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**

О Т Ч Е Т

по лабораторной работе № 5

Название: Исключения, файлы

Дисциплина: Языки программирования для работы с большими данными

Студент ИУ6-22М _____ Т.И. Кадыров
(Группа) (Подпись, дата) (И.О. Фамилия)

Преподаватель _____ П.В. Степанов
(Подпись, дата) (И.О. Фамилия)

Москва, 2024

Цель работы: освоить базовые принципы работы с исключениями и файлами на языке Java.

Вариант: 8.

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 решения приведен в листинге 1.

Листинг 1 — реализация решения

```
public class Complex {
    private double real;
    private double imaginary;

    public Complex() {
        this.real = 0;
        this.imaginary = 0;
    }

    public Complex(double real, double imaginary) {
        this.real = real;
        this.imaginary = imaginary;
    }

    public Complex(Complex other) {
        this.real = other.real;
        this.imaginary = other.imaginary;
    }

    public Complex add(Complex other) {
        return new Complex(this.real + other.real, this.imaginary + other.imaginary);
    }

    public Complex subtract(Complex other) {
        return new Complex(this.real - other.real, this.imaginary - other.imaginary);
    }

    public Complex multiply(Complex other) {
        double real = this.real * other.real - this.imaginary * other.imaginary;
        double imaginary = this.real * other.imaginary + this.imaginary * other.real;
        return new Complex(real, imaginary);
    }

    public Complex divide(Complex other) throws ArithmeticException {
```

```
        double denominator = other.real * other.real + other.imaginary * other.imaginary;
        if (denominator == 0) {
            throw new ArithmeticException("Divide by zero");
        }
        double real = (this.real * other.real + this.imaginary * other.imaginary) / denominator;
        double imaginary = (this.imaginary * other.real - this.real * other.imaginary) / denominator;
        return new Complex(real, imaginary);
    }

    public void set(Complex other) {
        this.real = other.real;
        this.imaginary = other.imaginary;
    }

    public static Complex[] sumOfVectors(Complex[] vector1, Complex[] vector2) {
        if (vector1.length != vector2.length)
            throw new IllegalArgumentException("Vectors must be of the same length");

        Complex[] sum = new Complex[vector1.length];
        for (int i = 0; i < vector1.length; i++) {
            sum[i] = vector1[i].add(vector2[i]);
        }
        return sum;
    }

    public String toString() {
        return this.real + " + " + this.imaginary + "i";
    }
}

public class Main {
    public static void main(String[] args) {
        Complex[] vector1 = new Complex[]{
                new Complex(1, 1), new Complex(1, 1), new Complex(0, 0)
        };
        Complex[] vector2 = new Complex[]{
                new Complex(1, 1), new Complex(2, 2), new Complex(0, 0)
        };
        Complex[] res = Complex.sumOfVectors(vector1, vector2);
        for (int i = 0; i < res.length; i++) {
            System.out.println(res[i]);
        }
        Complex c1 = new Complex(1, 1);
        Complex c2 = new Complex(0, 0);
        try {
            System.out.println(c1.divide(c2));
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

Задание 2: см. задание 1.

Код решения приведен в листинге 2.

Листинг 2 — реализация решения

```
public class QuadraticEquation {
    private double a;
    private double b;
    private double c;

    public QuadraticEquation(double a, double b, double c) throws CustomException {
        if (a == 0) {
            throw new CustomException("A is zero");
        }
        this.a = a;
        this.b = b;
        this.c = c;
    }

    public QuadraticEquation(double a) throws CustomException {
        this(a, 0, 0);
    }

    public double[] findRoots() throws CustomException {
        double discriminant = b * b - 4 * a * c;
        if (discriminant < 0) {
            throw new CustomException("Discriminant is less than zero");
        } else if (discriminant == 0) {
            return new double[]{-b / (2 * a)};
        } else {
            double sqrtDiscriminant = Math.sqrt(discriminant);
            return new double[]{
                    (-b + sqrtDiscriminant) / (2 * a),
                    (-b - sqrtDiscriminant) / (2 * a)
            };
        }
    }

    public double[] findExtremePoint() {
        double x = -b / (2 * a);
        double y = a * x * x + b * x + c;
        return new double[]{x, y};
    }

    public String findIntervalOfIncrease() {
        if (a > 0) {
            return "(-∞, +∞)";
        } else if (a < 0) {
            double extremePoint = findExtremePoint()[0];
            return "(-∞, " + extremePoint + ")";
        } else {
            return "There's no interval of increase or decrease.";
        }
    }
```

