пациенту " + patient);  
 }  
  
 public String getMedication() {  
 return medication;  
 }  
  
 public void setMedication(String medication) {  
 this.medication = medication;  
 }  
}

Код класса ProcedurePrecsiption:

/\*\*  
 \* Назначение процедуры  
 \*/  
class ProcedurePrescription extends Prescription {  
 private String procedure;  
  
 public ProcedurePrescription(Doctor who, String procedure) {  
 this.who = who;  
 this.procedure = procedure;  
 }  
  
 @Override  
 void execute(Patient patient) {  
 System.out.println("Врач " + who + " проводит процедуру: " + procedure + " пациенту " + patient);  
 }  
  
 public String getProcedure() {  
 return procedure;  
 }  
  
 public void setProcedure(String procedure) {  
 this.procedure = procedure;  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
 Hospital hospital = new Hospital();  
 Doctor doctor = new Doctor("Dr. Smith");  
 Patient patient = new Patient("John Doe");  
 patient.setAttendingDoctor(doctor);  
 hospital.admitPatient(patient);  
 Prescription procedurePrescription = new ProcedurePrescription(doctor,"X-ray");  
 patient.receivePrescription(procedurePrescription);  
  
 Prescription medicationPrescription = new MedicationPrescription(doctor,"Painkiller");  
 patient.receivePrescription(medicationPrescription);  
 patient.discharge();  
 }  
}

Результат работы программы представлен на рисунке 7:

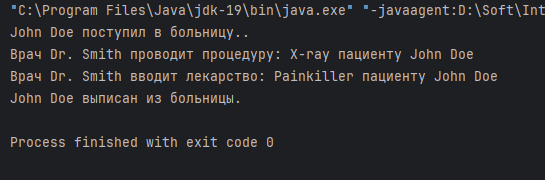


Рисунок 7 – Результаты работы программы

**Задание 7:** система Вступительные экзамены. Абитуриент регистрируется на Факультет, сдает Экзамены. Преподаватель выставляет Оценку. Система подсчитывает средний балл и определяет Абитуриентов, зачисленных в учебное заведение.

Код класса Abiturient:

import java.util.ArrayList;  
import java.util.List;  
  
public class Abiturient {  
 private String name;  
 private Faculty registeredFaculty;  
 private List<Exam> exams;  
 private double averageScore;  
 private boolean isAdmited;  
  
 public Abiturient(String name) {  
 this.name = name;  
 this.exams = new ArrayList<>();  
 }  
  
 public void registerForFaculty(Faculty faculty) {  
 this.registeredFaculty = faculty;  
 }  
  
 public void takeExam(Exam exam, Teacher teacher) {  
 teacher.gradeExam(this, exam);  
 }  
  
 public void calculateAverageScore() {  
 if (!exams.isEmpty()) {  
 double sum = exams.stream().mapToDouble(Exam::getScore).sum();  
 averageScore = sum / exams.size();  
 } else {  
 averageScore = 0.0;  
 }  
 }  
  
 public void displayAdmissionStatus() {  
 String admissionStatus = isAdmited() ? "поступил(а)" : "не поступил(а)";  
 System.out.println(name + " " + admissionStatus + " на факультет " + registeredFaculty.getName());  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public Faculty getRegisteredFaculty() {  
 return registeredFaculty;  
 }  
  
 public void setRegisteredFaculty(Faculty registeredFaculty) {  
 this.registeredFaculty = registeredFaculty;  
 }  
  
 public List<Exam> getExams() {  
 return exams;  
 }  
  
 public void setExams(List<Exam> exams) {  
 this.exams = exams;  
 }  
  
 public double getAverageScore() {  
 return averageScore;  
 }  
  
 public void setAverageScore(double averageScore) {  
 this.averageScore = averageScore;  
 }  
  
 public boolean isAdmited() {  
 return isAdmited;  
 }  
  
 public void setAdmited(boolean admited) {  
 isAdmited = admited;  
 }  
}

Код класса Exam:

public class Exam {  
 private String subject;  
 private int score;  
  
 public Exam(String subject) {  
 this.subject = subject;  
 }  
  
 public void setScore(int score) {  
 this.score = score;  
 }  
  
 public double getScore() {  
 return score;  
 }  
  
 public String getSubject() {  
 return subject;  
 }  
  
 public void setSubject(String subject) {  
 this.subject = subject;  
 }  
}

Код класса Faculty:

