|  |  |
| --- | --- |
| 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 **09.04.01/07 Интеллектуальные системы анализа,**

**обработки и интерпретации больших данных.**

**Отчет**

**по лабораторной работе № 5**

**Вариант № 12**

**Название:** исключения в java

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

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

Москва, 2024

**Цель:** изучить 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:

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.Scanner;

public class VectorR3 {

private double x, y, z;

public VectorR3(double x, double y, double z) {

this.x = x;

this.y = y;

this.z = z;

}

// Метод для проверки ортогональности с другим вектором

public boolean isOrthogonal(VectorR3 other) {

return (this.x \* other.x + this.y \* other.y + this.z \* other.z) == 0;

}

// Метод для проверки пересечения с другим неортогональным вектором

public boolean intersects(VectorR3 other) {

return !this.isCollinear(other) && !this.isOrthogonal(other);

}

// Проверка на коллинеарность

private boolean isCollinear(VectorR3 other) {

try {

double kx = this.x / other.x;

double ky = this.y / other.y;

double kz = this.z / other.z;

return (kx == ky) && (ky == kz);

} catch (ArithmeticException e) {

return false;

}

}

// Проверка на компланарность

public static boolean areCoplanar(VectorR3 v1, VectorR3 v2, VectorR3 v3) {

double volume = v1.x \* (v2.y \* v3.z - v3.y \* v2.z) -

v1.y \* (v2.x \* v3.z - v3.x \* v2.z) +

v1.z \* (v2.x \* v3.y - v3.x \* v2.y);

return volume == 0;

}

// Метод для сравнения векторов

@Override

public boolean equals(Object obj) {

if (this == obj) return true;

if (!(obj instanceof VectorR3)) return false;

VectorR3 other = (VectorR3) obj;

return Double.compare(this.x, other.x) == 0 &&

Double.compare(this.y, other.y) == 0 &&

Double.compare(this.z, other.z) == 0;

}

@Override

public String toString() {

return String.format("Vector(%f, %f, %f)", x, y, z);

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

ArrayList<VectorR3> vectors = new ArrayList<>();

try {

System.out.println("Введите количество векторов:");

int n = scanner.nextInt();

for (int i = 0; i < n; i++) {

System.out.println("Введите координаты вектора " + (i + 1) + " (x y z):");

double x = scanner.nextDouble();

double y = scanner.nextDouble();

double z = scanner.nextDouble();

vectors.add(new VectorR3(x, y, z));

}

// Проверяем ортогональность всех пар векторов

for (int i = 0; i < vectors.size(); i++) {

for (int j = i + 1; j < vectors.size(); j++) {

if (vectors.get(i).isOrthogonal(vectors.get(j))) {

System.out.printf("%s ортогонален %s%n", vectors.get(i), vectors.get(j));

}

}

}

// Проверяем компланарность всех троек векторов

for (int i = 0; i < vectors.size(); i++) {

for (int j = i + 1; j < vectors.size(); j++) {

for (int k = j + 1; k < vectors.size(); k++) {

if (VectorR3.areCoplanar(vectors.get(i), vectors.get(j), vectors.get(k))) {

System.out.printf("%s, %s и %s компланарны%n", vectors.get(i), vectors.get(j), vectors.get(k));

}

}

}

}

} catch (InputMismatchException e) {

System.err.println("Ошибка ввода: введите числовые значения.");

} catch (ArithmeticException e) {

System.err.println("Ошибка математической операции: " + e.getMessage());

} catch (NullPointerException e) {

System.err.println("Ошибка: попытка обращения к объекту по нулевой ссылке.");

} catch (Exception e) {

System.err.println("Произошла непредвиденная ошибка: " + e.getMessage());

} finally {

scanner.close();

}

}

}

Работа программы показана на рисунке 1.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 1 – Работа 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:

//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.

//4. 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

import java.util.ArrayList;

import java.util.InputMismatchException;

import java.util.Scanner;

public class Matrix {

private final double[][] data;

private final int n;

public Matrix(int size) {

this.n = size;

data = new double[size][size];

}

public Matrix(double[][] data) {

if (data.length != data[0].length) {

throw new IllegalArgumentException("Matrix must be square.");

}

this.n = data.length;

this.data = new double[n][n];

for (int i = 0; i < n; i++) {

System.arraycopy(data[i], 0, this.data[i], 0, n);

}

}

// Метод для ввода матрицы с клавиатуры

public static Matrix inputMatrix(int size, Scanner scanner) {

double[][] data = new double[size][size];