```
    public String findIntervalOfDecrease() {
        if (a < 0) {
            return "(-∞, +∞)";
        } else if (a > 0) {
            double extremePoint = findExtremePoint()[0];
            return "(" + extremePoint + ", +∞)";
        } else {
            return "There's no interval of increase or decrease.";
        }
    }
}

public class CustomException extends Exception {
    public CustomException(String message) {
        super(message);
    }
}

public class Main {
    public static void main(String[] args) {
        QuadraticEquation[] equations = new QuadraticEquation[3];
        try {
            equations[0] = new QuadraticEquation(1, -3, 2);
            equations[1] = new QuadraticEquation(1, 4, 4);
            equations[2] = new QuadraticEquation(1, -6, 9);
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
            return;
        }
        double minRoot, maxRoot;
        try {
            minRoot = equations[0].findRoots()[0];
            maxRoot = equations[0].findRoots()[0];
        } catch (Exception ex) {
            System.out.println(ex.getMessage());
            return;
        }
        for (QuadraticEquation equation : equations) {
            try {
                double[] roots = equation.findRoots();
                for (double root : roots) {
                    if (!Double.isNaN(root)) {
                        if (root < minRoot) {
                            minRoot = root;
                        }
                        if (root > maxRoot) {
                            maxRoot = root;
                        }
                    }
                }
            } catch (Exception ex) {
                System.out.println(ex.getMessage());
            }
        }
```

```
        System.out.println("Minimum root: " + minRoot);
        System.out.println("Maximum root: " + maxRoot);
    }
}
```

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

Код решения приведен в листинге 3.

Листинг 3 — реализация решения

```
package transport;

import java.time.LocalDate;
import java.time.temporal.ChronoUnit;
import java.util.ArrayList;

public class Car {
    private int id;
    private String brand;
    private String model;
    private int year;
    private String color;
    private double price;
    private String registrationNumber;

    public int getId() {
        return id;
    }

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

    public String getBrand() {
        return brand;
    }

    public void setBrand(String brand) {
        this.brand = brand;
    }

    public String getModel() {
        return model;
    }

    public void setModel(String model) {
        this.model = model;
    }

    public int getYear() {
```

```
    return year;
  }

  public void setYear(int year) {
    this.year = year;
  }

  public String getColor() {
    return color;
  }

  public void setColor(String color) {
    this.color = color;
  }

  public double getPrice() {
    return price;
  }

  public void setPrice(double price) {
    this.price = price;
  }

  public String getRegistrationNumber() {
    return registrationNumber;
  }

  public void setRegistrationNumber(String registrationNumber) {
    this.registrationNumber = registrationNumber;
  }

  public Car(int id, String brand, String model, int year, String color, double price, String registrationNumber) {
    this.id = id;
    this.brand = brand;
    this.model = model;
    this.year = year;
    this.color = color;
    this.price = price;
    this.registrationNumber = registrationNumber;
  }

  public String toString() {
    return "Car ID: " + id + ", Brand: " + brand + ", Model: " + model + ", Year: " + year + ", Color: " + color + ", Price: " +
price + ", Registration Number: " + registrationNumber;
  }

  public long getYearsInUse() {
    int currentYear = LocalDate.now().getYear();
    return ChronoUnit.YEARS.between(LocalDate.of(year, 1, 1), LocalDate.of(currentYear, 1, 1));
  }
}

package transport;

