МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАГИИ
Ордена Трудового Красного Знамени федеральное государственное бюджетное учреждение высшего образования «Московский технический университет связи и информатики»

Кафедра «Информационные технологии»
Предмет «Математические Основы Баз Данных»

Лабораторная работа №4

Обработка исключений

Вариант 22

Выполнил:
студент гр. БПИ2402
Поляков Н.А.

Москва
2025

Содержание

1 Цель работы

Научиться работать с ошибками в ходе выполнения кода. Освоить для этого оператор try-catch и все его особенности.

2 Индивидуальное задание

Задания для выполнения лабораторной работы

Задание 1.

Написать программу, которая будет находить среднее арифметическое элементов массива. При этом программа должна обрабатывать ошибки, связанные с выходом за границы массива и неверными данными (например, если элемент массива не является числом). Пример реализации метода:

Задание 2.

Написать программу, которая будет копировать содержимое одного файла в другой. При этом программа должна обрабатывать возможные ошибки, связанные:

- с открытием и закрытием файлов (вариант 1);
- чтением и записью файлов (вариант 2).

Задание 3.

Создайте Java-проект для работы с исключениями. Для каждой из восьми указанных ниже задач напишите собственный класс для обработки исключений. *Создайте обработчик исключений, который логирует информацию о каждом выброшенном исключении в текстовый файл.*

6. Создайте класс CustomInputMismatchException, который будет использоваться для обработки исключения InputMismatchException. Напишите программу, которая считывает целое число с клавиатуры, и, если ввод пользователя не является числом, выбрасывайте исключение CustomInputMismatchException.

3 Выполнение

3.1 Задание 1

Класс CustomDivisionException наследует класс Exception. Для этого простого задания достаточно лишь написать в нём конструктор. Если бы мы хотели более продвинутую работу с ошибками, в этом классе можно было бы, например, хранить величину делимого.

```
1 class CustomDivisionException extends Exception {
2   public CustomDivisionException(String message) {
3     System.out.println("Деление на ноль!");
4     super(message);
5   }
6 }
7
```

Рис. 1

Использование нового класса тоже элементарно: участок кода, где может произойти деление на ноль, нужно засунуть в блок try-catch и обработать данную ошибку. Вот полная программа:

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

public class Main {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Введите два числа: ");

    double input[] = Arrays.stream(scanner.nextLine().split(" "))
                           .map(String::trim)
                           .mapToDouble(Double::parseDouble)
                           .toArray();

    double result = 0;

    try {
      if (input[1] == 0)
        throw new CustomDivisionException("");
      else {
        result = input[0] / input[1];
        System.out.printf("Ответ: %f\n", result);
      }
    } catch (CustomDivisionException e) {
      System.out.println("Нельзя делить на ноль!");
    } catch (Exception e) {
      System.out.println("Ошибка ввода.");
    }

    scanner.close();
  }
}
```

Рис. 2

3.2 Задание 2

Для начала создадим классы для кастомных ошибок:

```java
import java.io.*;
import java.nio.file.*;

public class Program {
  static class FileOpeningException extends Exception {
    public FileOpeningException(String filename) {
      super("Не удалось открыть файл " + filename);
    }
  }