System.out.println("Введите элементы матрицы:");

for (int i = 0; i < size; i++) {

for (int j = 0; j < size; j++) {

data[i][j] = scanner.nextDouble();

}

}

return new Matrix(data);

}

// Сложение матриц

public Matrix add(Matrix other) {

if (this.n != other.n) {

throw new IllegalArgumentException("Matrices must have the same size to add.");

}

Matrix result = new Matrix(n);

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

result.data[i][j] = this.data[i][j] + other.data[i][j];

}

}

return result;

}

// Вычитание матриц

public Matrix subtract(Matrix other) {

if (this.n != other.n) {

throw new IllegalArgumentException("Matrices must have the same size to subtract.");

}

Matrix result = new Matrix(n);

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

result.data[i][j] = this.data[i][j] - other.data[i][j];

}

}

return result;

}

// Умножение матриц

public Matrix multiply(Matrix other) {

if (this.n != other.n) {

throw new IllegalArgumentException("Matrices must have the same size to multiply.");

}

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.data[i][j] += this.data[i][k] \* other.data[k][j];

}

}

}

return result;

}

// Вычисление первой нормы матрицы

public double firstNorm() {

double maxSum = 0;

for (int i = 0; i < n; i++) {

double sum = 0;

for (int j = 0; j < n; j++) {

sum += Math.abs(data[i][j]);

}

maxSum = Math.max(maxSum, sum);

}

return maxSum;

}

// Вычисление второй нормы матрицы

public double secondNorm() {

double maxSum = 0;

for (int j = 0; j < n; j++) {

double sum = 0;

for (int i = 0; i < n; i++) {

sum += Math.abs(data[i][j]);

}

maxSum = Math.max(maxSum, sum);

}

return maxSum;

}

// Вспомогательный метод для вывода матрицы

public void printMatrix() {

for (int i = 0; i < n; i++) {

for (int j = 0; j < n; j++) {

System.out.printf("%8.2f", data[i][j]);

}

System.out.println();

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.println("Введите количество матриц:");

int numMatrices = scanner.nextInt();

ArrayList<Matrix> matrices = new ArrayList<>();

for (int i = 0; i < numMatrices; i++) {

System.out.printf("Введите размер матрицы %d: ", i + 1);

int size = scanner.nextInt();

Matrix matrix = Matrix.inputMatrix(size, scanner);

matrices.add(matrix);

}

// Инициализация для отслеживания минимальных норм

double minFirstNorm = Double.MAX\_VALUE;

double minSecondNorm = Double.MAX\_VALUE;

Matrix minFirstNormMatrix = null;

Matrix minSecondNormMatrix = null;

// Вычисление минимальных норм для массива матриц

for (Matrix matrix : matrices) {

double firstNorm = matrix.firstNorm();

double secondNorm = matrix.secondNorm();

if (firstNorm < minFirstNorm) {

minFirstNorm = firstNorm;

minFirstNormMatrix = matrix;

}

if (secondNorm < minSecondNorm) {

minSecondNorm = secondNorm;

minSecondNormMatrix = matrix;

}

}

// Вывод матриц с наименьшей первой и второй нормами

System.out.println("Matrix with the smallest first norm:");

if (minFirstNormMatrix != null) {

minFirstNormMatrix.printMatrix();

}

System.out.println("\nMatrix with the smallest second norm:");

if (minSecondNormMatrix != null) {

minSecondNormMatrix.printMatrix();

}

} catch (InputMismatchException e) {

System.err.println("Ошибка ввода: введите числовые значения.");

} catch (IllegalArgumentException e) {

System.err.println("Ошибка: " + e.getMessage());

} catch (Exception e) {

System.err.println("Произошла непредвиденная ошибка: " + e.getMessage());

} finally {

scanner.close();

}

}

Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описание

Рисунок 2 – Работа программы

Работа программы показана на рисунке 2.

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

Код класса Patient:

//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.

import java.util.InputMismatchException;

import java.util.Scanner;

public class Patient {

private int id;

private String lastName;

private String firstName;

private String middleName;

private String address;

private String phone;

private String medicalCardNumber;

private String diagnosis;

// Конструктор

public Patient(int id, String lastName, String firstName, String middleName,

String address, String phone, String medicalCardNumber, String diagnosis) throws InvalidMedicalCardNumberException {

this.id = id;

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.address = address;

this.phone = phone;

setMedicalCardNumber(medicalCardNumber);

this.diagnosis = diagnosis;