import java.util.ArrayList;
```

```
public class CarManager {
  private ArrayList<Car> cars = new ArrayList<>();

  public void addCar(Car car) {
    cars.add(car);
  }

  public ArrayList<Car> getCarsByBrand(String brand) throws LongBrandNameException, CarNotFoundException {
    if (brand.length() > 30) {
      throw new LongBrandNameException();
    }
    ArrayList<Car> carsByBrand = new ArrayList<>();
    for (Car car : cars) {
      if (car.getBrand().equalsIgnoreCase(brand)) {
        carsByBrand.add(car);
      }
    }
    if (carsByBrand.isEmpty()) {
      throw new CarNotFoundException();
    }
    return carsByBrand;
  }

  public ArrayList<Car> getCarsByModelAndYears(String model, int years) throws CarNotFoundException {
    ArrayList<Car> carsByModelAndYears = new ArrayList<>();
    for (Car car : cars) {
      if (car.getModel().equalsIgnoreCase(model) && car.getYearsInUse() > years) {
        carsByModelAndYears.add(car);
      }
    }
    if (carsByModelAndYears.isEmpty()) {
      throw new CarNotFoundException();
    }
    return carsByModelAndYears;
  }

  public ArrayList<Car> getCarsByYearAndPrice(int year, double price) {
    ArrayList<Car> carsByYearAndPrice = new ArrayList<>();
    for (Car car : cars) {
      if (car.getYear() == year && car.getPrice() > price) {
        carsByYearAndPrice.add(car);
      }
    }
    return carsByYearAndPrice;
  }
}

package transport;

public class CarNotFoundException extends Exception {
  public CarNotFoundException(String message) {
    super(message);
  }
   public CarNotFoundException() {
```

```
    super("Car is not found");
  }
}

package transport;

public class LongBrandNameException extends Exception {
  public LongBrandNameException(String message) {
    super(message);
  }

  public LongBrandNameException() {
    super("Brand name is too long");
  }
}

import transport.Car;
import transport.CarManager;
import transport.CarNotFoundException;

import java.util.ArrayList;

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

    CarManager carManager = new CarManager();
    carManager.addCar(new Car(1, "Toyota", "Corolla", 2010, "Red", 15000, "1234AB"));
    carManager.addCar(new Car(2, "Honda", "Civic", 2015, "Blue", 20000, "5678CD"));
    carManager.addCar(new Car(3, "Toyota", "Camry", 2018, "White", 25000, "9012EF"));

    try {
      ArrayList<Car> toyotaCars = carManager.getCarsByBrand("Toyotaaa");
      System.out.println("Cars by brand Toyota:");
      for (Car car : toyotaCars) {
        System.out.println(car);
      }
    } catch (Exception ex) {
      System.out.println(ex.getMessage());
    }

    try {
      ArrayList<Car> oldCivicCars = carManager.getCarsByModelAndYears("Civic", 5);
      System.out.println("Older than 5 years Civic cars:");
      for (Car car : oldCivicCars) {
        System.out.println(car);
      }
    } catch (CarNotFoundException ex) {
      System.out.println(ex.getMessage());
    }

    ArrayList<Car> expensive2018Cars = carManager.getCarsByYearAndPrice(2018, 20000);
    System.out.println("Expensive 2018 cars:");
    for (Car car : expensive2018Cars) {
      System.out.println(car);
    }
```

```
    }
}
```

Задание 4: см. задание 3.

Код решения приведен в листинге 4.

Листинг 4 — реализация решения

