|  |  |
| --- | --- |
| 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 № 5**

**Вариант № 3**

**Название:** Исключения и файлы

**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:** изучить и освоить работу с исключениями и файлами в Java.

**Задание 1:** выполнить задания на основе варианта 1 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

Код класса VectorR3:

public class Mobile {  
 private Model model;  
  
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) {  
 try (Scanner in = new Scanner(System.in)) {  
 List<VectorR3> vectors = new ArrayList<>();  
 System.out.print("Введите количество векторов: ");  
 int m;  
 while (true) {  
 try {  
 m = in.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Необходимо ввести целое число! Повторите ввод!");  
 in.next();  
 }  
 }  
  
 System.out.println("Введите координаты векторов:");  
 for (int i = 0; i < m; i++) {  
 try {  
 double x = in.nextDouble();  
 double y = in.nextDouble();  
 double z = in.nextDouble();  
 vectors.add(new VectorR3(new double[]{x, y, z}));  
 } catch (InputMismatchException e) {  
 System.out.println("Необходимо ввести 3 числа! Повторите ввод");  
 in.next();  
 i--; //Откат назад  
 }  
 }  
  
 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("}");  
 }  
  
  
 } catch (OutOfMemoryError e) {  
 System.out.println("Недостаточно памяти!");  
 } catch (NoSuchElementException e) {  
 System.out.println("Не были введены все данные!");  
 } catch (Exception e) {  
 System.out.println("Произошла непредвиденная ошибка: " + e.getMessage());  
 }  
 }  
  
 // 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.

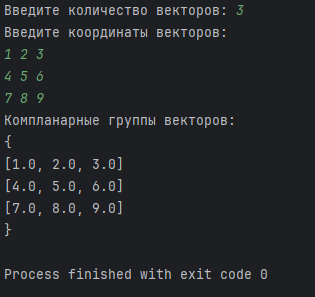


Рисунок 1 – Результат работы программы

**Задание 2:** выполнить задания на основе варианта 1 лабораторной работы 3, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

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:

import java.util.InputMismatchException;  
import java.util.NoSuchElementException;  
import java.util.Scanner;  
  
public class Main {  
 public static void main(String[] args) {  
  
 try (Scanner in = new Scanner(System.in)) {  
 int m;  
 System.out.println("Введите количество матриц");  
 while (true) {  
 try {  
 m = in.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Ошибка ввода. Введите целое число.");  
 in.next();  
 }  
 }  
  
 Matrix[] matrices = new Matrix[m];  
 for (int i = 0; i < m; i++) {  
 try {  
 System.out.println("Введите размерность матрицы " + i + 1 + ":");  
 int n = in.nextInt();  
 System.out.println("Введите матрицу");  
 double[][] arr = new double[n][n];  
 for (int j = 0; j < n; j++) {  
 for (int k = 0; k < n; k++) {  
 arr[j][k] = in.nextDouble();  
 }  
 }  
 Matrix matrix = new Matrix(n, arr);  
 matrices[i] = matrix;  
 } catch (InputMismatchException e) {  
 System.out.println("Ошибка ввода! Введите матрицу заново");  
 i--;  
 }  
 }  
  
  
 double minNorm1 = matrices[0].calculateFirstNorm();  
 double minNorm2 = matrices[0].calculateSecondNorm();  
 int minNorm1Ind = 0;  
 int minNorm2Ind = 0;  
  
 for (int i = 1; i < m; 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());  
 } catch (OutOfMemoryError e) {  
 System.out.println("Недостаточно памяти");  
 } catch (NoSuchElementException e) {  
 System.out.println("Не все данные введены");  
 } catch (Exception e) {  
 System.out.println("Произошла непредвиденная ошибка: " + e.getMessage());  
 }  
  
 }  
}

Результат работы программы показан на рисунке 2.

