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. Вариант 2

Выполнил: студент группы БПИ2401

Исламов Эмин Маратович

Проверил: Харрасов Камиль Раисович

Москва, 2025

Цель работы

Изучить механизм обработки исключений в Java, научиться использовать конструкцию try-catch-finally, обрабатывать распространённые ошибки, создавать собственные классы исключений и применять их в программах.

Теоретическая часть

Исключения в Java являются механизмом обработки ошибок, возникающих во время выполнения программы.

Все исключения являются потомками класса **Throwable**. От него наследуются:

Error

Ошибки JVM, связанные с серьёзными проблемами:

- OutOfMemoryError
- StackOverflowError

Такие ошибки не рекомендуется обрабатывать — они сигнализируют о критическом сбое.

Exception

Ошибки, возникающие в ходе работы программы:

- деление на ноль
- выход за пределы массива
- ошибка ввода
- проблемы с файлами

Exception делится на:

1) Checked exceptions — проверяемые

Должны быть обработаны или указаны в throws:

- IOException

- FileNotFoundException

2) Unchecked exceptions

Наследники RuntimeException:

- NullPointerException
- ArithmeticException
- ArrayIndexOutOfBoundsException

Конструкция try–catch–finally

```
try {
    // опасный код
} catch (ExceptionType e) {
    // обработка ошибки
} finally {
    // выполняется всегда
}
```

finally используется для закрытия ресурсов, потоков, соединений.

Java также позволяет создавать собственные исключения, наследуясь от Exception или RuntimeException.

ЗАДАНИЕ 1

Написать программу, которая вычисляет среднее арифметическое элементов массива.

Обработать ошибки:

- выход за пределы массива
- неверный ввод пользователя

(например, если элемент не является числом)

```
public 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];
            }
            double average = sum / arr.length;
            System.out.println("Среднее арифметическое:
" + average);
        }
        catch (ArrayIndexOutOfBoundsException e) {
            System.out.println("Ошибка: выход за пределы
массива.");
        }
        catch (Exception e) {
            System.out.println("Ошибка вычислений.");
        }
    }
}
```

- В блоке try выполняется перебор массива.
- Если индекс выходит за пределы массива — ловится ArrayIndexOutOfBoundsException.
- Другие ошибки обрабатываются общим catch (Exception e).

ЗАДАНИЕ 2

Написать программу, копирующую содержимое одного файла в другой.

Необходимо обработать:

1. **обработка ошибок открытия/закрытия файла**
2. **обработка ошибок чтения/записи**

```
import java.io.FileInputStream;
import java.io.FileOutputStream;
import java.io.FileNotFoundException;
import java.io.IOException;

public class FileCopyAllErrors {

    public static void main(String[] args) {

        FileInputStream input = null;
        FileOutputStream output = null;

        try {
            // Попытка открыть файлы
            input = new FileInputStream("input.txt");
            output = new FileOutputStream("output.txt");

            int data;
            // Попытка прочитать и записать данные
```

```
            while ((data = input.read()) != -1) {
                output.write(data);
            }

            System.out.println("Файл успешно скопирован.");

        }
        catch (FileNotFoundException e) {
            // Ошибки открытия файла
            System.out.println("Ошибка: не удалось открыть один из файлов.");
        }
        catch (IOException e) {
            // Ошибка чтения или записи
            System.out.println("Ошибка чтения или записи файла.");
        }
        catch (Exception e) {
            // Ловим всё остальное на всякий случай
            System.out.println("Произошла непредвиденная ошибка.");
        }
        finally {
            // Попытка закрыть файлы
            try {
```

```
                if (input != null) {
                    input.close();
                }
                if (output != null) {
                    output.close();
                }
            }
            catch (IOException e) {
                System.out.println("Ошибка при закрытии
файла.");
            }
        }
    }
}
```

ЗАДАНИЕ 3

Вариант 2 — CustomAgeException

Создать собственное исключение CustomAgeException,

которое выбрасывается при недопустимом возрасте (меньше 0 или больше 120).

Класс собственного исключения

```
public class CustomAgeException extends Exception {

    public CustomAgeException(String message) {
        super(message);
```

```
    }
}
```

Основная программа

```
import java.util.Scanner;

public class AgeChecker {

    public static void checkAge(int age) throws CustomAgeException {
        if (age < 0 || age > 120) {
            throw new CustomAgeException("Недопустимый возраст: " + age);
        }
        System.out.println("Возраст корректный: " + age);
    }

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);
        System.out.print("Введите возраст: ");

        try {
            int age = scanner.nextInt();
            checkAge(age);
```

```
        }
        catch (CustomAgeException e) {
            System.out.println("Ошибка: " + e.getMessage());
        }
        catch (Exception e) {
            System.out.println("Ошибка ввода. Введите число.");
        }
        finally {
            System.out.println("Проверка завершена.");
        }
    }
}
```

Вывод

В ходе лабораторной работы были изучены основные механизмы работы с исключениями в Java. Реализованы три программы:

- программа для вычисления среднего значения массива с обработкой ошибок;
- две программы для копирования файлов с различным уровнем обработки ошибок (открытие файла, чтение/запись, закрытие ресурсов);
- программа с пользовательским исключением CustomAgeException.

Были отработаны конструкции try, catch, finally, механизм проброса исключений через throws, а также создание собственных классов исключений.

Контрольные вопросы

1. Что такое исключение в Java?

Это событие, нарушающее нормальный ход выполнения программы и передающее управление обработчику ошибок.

2. Какие ключевые классы исключений вы знаете?

Throwable, Error, Exception, RuntimeException и их наследники.

3. Что такое проверяемые и непроверяемые исключения?

Checked — обязаны быть обработаны.

Unchecked — возникают во время выполнения, наследники RuntimeException.

4. Какие исключения необходимо обрабатывать?

Ошибки ввода-вывода, работы с файлами, некорректного ввода, логические ошибки.

5. Какие исключения относятся к Error и как их обрабатывать?

StackOverflowError, OutOfMemoryError.

Обычно НЕ обрабатываются.

6. Какие исключения относятся к RuntimeException?

NullPointerException, ArithmeticException, ArrayIndexOutOfBoundsException и др.

7. Как создать собственный класс исключения?

Наследоваться от Exception или RuntimeException, вызвать super(message).

8. Как обрабатываются исключения?

Через конструкции try-catch-finally.

Проброс через throws.

9. Можно ли использовать try без catch или finally?

Нет, должен быть хотя бы один блок — catch или finally.

10. Что произойдёт, если исключение возникнет в блоке finally?

Оно заменит собой любое исключение из try или catch.

11. Как пробросить исключение выше по стеку?

Добавить throws ExceptionName в объявление метода.

12. Разница между finally и try-with-resources?

finally выполняется всегда;

try-with-resources автоматически закрывает ресурсы.

13. Какие классы можно использовать в try-with-resources?

Любые, реализующие интерфейс AutoCloseable.

14. Можно ли в одном try написать несколько catch?

Да. От частных к более общим.

15. В чем разница между throw и throws?

throw — фактически бросает исключение.

throws — объявляет возможность его бросания.

16. Что такое StackOverflowError и OutOfMemoryError?

Ошибки JVM, возникающие из-за переполнения стека или памяти.

Обрабатывать их обычно нельзя / не нужно.