```
package product;

import java.util.ArrayList;

public class Product {
    private int id;
    private String name;
    private String UPC;
    private String manufacturer;
    private double price;
    private int shelfLife;
    private int quantity;

    public Product(int id, String name, String UPC, String manufacturer, double price, int shelfLife, int quantity) {
        this.id = id;
        this.name = name;
        this.UPC = UPC;
        this.manufacturer = manufacturer;
        this.price = price;
        this.shelfLife = shelfLife;
        this.quantity = quantity;
    }

    public int getId() {
        return id;
    }

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

    public String getName() {
        return name;
    }

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

    public String getUPC() {
        return UPC;
    }

    public void setUPC(String UPC) {
        this.UPC = UPC;
```

```
    }

    public String getManufacturer() {
        return manufacturer;
    }

    public void setManufacturer(String manufacturer) {
        this.manufacturer = manufacturer;
    }

    public double getPrice() {
        return price;
    }

    public void setPrice(double price) {
        this.price = price;
    }

    public int getShelfLife() {
        return shelfLife;
    }

    public void setShelfLife(int shelfLife) {
        this.shelfLife = shelfLife;
    }

    public int getQuantity() {
        return quantity;
    }

    public void setQuantity(int quantity) {
        this.quantity = quantity;
    }

    public String toString() {
        return "Product ID: " + id + ", Name: " + name + ", UPC: " + UPC + ", Manufacturer: " + manufacturer + ", Price: " + price
+ ", Shelf Life: " + shelfLife + ", Quantity: " + quantity;
    }
}

package product;

import java.util.ArrayList;

public class ProductManager {
    private ArrayList<Product> products = new ArrayList<>();

    public void addProduct(Product product) {
        products.add(product);
    }

    public ArrayList<Product> getProductsByName(String name) {
        ArrayList<Product> productsByName = new ArrayList<>();
        for (Product product : products) {
            if (product.getName().equalsIgnoreCase(name)) {
```

```
            productsByName.add(product);
        }
    }
    return productsByName;
}

public ArrayList<Product> getProductsByNameAndPrice(String name, double maxPrice) throws ProductNotFoundException{
    ArrayList<Product> productsByNameAndPrice = new ArrayList<>();
    for (Product product : products) {
        if (product.getName().equalsIgnoreCase(name) && product.getPrice() <= maxPrice) {
            productsByNameAndPrice.add(product);
        }
    }
    if (productsByNameAndPrice.isEmpty()) {
        throw new ProductNotFoundException();
    }
    return productsByNameAndPrice;
}

public ArrayList<Product> getProductsByShelfLife(int minShelfLife) throws ProductNotFoundException {
    ArrayList<Product> productsByShelfLife = new ArrayList<>();
    for (Product product : products) {
        if (product.getShelfLife() > minShelfLife) {
            productsByShelfLife.add(product);
        }
    }
    if (productsByShelfLife.isEmpty()) {
        throw new ProductNotFoundException();
    }
    return productsByShelfLife;
}
}

package product;

public class ProductNotFoundException extends Exception {
    public ProductNotFoundException(String message) {
        super(message);
    }
    public ProductNotFoundException() {
        super("Product not found");
    }
}

import product.Product;
import product.ProductManager;
import product.ProductNotFoundException;

import java.util.ArrayList;

public class Main {
    public static void main(String[] args) {
        ProductManager productManager = new ProductManager();
        productManager.addProduct(new Product(1, "Milk", "123456789012", "Farmers Inc.", 2.50, 7, 10));
        productManager.addProduct(new Product(2, "Bread", "234567890123", "Bakery Ltd.", 1.50, 5, 20));
```

```
        productManager.addProduct(new Product(3, "Eggs", "345678901234", "Eggcellent Farms", 3.00, 10, 15));

        ArrayList<Product> milkProducts = productManager.getProductsByName("Milk");
        System.out.println("Products with name Milk:");
        for (Product product : milkProducts) {
            System.out.println(product);
        }

        try {
            ArrayList<Product> cheapBreadProducts = productManager.getProductsByNameAndPrice("Bread", 2.00);
            System.out.println("Cheap Bread products:");
            for (Product product : cheapBreadProducts) {
                System.out.println(product);
            }
        } catch (ProductNotFoundException ex) {
            System.out.println(ex.getMessage());
        }

