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

Кафедра Математическая кибернетика и информационные технологии

Лабораторная работа №4

Обработка исключений

по дисциплине
«Информационные технологии и программирование»

Выполнил: студент гр. БВТ2403
Толибов Р.Ш.

Руководитель:

Москва, 2025 г.

Цель работы:
Научиться работать с исключениями в Java, понять работу try-catch, try-with-resources, научиться создавать собственные исключения

Ход работы:

Задание 1.
Было необходимо написать программу, которая будет находить среднее арифметическое число элементов массива и обрабатывать возможные ошибки.

```
public class ArrayAverage {
    public static void main(String[] args) {
        Object[] arr = {1, 2, "три", 5, 6};
        int sum = 0;
        int count = 0;

        try {
            for (int i = 0; i <= arr.length; i++) {
                if (arr[i] instanceof Integer) {
                    sum += (Integer) arr[i];
                    count++;
                } else {
                    throw new NumberFormatException("Элемент массива не
является числом: " + "'" + arr[i] + "'");
                }
            }
            double average = (double) sum / count;
            System.out.println("Среднее арифметическое элементов массива: " +
average);

        } catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Ошибка: выход за границы массива.");
        } catch (NumberFormatException e) {
            System.out.println("Ошибка: неверный формат данных. " +
e.getMessage());
        }
    }
}
```

Здесь мы создаем массив типа Object (для работы с разными типами, чтобы можно было продемонстрировать ошибку). Далее создаем переменные sum, count и блок try-catch, в котором находится цикл, проверяющий, является ли элемент массива экземпляром класса Integer. Если нет, то «выбрасывается» ошибка, которая будет обработана в блоке catch. Также в цикле мы взяли i <=

arr.length, вместо <, для того, чтобы вызвать и продемонстрировать ошибку «ArrayIndexOutOfBoundsException»

Задание 2.

Требовалось создать программу, копирующую содержание одного файла в другой.

```
import java.io.*;

public class FileCopy {
    public static void main(String[] args) {
        try (FileInputStream in = new FileInputStream("input.txt");
             FileOutputStream out = new FileOutputStream("output.txt")) {
            int byteData;
            while ((byteData = in.read()) != -1) {
                out.write(byteData);
            }
            System.out.println("Файл успешно скопирован");
        } catch (FileNotFoundException e) {
            System.out.println("Ошибка: файл не найден");
        } catch (IOException e) {
            System.out.println("Ошибка ввода-вывода: " + e.getMessage());
        }
    }
}
```

Импортируем пакет java.io.* для работы с исключениями. Создаем блок try-with-resources, который будет автоматически закрывать файлы по окончании работы с ними. Объявляю переменную byteData и запускаю цикл: до тех пор, пока переменная byteData не равна -1 (т.е. концу файла), записывать символы по байту из одного файла в другой.

В блоках catch обрабатываются такие ошибки, как FileNotFoundException (файл не найден) и IOException (ошибка ввода-вывода).

Задание 3.

Вариант 1.

Необходимо было создать класс CustomDivisionException, который будет использоваться для обработки исключений при делении на ноль. Также необходимо создать логгер исключений.

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

public class Division {
    public static void main(String[] args) {
        int a = 10;
        int b = 0;

        try {
            double result = divide(a, b);
            System.out.println("Результат деления: " + result);
        } catch (CustomDivisionException e) {
            System.out.println("Ошибка: " + e.getMessage());
            ExceptionLogger.log(e);
        }
    }


    public static double divide(int numerator, int denominator) throws
CustomDivisionException {
        if (denominator == 0) {
            throw new CustomDivisionException("Деление на ноль недопустимо!");
        }
        return (double) numerator / denominator;
    }
}
```

Создадим класс-наследник CustomDivisionException, который является дочерним классом Exception. Объявим class Division, в блоке try-catch распишем логику.

Создадим метод divide, который принимает 2 числа (числитель и знаменатель). Если знаменатель равен 0, то выбрасывается ошибка CustomDivisionException с сообщением «Деление на ноль недопустимо!» и записывается через метод логгера в файл error_log.txt. В ином случае, программа выполнит код без исключений.

Логгер представляет из себя такую программу:

```
import java.io.*;
import java.time.LocalDateTime;

public class ExceptionLogger {
    public static void log(Exception e) {
        try (FileWriter fw = new FileWriter("error_log.txt", true)) {
            fw.write(LocalDateTime.now() + " — " + e.getClass().getName() + ":
" + e.getMessage() + "\n");
        } catch (IOException ex) {
            System.out.println("Ошибка при записи лога: " + ex.getMessage());
        }
    }
}
```

Здесь импортируем 2 пакета, для обработки ошибок и записи времени. Используем try-with-resources, чтобы автоматически закрывать файл после записи лога ошибки. В случае ошибки ввода-вывода, логгер обработает её.

Вывод:

В ходе работы я освоил принципы работы исключений в Java, понял логику блоков try-with-resources и try-catch, создал собственный логгер и исключение.

https://github.com/TRuslan666/ITiP/tree/main/лаб4