}

// Методы set и get для каждого поля

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

public int getId() { return id; }

public void setLastName(String lastName) { this.lastName = lastName; }

public String getLastName() { return lastName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getFirstName() { return firstName; }

public void setMiddleName(String middleName) { this.middleName = middleName; }

public String getMiddleName() { return middleName; }

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

public String getAddress() { return address; }

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

public String getPhone() { return phone; }

public void setMedicalCardNumber(String medicalCardNumber) throws InvalidMedicalCardNumberException {

if (!medicalCardNumber.matches("\\d{6}")) {

throw new InvalidMedicalCardNumberException("Номер медицинской карты должен состоять из 6 цифр.");

}

this.medicalCardNumber = medicalCardNumber;

}

public String getMedicalCardNumber() { return medicalCardNumber; }

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

public String getDiagnosis() { return diagnosis; }

// toString метод для представления объекта в виде строки

@Override

public String toString() {

return "Patient{" +

"id=" + id +

", lastName='" + lastName + '\'' +

", firstName='" + firstName + '\'' +

", middleName='" + middleName + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", medicalCardNumber='" + medicalCardNumber + '\'' +

", diagnosis='" + diagnosis + '\'' +

'}';

}

// Метод для вывода списка пациентов с данным диагнозом

public static void printPatientsWithDiagnosis(Patient[] patients, String diagnosis) {

for (Patient patient : patients) {

if (patient.getDiagnosis().equalsIgnoreCase(diagnosis)) {

System.out.println(patient);

}

}

}

// Метод для вывода списка пациентов, номер медицинской карты которых находится в заданном интервале

public static void printPatientsByMedicalCardRange(Patient[] patients, String minNumber, String maxNumber) {

for (Patient patient : patients) {

if (patient.getMedicalCardNumber().compareTo(minNumber) >= 0 &&

patient.getMedicalCardNumber().compareTo(maxNumber) <= 0) {

System.out.println(patient);

}

}

}

// Метод для ввода пациента с клавиатуры

public static Patient inputPatient(Scanner scanner) throws InputDataException, InvalidMedicalCardNumberException {

try {

System.out.println("Введите ID:");

int id = scanner.nextInt();

scanner.nextLine(); // consume newline

System.out.println("Введите фамилию:");

String lastName = scanner.nextLine();

System.out.println("Введите имя:");

String firstName = scanner.nextLine();

System.out.println("Введите отчество:");

String middleName = scanner.nextLine();

System.out.println("Введите адрес:");

String address = scanner.nextLine();

System.out.println("Введите телефон:");

String phone = scanner.nextLine();

System.out.println("Введите номер медицинской карты (6 цифр):");

String medicalCardNumber = scanner.nextLine();

System.out.println("Введите диагноз:");

String diagnosis = scanner.nextLine();

return new Patient(id, lastName, firstName, middleName, address, phone, medicalCardNumber, diagnosis);

} catch (InputMismatchException e) {

throw new InputDataException("Некорректный ввод данных.");

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.println("Введите количество пациентов:");

int numPatients = scanner.nextInt();

scanner.nextLine(); // consume newline

Patient[] patients = new Patient[numPatients];

for (int i = 0; i < numPatients; i++) {

System.out.printf("Введите данные для пациента %d:\n", i + 1);

patients[i] = inputPatient(scanner);

}

// Вывод пациентов с диагнозом "Грипп"

System.out.println("Пациенты с диагнозом Грипп:");

printPatientsWithDiagnosis(patients, "Грипп");

// Вывод пациентов с номером медицинской карты в заданном интервале

System.out.println("\nПациенты с номером медицинской карты от 123450 до 123459:");

printPatientsByMedicalCardRange(patients, "123450", "123459");

} catch (InputDataException | InvalidMedicalCardNumberException e) {

System.err.println("Ошибка: " + e.getMessage());

} catch (Exception e) {

System.err.println("Произошла непредвиденная ошибка: " + e.getMessage());

} finally {

scanner.close();

}

}

}

Работа программы показана на рисунке 3.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

Рисунок 3 – Работа программы

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

Код класса Abiturient:

//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).

import java.util.Arrays;

import java.util.Comparator;

import java.util.InputMismatchException;

import java.util.Scanner;

public class Abiturient {

private int id;

private String lastName;

private String firstName;

private String middleName;

private String address;

private String phone;

private int[] grades;

// Конструктор

public Abiturient(int id, String lastName, String firstName, String middleName,

String address, String phone, int[] grades) throws InvalidGradeException {

this.id = id;

this.lastName = lastName;

this.firstName = firstName;

this.middleName = middleName;

this.address = address;

this.phone = phone;

setGrades(grades);