        try {
            ArrayList<Product> longShelfLifeProducts = productManager.getProductsByShelfLife(6);
            System.out.println("Products with shelf life longer than 6 days:");
            for (Product product : longShelfLifeProducts) {
                System.out.println(product);
            }
        } catch (ProductNotFoundException ex) {
            System.out.println(ex.getMessage());
        }
    }
}
```

Задание 5: Определить частоту повторяемости букв и слов в стихотворении Александра Пушкина.

Код решения приведен в листинге 5.

Листинг 5 — реализация решения

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.util.HashMap;
import java.util.Map;

public class Main {
    public static void main(String[] args) {
        if (args.length != 1) {
            System.err.println("Usage: java Main <inputFilePath>");
            System.exit(1);
        }

        String filePath = args[0];
        Map<Character, Integer> letterFrequency = new HashMap<>();
        Map<String, Integer> wordFrequency = new HashMap<>();
```

```
    try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
      String line;
      while ((line = reader.readLine()) != null) {
        line = line.toLowerCase();
        String[] words = line.split("\\s+");
        for (String word : words) {
          wordFrequency.put(word, wordFrequency.getOrDefault(word, 0) + 1);
          for (char letter : word.toCharArray()) {
            if (Character.isLetter(letter)) {
              letterFrequency.put(letter, letterFrequency.getOrDefault(letter, 0) + 1);
            }
          }
        }
      }
    } catch (IOException e) {
      System.err.println("Error reading file: " + e.getMessage());
      System.exit(1);
    }

    System.out.println("Частота повторяемости букв:");
    for (Map.Entry<Character, Integer> entry : letterFrequency.entrySet()) {
      System.out.println(entry.getKey() + ": " + entry.getValue());
    }

    System.out.println("\nЧастота повторяемости слов:");
    for (Map.Entry<String, Integer> entry : wordFrequency.entrySet()) {
      System.out.println(entry.getKey() + ": " + entry.getValue());
    }
  }
}
```

Задание 6: Входной файл содержит совокупность строк. Строка файла содержит строку квадратной матрицы. Ввести матрицу в двумерный массив (размер матрицы найти). Вывести исходную матрицу и результат ее транспонирования.

Код решения приведен в листинге 6.

Листинг 6 — реализация решения

```
import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;

public class Main {
  public static void main(String[] args) {
    if (args.length != 1) {
      System.exit(1);
    }

    String filePath = args[0];
```

```
try (BufferedReader reader = new BufferedReader(new FileReader(filePath))) {
    String firstLine = reader.readLine();
    if (firstLine == null) {
        System.err.println("Empty file");
        return;
    }

    String[] elements = firstLine.split("\\s+");
    int rows = elements.length;
    int columns = 1;
    for (int i = 0; i < elements.length; i++) {
        if (elements[i].length() > 1) {
            System.err.println("Invalid matrix format");
            return;
        }
    }

    char[][] matrix = new char[rows][rows];
    for (int i = 0; i < rows; i++) {
        if (i != 0) {
            String line = reader.readLine();
            if (line == null) {
                System.err.println("Invalid number of rows");
                return;
            }
            String[] rowElements = line.split("\\s+");
            if (rowElements.length != rows) {
                System.err.println("Invalid number of elements in row " + (i + 1));
                return;
            }
            for (int j = 0; j < rows; j++) {
                if (rowElements[j].length() != 1) {
                    System.err.println("Invalid matrix format");
                    return;
                }
                matrix[i][j] = rowElements[j].charAt(0);
            }
        } else {
            for (int j = 0; j < rows; j++) {
                matrix[i][j] = elements[j].charAt(0);
            }
        }
    }

    System.out.println("Исходная матрица:");
    printMatrix(matrix);

    char[][] transposedMatrix = transposeMatrix(matrix);

    System.out.println("\nТранспонированная матрица:");
    printMatrix(transposedMatrix);

} catch (IOException e) {
    System.err.println("Error reading file: " + e.getMessage());
```

```
    }
  }

  private static void printMatrix(char[][] matrix) {
    for (char[] row : matrix) {
      for (char element : row) {
        System.out.print(element + " ");
      }
      System.out.println();
    }
  }

  private static char[][] transposeMatrix(char[][] matrix) {
    int rows = matrix.length;
    int columns = matrix[0].length;
    char[][] transposedMatrix = new char[columns][rows];
    for (int i = 0; i < rows; i++) {
      for (int j = 0; j < columns; j++) {
        transposedMatrix[j][i] = matrix[i][j];
      }
    }
    return transposedMatrix;
  }
}
```

Задание 7: Из текста Java-программы удалить все виды комментариев.