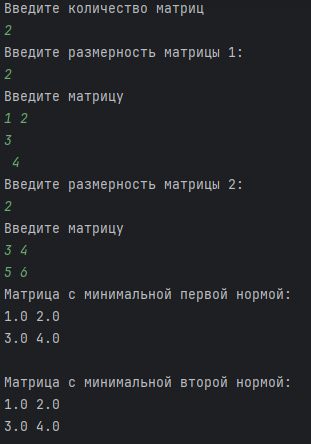


Рисунок 2 – Результат работы программы

**Задание 3:** выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

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.\*;  
import java.util.stream.Collectors;  
  
public class Main {  
 public static void main(String[] args) {  
 List<Patient> patients = new ArrayList<>();  
 Scanner in = new Scanner(System.in);  
 System.out.println("Введите количество пациентов");  
 int n;  
 while(true) {  
 try {  
 n = in.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Невернный ввод. Повторите ещё раз.");  
 in.next();  
 }  
 }  
 in.nextLine();  
 for(int i = 0; i < n; i++) {  
 System.out.println("Введите данные пациента (ФИО, адрес, телефон, карта, диагноз)");  
 try {  
 System.out.println("Имя");  
 String name = in.nextLine();  
 System.out.println("Фамилия");  
 String second = in.nextLine();  
 System.out.println("Отчество");  
 String patronymic = in.nextLine();  
 System.out.println("Адрес");  
 String address = in.nextLine();  
 System.out.println("Телефон");  
 String phone = in.nextLine();  
 System.out.println("Номер карты");  
 int card = in.nextInt();  
 in.nextLine();  
 System.out.println("Диагноз");  
 String diag = in.nextLine();  
 Patient patient = new Patient(i, name, second, patronymic, address, phone, card, diag);  
 System.out.println(patient);  
 patients.add(patient);  
 } catch (InputMismatchException e) {  
 System.out.println("Ошибка ввода! Введите пациента заново!");  
 i--;  
 }  
 }  
 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, b;  
 while(true) {  
 try {  
 a = in.nextInt();  
 b = in.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Ошибка ввода! Повоторите.");  
 in.next();  
 }  
 }  
 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.

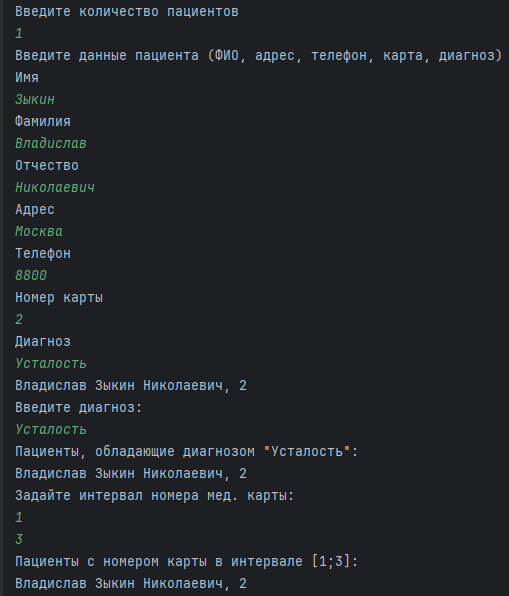
****

Рисунок 3 – Результат работы программы

**Задание 4:** выполнить задания из варианта 2 лабораторной работы 3, реализуя собственные обработчики исключений и исключения ввода/вывода.

Код интерфейса 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) {  
 Scanner in = new Scanner(System.in);  
 System.out.println("Введите кол-во абитуриентов:");  
 List<Abiturient> abits = new ArrayList<>();  
 int n;  
 while (true) {  
 try {  
 n = in.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Ошибка ввода, повторите!");  
 in.next();  
 }  
 }  
 for(int i = 0; i < n; i++) {  
 try {  
 System.out.println("Введите данные абитуриента:");  
 String last = in.nextLine();  
 String first = in.nextLine();  
 String patr = in.nextLine();  
 String address = in.nextLine();  
 String phone = in.nextLine();  
 int[] scores = new int[6];  
 for(int j = 0; j < 6; j++) {  
 scores[j] = in.nextInt();  
 }  
 Abiturient a = new Abiturient(i,last, first, patr, address, phone, scores);  
 abits.add(a);  
 } catch (InputMismatchException e) {  
 System.out.println("Ошибка ввода! Введите абитуриента повторно");  
 in.next();  
 i--;  
 }  
  
 }  
  
  
 System.out.println("Введите проходной:");  
 double pr;  
 while (true) {  
 try {  
 pr = in.nextDouble();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Неверный ввод! Повторите!");  
 in.next();  
 }  
 }  
 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;  
 while (true) {  
 try {  
 count = in.nextInt();  
 break;  
 } catch (InputMismatchException e) {  
 System.out.println("Неверный ввод! Повторите!");  
 in.next();  
 }  
 }  
 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 > 0?abiturients.size() - n : abiturients.size() )  
 .filter(a->a.getAvg() >= pr)  
 .collect(Collectors.toList());  
 }  
  
}

Результат работы программы показан на рисунке 4.