}

// Методы set и get

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

public int getId() { return id; }

public void setLastName(String lastName) { this.lastName = lastName; }

public String getLastName() { return lastName; }

public void setFirstName(String firstName) { this.firstName = firstName; }

public String getFirstName() { return firstName; }

public void setMiddleName(String middleName) { this.middleName = middleName; }

public String getMiddleName() { return middleName; }

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

public String getAddress() { return address; }

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

public String getPhone() { return phone; }

public void setGrades(int[] grades) throws InvalidGradeException {

for (int grade : grades) {

if (grade < 1 || grade > 5) {

throw new InvalidGradeException("Оценки должны быть в диапазоне от 1 до 5.");

}

}

this.grades = grades;

}

public int[] getGrades() { return grades; }

// Вспомогательный метод для подсчета среднего балла

public double getAverageGrade() {

return Arrays.stream(grades).average().orElse(0.0);

}

// Переопределение метода toString

@Override

public String toString() {

return "Abiturient{" +

"id=" + id +

", lastName='" + lastName + '\'' +

", firstName='" + firstName + '\'' +

", middleName='" + middleName + '\'' +

", address='" + address + '\'' +

", phone='" + phone + '\'' +

", grades=" + Arrays.toString(grades) +

", average=" + String.format("%.2f", getAverageGrade()) +

'}';

}

// Методы для работы с массивом объектов

public static void printAbiturientsWithBadGrades(Abiturient[] abiturients, int minPassingGrade) {

for (Abiturient abiturient : abiturients) {

for (int grade : abiturient.getGrades()) {

if (grade < minPassingGrade) {

System.out.println(abiturient);

break;

}

}

}

}

public static void printAbiturientsWithAverageAbove(Abiturient[] abiturients, double average) {

for (Abiturient abiturient : abiturients) {

if (abiturient.getAverageGrade() > average) {

System.out.println(abiturient);

}

}

}

public static void printTopNAbiturients(Abiturient[] abiturients, int n) {

Arrays.sort(abiturients, Comparator.comparingDouble(Abiturient::getAverageGrade).reversed());

for (int i = 0; i < Math.min(n, abiturients.length); i++) {

System.out.println(abiturients[i]);

}

}

// Метод для ввода абитуриентов с клавиатуры

public static Abiturient inputAbiturient(Scanner scanner) throws InputDataException, InvalidGradeException {

try {

System.out.println("Введите ID:");

int id = scanner.nextInt();

scanner.nextLine(); // consume newline

System.out.println("Введите фамилию:");

String lastName = scanner.nextLine();

System.out.println("Введите имя:");

String firstName = scanner.nextLine();

System.out.println("Введите отчество:");

String middleName = scanner.nextLine();

System.out.println("Введите адрес:");

String address = scanner.nextLine();

System.out.println("Введите телефон:");

String phone = scanner.nextLine();

System.out.println("Введите оценки (через пробел):");

String[] gradesStr = scanner.nextLine().split(" ");

int[] grades = Arrays.stream(gradesStr).mapToInt(Integer::parseInt).toArray();

return new Abiturient(id, lastName, firstName, middleName, address, phone, grades);

} catch (InputMismatchException e) {

throw new InputDataException("Некорректный ввод данных.");

} catch (NumberFormatException e) {

throw new InputDataException("Оценки должны быть числовыми значениями.");

}

}

public static void main(String[] args) {

Scanner scanner = new Scanner(System.in);

try {

System.out.println("Введите количество абитуриентов:");

int numAbiturients = scanner.nextInt();

scanner.nextLine(); // consume newline

Abiturient[] abiturients = new Abiturient[numAbiturients];

for (int i = 0; i < numAbiturients; i++) {

System.out.printf("Введите данные для абитуриента %d:\n", i + 1);

abiturients[i] = inputAbiturient(scanner);

}

// Вывод абитуриентов с неудовлетворительными оценками

System.out.println("Абитуриенты с неудовлетворительными оценками:");

printAbiturientsWithBadGrades(abiturients, 3);

// Вывод абитуриентов средний балл у которых выше заданного

System.out.println("\nАбитуриенты со средним баллом выше 4.5:");

printAbiturientsWithAverageAbove(abiturients, 4.5);

// Выбор и вывод n абитуриентов с самым высоким средним баллом

System.out.println("\nЛучшие 2 абитуриента по среднему баллу:");

printTopNAbiturients(abiturients, 2);

} catch (InputDataException | InvalidGradeException e) {

System.err.println("Ошибка: " + e.getMessage());

} catch (Exception e) {

System.err.println("Произошла непредвиденная ошибка: " + e.getMessage());

} finally {

scanner.close();

}

}

}

**Вывод:** были изучена работа с исключениями и файлами в java.