Код решения приведен в листинге 7.

Листинг 7 — реализация решения

```
import java.io.*;

/* Многострочный комментарий */
public class Main {
  public static void main(String[] args) {
    String outputDirectoryPath = "output";
    File outputDirectory = new File(outputDirectoryPath);
    if (!outputDirectory.exists()) {
      if (outputDirectory.mkdir()) {
        System.out.println("Директория для результатов создана: " + outputDirectory.getAbsolutePath());
      } else {
        System.err.println("Ошибка: Не удалось создать директорию для результатов");
        return;
      }
    }

    String sourceFilePath = "src/Main.java";
    String targetFilePath = outputDirectoryPath + "/MainWithoutComments.java";
    try {
      if (!new File(targetFilePath).createNewFile()) {
        return;
      }
    } catch (Exception ex) {
```

```
            System.out.println(ex.getMessage());
        }
        try (BufferedReader reader = new BufferedReader(new FileReader(sourceFilePath));
             BufferedWriter writer = new BufferedWriter(new FileWriter(targetFilePath))) {

            String line;
            while ((line = reader.readLine()) != null) {
                line = removeComments(line);
                writer.write(line);
                writer.newLine();
            }

            System.out.println("Файл без комментариев успешно создан: " + targetFilePath);

        } catch (IOException e) {
            System.err.println("Ошибка: " + e.getMessage());
        }
    }

    // Метод для удаления комментариев из строки
    private static String removeComments(String line) {
        line = line.replaceAll("//.*", "");
        line = line.replaceAll("/\\*.*?\\*/", "");
        return line;
    }
}
```

Задание 8: Прочитать строки из файла и поменять местами первое и последнее слова в каждой строке.

Код решения приведен в листинге 8.

Листинг 8 — реализация решения

```
import java.io.*;

public class Main {
    public static void main(String[] args) {
        String outputDirectoryPath = "output";
        File outputDirectory = new File(outputDirectoryPath);
        if (!outputDirectory.exists()) {
            if (outputDirectory.mkdir()) {
                System.out.println("Директория для результатов создана: " + outputDirectory.getAbsolutePath());
            } else {
                System.err.println("Ошибка: Не удалось создать директорию для результатов");
                return;
            }
        }

        File targetFile = new File(outputDirectory, "/output.txt");
        try {
            if (targetFile.createNewFile()) {
                System.out.println("Файл успешно создан: " + targetFile.getAbsolutePath());
```

```
        } else {
            System.err.println("Файл уже существует: " + targetFile.getAbsolutePath());
        }
    } catch (IOException e) {
        System.err.println("Ошибка при создании файла: " + e.getMessage());
        return;
    }

    String sourceFilePath = "static/example.txt";

    try (BufferedReader reader = new BufferedReader(new FileReader(sourceFilePath));
         BufferedWriter writer = new BufferedWriter(new FileWriter(targetFile))) {

        String line;
        while ((line = reader.readLine()) != null) {
            String[] words = line.split("\\s+");
            if (words.length >= 2) {
                String firstWord = words[0];
                words[0] = words[words.length - 1];
                words[words.length - 1] = firstWord;
            }
            writer.write(String.join(" ", words));
            writer.newLine();
        }

    } catch (IOException e) {
        System.err.println("Ошибка: " + e.getMessage());
    }
  }
}
```

Вывод: в ходе лабораторной работы были освоены базовые принципы работы с исключениями и файлами на языке Java.