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

Отчет по информационным технологиям и программированию

на тему: **«Обработка исключений»**

Выполнил: студент группы БПИ2403

ФИО: Сон Владимир Сергеевич

Руководитель: Рыбаков Егор Дмитриевич

Москва, 2025

Цель работы: Изучить механизм обработки исключений в Java, включая иерархию классов исключений (Throwable, Error, Exception, RuntimeException), различия между проверяемыми и непроверяемыми исключениями, а также освоить способы их объявления и обработки с использованием конструкций try-catch-finally и ключевого слова throws.

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

Задание 2.

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

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.

Ход работы:

```java
class ArrayAverage {
    public static void main(String[] args) {
        int[] arr = {1, 2, 3, 4, 5};
        int sum = 0;
        try {
            for (int i = 0; i <= arr.length; i++) {
                sum += arr[i];
            }

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

```java
import java.io.*;

class FileCopy {
    public static void main(String[] args) {
        try (BufferedReader reader = new BufferedReader(new FileReader("input.txt"));
             BufferedWriter writer = new BufferedWriter(new FileWriter("output.txt"))) {

            String line;
            while ((line = reader.readLine()) != null) {
                writer.write(line);
                writer.newLine();
            }
            System.out.println("Файл скопирован");

        } catch (FileNotFoundException e) {
            System.out.println("Ошибка: файл не найден - " + e.getMessage());
        } catch (IOException e) {
            System.out.println("Ошибка ввода/вывода - " + e.getMessage());
        }
    }
}
```

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

class InputValidator {
    public static int readPositiveInteger(String input) throws CustomInputMismatchException {
        try {
            int value = Integer.parseInt(input);
            if (value <= 0) {
                throw new CustomInputMismatchException(
                    "Введено не положительное число: " + value
                );
            }
            return value;
        } catch (NumberFormatException e) {
            throw new CustomInputMismatchException("Введена не числовая строка: '" + input + "'");
        }
    }

    public static void main(String[] args) {
        try {
            System.out.println("Число: " + readPositiveInteger("42"));
            System.out.println("Число: " + readPositiveInteger("ow;ejkfn"));
        } catch (CustomInputMismatchException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }
        try {
            System.out.println("Число: " + readPositiveInteger("-5"));
        } catch (CustomInputMismatchException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }
    }
}
```

Вывод:

В ходе работы был рассмотрен механизм обработки исключений в Java, который позволяет управлять ошибками и нештатными ситуациями в процессе выполнения программы. Были изучены основные классы иерархии исключений, различия между проверяемыми и непроверяемыми исключениями, а также способы их объявления и обработки. Применение конструкции try-catch-finally и ключевого слова throws способствует созданию более надежного и устойчивого к сбоям кода.