  static class FileWritingException extends Exception {
    public FileWritingException(String filename) {
      super("Не удалось записать в файл " + filename);
    }
  }
```

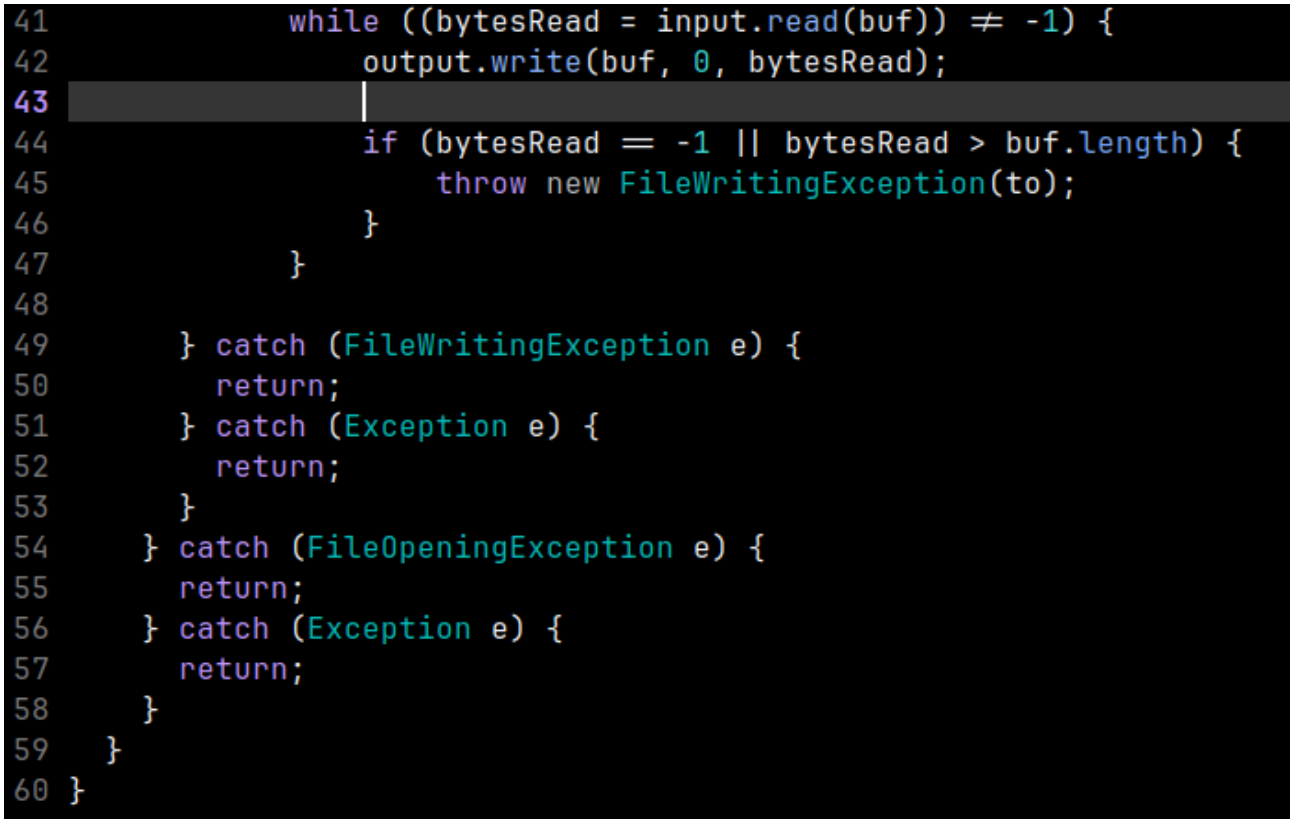
Рис. 3

Далее копируем файл в блоке try, где именно будет обрабатывать ошибки, связанные с открытием:

```java
  public static void main(String[] args) {
    if (args.length != 2) {
      System.out.println("Usage: java Main.java <file to copy FROM> <file to copy IN>");
      return;
    }
    String from = args[0];
    String to = args[1];

    try {
      File source = new File(from);
      if (!source.exists() || !source.isFile())
        throw new FileOpeningException(from);

      if (from.equals(to))
        throw new Exception("Вы пытаетесь скопировать файл в него самого!");

      File copy = new File(to);

      try (FileInputStream input = new FileInputStream(source);
           FileOutputStream output = new FileOutputStream(copy)) {

        byte[] buf = new byte[8192];
            int bytesRead;

            while ((bytesRead = input.read(buf)) != -1) {
                output.write(buf, 0, bytesRead);
```

Рис. 4

В этом же блоке создаём ещё один try, только теперь обрабатываем ошибки, связанные с записью:



```
41              while ((bytesRead = input.read(buf)) != -1) {
42                  output.write(buf, 0, bytesRead);
43                  
44                  if (bytesRead == -1 || bytesRead > buf.length) {
45                      throw new FileWritingException(to);
46                  }
47              }
48
49        } catch (FileWritingException e) {
50          return;
51        } catch (Exception e) {
52          return;
53        }
54      } catch (FileOpeningException e) {
55        return;
56      } catch (Exception e) {
57        return;
58      }
59    }
60 }
```

Рис. 5

3.3 Задание 3

Сначала создаём кастомный класс для ошибки неправильного ввода. В нём будем хранить сообщение об ошибке и сам неправильный ввод:

```java
import java.util.InputMismatchException;

public class CustomInputMismatchException extends InputMismatchException {
  String MismatchedInput;
  public CustomInputMismatchException(String MismatchedInput) {
    super(MismatchedInput + " не является числом!");
    this.MismatchedInput = MismatchedInput;
  }
}
```

Рис. 6

Вот так он работает:

```java
import java.util.Scanner;

public class Program {
  public static void main(String[] args) {
    Scanner scanner = new Scanner(System.in);
    System.out.println("Введите любое число: ");

    try {
      if (!scanner.hasNextInt())
        throw new CustomInputMismatchException(scanner.nextLine());
      else {
        int number = scanner.nextInt();
        System.out.printf("%d действительно является числом!\n", number);
      }
    } catch (CustomInputMismatchException e) {
      scanner.close();
      System.out.println(e.getMessage());
      return;
    }
    scanner.close();
  }
}
```

Рис. 7

4 Вывод

Я научился обрабатывать ошибки на языке Java, а также создавать свои собственные классы ошибок.

5 Github

https://github.com/KLARKOFF/IT-and-Programming-labs