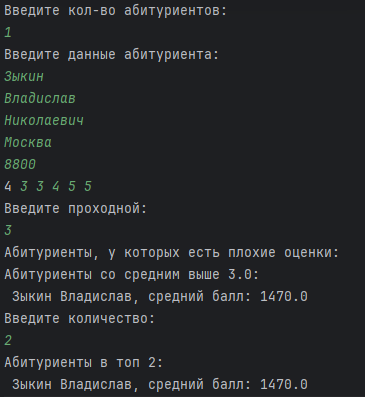


Рисунок 4 – Результат работы программы

**Задание 5:** В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

• каждая строка состоит из одного слова;

• каждая строка состоит из нескольких слов.

В каждой строке найти слова, начинающиеся с гласной буквы.

Код класса Main:

import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
import java.util.regex.Pattern;  
  
public class Main {  
  
 public static void main(String[] args) {  
  
 String filePath = "input.txt";  
 File file = new File(filePath);  
  
 try (Scanner scanner = new Scanner(file)) {  
 while (scanner.hasNextLine()) {  
 String line = scanner.nextLine();  
 findAndPrintWordsStartingWithVowel(line);  
 }  
 } catch (FileNotFoundException e) {  
 System.out.println("Файл не найден: " + e.getMessage());  
 }  
 }  
  
 private static void findAndPrintWordsStartingWithVowel(String line) {  
 // Добавляем русские гласные к английским  
 String vowels = "aeiouAEIOUаеёиоуыэюяАЕЁИОУЫЭЮЯ";  
 // Разбиваем строку на слова  
 String[] words = line.split("\\s+|,\\s\*|\\.\\s\*"); // Делим по пробелам, запятым и точкам  
 for (String word : words) {  
 if (!word.isEmpty() && vowels.indexOf(word.charAt(0)) != -1) {  
 System.out.println(word);  
 }  
 }  
 }  
}

Результат работы программы показан на рисунке 5.

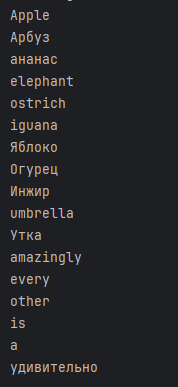


Рисунок 5 – Результаты работы программы

**Задание 6:** в следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

• каждая строка состоит из одного слова;

• каждая строка состоит из нескольких слов.

Найти и вывести слова текста, для которых последняя буква одного слова совпадает с первой буквой следующего слова.

Код класса Main:

import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String filePath = "input.txt";  
 File file = new File(filePath);  
  
 try (Scanner scanner = new Scanner(file)) {  
 StringBuilder contentBuilder = new StringBuilder();  
 while (scanner.hasNextLine()) {  
 contentBuilder.append(scanner.nextLine()).append(" ");  
 }  
 String content = contentBuilder.toString();  
 findAndPrintMatchingWords(content);  
 } catch (FileNotFoundException e) {  
 System.out.println("Файл не найден: " + e.getMessage());  
 }  
 }  
  
 private static void findAndPrintMatchingWords(String content) {  
 // Удаляем знаки препинания для упрощения разбиения на слова  
 String[] words = content.replaceAll("[,.]", "").split("\\s+");  
  
 for (int i = 0; i < words.length - 1; i++) {  
 String currentWord = words[i];  
 String nextWord = words[i + 1];  
  
 if (!currentWord.isEmpty() && !nextWord.isEmpty() &&  
 currentWord.charAt(currentWord.length() - 1) == nextWord.charAt(0)) {  
 System.out.println(currentWord + " " + nextWord);  
 }  
 }  
 }  
}

Результат работы программы представлен на рисунке 6.

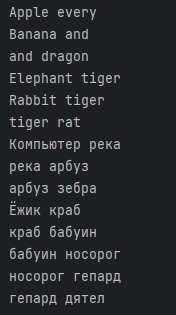


Рисунок 6 – Результат работы программы

**Задание 7**: при выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File.

Прочитать текст Java-программы и в каждом слове длиннее двух символов все строчные символы заменить прописными.