import java.util.ArrayList;  
import java.util.Comparator;  
import java.util.List;  
  
class Faculty {  
 private String name;  
 private int budgets;  
 private List<Abiturient> abiturientList;  
  
 public Faculty(String name, int budgets) {  
 this.name = name;  
 this.budgets = budgets;  
 this.abiturientList = new ArrayList<>();  
 }  
  
 public void calculateWhoAdmitted() {  
 abiturientList.forEach(Abiturient::calculateAverageScore);  
 abiturientList.stream()  
 .sorted(Comparator.comparingDouble(Abiturient::getAverageScore).reversed())  
 .limit(budgets)  
 .forEach(a -> a.setAdmited(true));  
 }  
  
 public void printResults() {  
 abiturientList.forEach(Abiturient::displayAdmissionStatus);  
 }  
  
 public void addAbiturient(Abiturient abiturient) {  
 abiturientList.add(abiturient);  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public int getBudgets() {  
 return this.budgets;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public void setBudgets(int budgets) {  
 this.budgets = budgets;  
 }  
  
 public List<Abiturient> getAbiturientList() {  
 return abiturientList;  
 }  
  
 public void setAbiturientList(List<Abiturient> abiturientList) {  
 this.abiturientList = abiturientList;  
 }  
}

Код класса Teacher:

import java.util.Random;  
  
class Teacher {  
 private String name;  
  
 public Teacher(String name) {  
 this.name = name;  
 }  
  
 public void gradeExam(Abiturient abiturient, Exam exam) {  
 Random random = new Random();  
 int score = random.nextInt(4)+2; // Рандомно от 2 до 5  
 exam.setScore(score);  
 System.out.println("Преподаватель " + name + " выставил оценку " + score + " абитуриенту " + abiturient.getName());  
 }  
}

Код класса Main:

public class Main {  
 public static void main(String[] args) {  
  
 Faculty faculty = new Faculty("Прикладная информатика", 2);  
  
 Abiturient abiturient1 = new Abiturient("ВВ АА ЫЫ");  
 Abiturient abiturient2 = new Abiturient("sd fsadh ffds");  
 Abiturient abiturient3 = new Abiturient("sdf sdf sdf");  
  
 //Регистрация на факультеты  
 faculty.addAbiturient(abiturient1);  
 faculty.addAbiturient(abiturient2);  
 faculty.addAbiturient(abiturient3);  
 abiturient1.registerForFaculty(faculty);  
 abiturient2.registerForFaculty(faculty);  
 abiturient3.registerForFaculty(faculty);  
  
 Exam mathExam = new Exam("Математика");  
 Exam infoExam = new Exam("Информатика");  
  
 // Преподователь  
 Teacher teacherMath = new Teacher("ВВ фф СС");  
 Teacher teacherInfo = new Teacher("FFd dsf dfsdf");  
  
 // Сдача экзаменов  
 abiturient1.takeExam(mathExam, teacherMath);  
 abiturient2.takeExam(mathExam, teacherMath);  
 abiturient3.takeExam(mathExam, teacherMath);  
  
 abiturient1.takeExam(infoExam, teacherInfo);  
 abiturient2.takeExam(infoExam, teacherInfo);  
 abiturient3.takeExam(infoExam, teacherInfo);  
  
 faculty.calculateWhoAdmitted();  
 faculty.printResults();  
  
  
 }  
}

Результат работы программы представлен на рисунке 8:

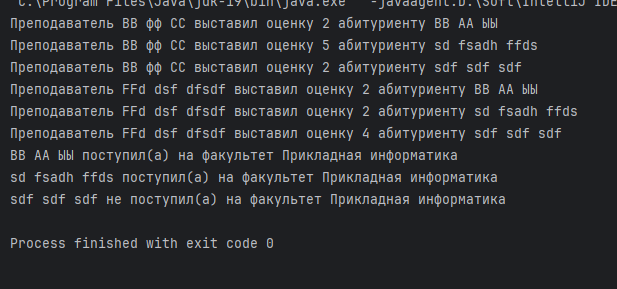


Рисунок 8 – Результат работы программы

**Вывод:** были освоены принципы ООП на языке программирования Java.