Код класса Main:

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String sourceFilePath = "Test.java"; // Замените на путь к исходному файлу с Java-программой  
 String outputDirPath = "output/"; // Замените на путь к директории для вывода  
 String outputFileName = "Test.java";  
  
 File sourceFile = new File(sourceFilePath);  
 File outputDir = new File(outputDirPath);  
  
 // Создаем директорию, если она еще не существует  
 if (!outputDir.exists()) {  
 outputDir.mkdirs();  
 }  
  
 File outputFile = new File(outputDir, outputFileName);  
  
 try (Scanner scanner = new Scanner(sourceFile);  
 PrintWriter writer = new PrintWriter(outputFile)) {  
 while (scanner.hasNextLine()) {  
 String line = scanner.nextLine();  
 writer.println(processLine(line));  
 }  
 } catch (FileNotFoundException e) {  
 System.out.println("Файл не найден: " + e.getMessage());  
 }  
 }  
  
 private static String processLine(String line) {  
 StringBuilder processedLine = new StringBuilder();  
 StringBuilder wordBuffer = new StringBuilder();  
  
 for (char ch : line.toCharArray()) {  
 if (Character.isLetter(ch)) {  
 wordBuffer.append(ch);  
 } else {  
 if (wordBuffer.length() > 2) {  
 processedLine.append(wordBuffer.toString().toUpperCase());  
 } else {  
 processedLine.append(wordBuffer.toString());  
 }  
 wordBuffer.setLength(0); // Очищаем буфер  
 processedLine.append(ch); // Добавляем не-буквенный символ  
 }  
 }  
  
 // Обработка последнего слова в строке, если оно существует  
 if (wordBuffer.length() > 2) {  
 processedLine.append(wordBuffer.toString().toUpperCase());  
 } else {  
 processedLine.append(wordBuffer.toString());  
 }  
  
 return processedLine.toString();  
 }  
}

Результаты работы программы представлены на рисунке 7.

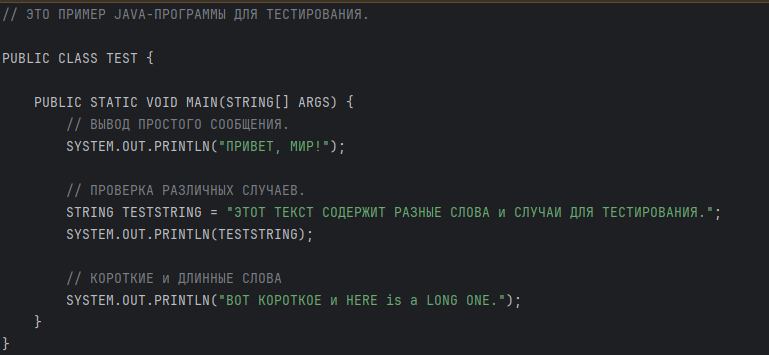


Рисунок 7 – Результаты работы программы

**Задание 8**: при выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File.

В файле, содержащем фамилии студентов и их оценки, записать прописными буквами фамилии тех студентов, которые имеют средний балл более “7”.

Код класса Main:

import java.io.File;  
import java.io.FileNotFoundException;  
import java.io.PrintWriter;  
import java.util.Scanner;  
  
public class Main {  
  
 public static void main(String[] args) {  
 String sourceFilePath = "input.txt";  
 String outputDirPath = "output/";  
 String outputFileName = "output.txt";  
  
 File inputFile = new File(sourceFilePath);  
 File outputDir = new File(outputDirPath);  
  
 // Создание директории для вывода, если она не существует  
 if (!outputDir.exists()) {  
 outputDir.mkdirs();  
 }  
  
 File outputFile = new File(outputDir, outputFileName);  
  
 try (Scanner scanner = new Scanner(inputFile);  
 PrintWriter writer = new PrintWriter(outputFile)) {  
 while (scanner.hasNextLine()) {  
 String line = scanner.nextLine();  
 writer.println(processLine(line));  
 }  
 } catch (FileNotFoundException e) {  
 System.err.println("Ошибка: Файл не найден - " + e.getMessage());  
 }  
 }  
  
 private static String processLine(String line) {  
 String[] parts = line.split(" ");  
 if (parts.length < 2) return line; // Если строка не содержит оценок  
  
 double sum = 0;  
 for (int i = 1; i < parts.length; i++) {  
 sum += Integer.parseInt(parts[i]);  
 }  
 double average = sum / (parts.length - 1);  
  
 if (average > 7) {  
 return parts[0].toUpperCase() + line.substring(parts[0].length());  
 } else {  
 return line;  
 }  
 }  
}

Результат работы программы показан на рисунке 8.

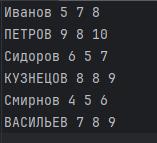


Рисунок 8 – Результат работы программы

**Вывод:** была освоена работа с исключениями и файлами в Java.