МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ

Ордена Трудового Красного Знамени

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики»

Лабораторная работа № 4

«Исключения»

Выполнила: Студентка группы БВТ2306

Максимова Мария

Москва 2024

GitHub: https://github.com/Mashaaaaa7/Lab4

Задание 1: Необходимо 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).

Задание 2: Необходимо написать программу, которая будет копировать содержимое одного файла в другой. При этом программа должна обрабатывать возможные ошибки, связанные с: Чтением и записью файлов

Задание 3: Создайте Java-проект для работы с исключениями. Для каждой из восьми задач, напишите свой собственный класс для обработки исключений. Создайте обработчик исключений, который логирует информацию о каждом выброшенном исключении в текстовый файл.

Вариант 2: Создайте класс CustomAgeException, который будет использоваться для обработки недопустимых возрастов. Реализуйте программу, которая проверяет возраст пользователя с использованием этого класса, и, если возраст меньше 0 или больше 120, выбрасывайте исключение CustomAgeException.

Задание 1

```
public class ArrayAverage {

    public static double calculateAverage(int[] arr) {
        if (arr == null || arr.length == 0) {
            System.err.println("Массив пуст или равен нулю");
            return Double.NaN;// Возвращаем NaN для пустого массива
        }
        int sum = 0;
        for (int num : arr) {
            sum += num;
        }
        return (double) sum / arr.length;
    }


    public static void main(String[] args) {
        int[] arr = {3,4,5,5};
        double sum;

        try {
            sum = calculateAverage(arr);
            if (Double.isInfinite(sum)) { // Проверка на бесконечность
                System.err.println("Ошибка: деление на ноль.");
            } else {
                System.out.println("Среднее арифметическое " + sum);
            }
        } catch (Exception a) {
            System.err.println("Другая причина ошибки" + a.getMessage());
```

```
        }
    }
}
```

public class ArrayAverage {:

- Объявляет класс с именем ArrayAverage.
- public означает, что класс доступен из любой другой части программы.

public static double calculateAverage(int[] arr) {:

- Объявляет статическую функцию calculateAverage, которая принимает массив целых чисел int[] arr и возвращает результат типа double.
- static означает, что функция принадлежит классу, а не экземпляру класса.
- double — тип данных с плавающей точкой для результата вычисления среднего.

if (arr == null || arr.length == 0) {:

- Проверяет, пуст ли массив arr или равен ли null.
- || — оператор логического “ИЛИ”.

System.err.println("Массив пуст или равен нулю");:

- Выводит сообщение об ошибке в консоль ошибок.

return Double.NaN;: **for (int num : arr) {**:

- Использует цикл for-each для перебора элементов массива arr.
- num — переменная, которая приобретает значение каждого элемента массива в каждой итерации цикла.

sum += num;:

- Прибавляет значение текущего элемента num к сумме sum.
- += — сокращенный оператор прибавления.

return (double) sum / arr.length;:

- Делит сумму sum на количество элементов в массиве arr.length и возвращает результат типа double.
- (double) — явное приведение типа int к double, чтобы получить результат с плавающей точкой.

int[] arr = {3,4,5,5};:

- Создает массив целых чисел arr с значениями 3, 4, 5, 5.

double sum;:

- Объявляет переменную sum типа double для хранения результата вычисления среднего.

try {:

- Начинает блок try, в котором может произойти исключение.

sum = calculateAverage(arr);:

- Вызывает функцию calculateAverage с массивом arr и присваивает результат переменной sum.

if (Double.isInfinite(sum)) {:

- Проверяет, равно ли значение sum бесконечности (Infinity).
- Double.isInfinite(sum) — статический метод класса Double, который проверяет, является ли переданное значение бесконечностью.

System.err.println("Ошибка: деление на ноль.");:

- Выводит сообщение об ошибке в консоль ошибок.

else {:

- Выполняется, если значение sum не равно бесконечности.

System.out.println("Среднее арифметическое " + sum);:

- Выводит результат вычисления среднего в консоль.

catch (Exception a) {:

- Ловит любое исключение (Exception), которое может возникнуть в блоке try.

System.err.println("Другая причина ошибки" + a.getMessage());:

- Выводит сообщение об ошибке в консоль ошибок, включая текст сообщения исключения.

int sum = 0;:

Инициализирует переменную sum значением 0

Задание 2

```
import java.io.FileWriter;
import java.io.IOException;

public class CustomAgeException extends Exception { //Exception-исключение
    public CustomAgeException(String message) {
        super(message);//принимает строку message, которую использует для
сообщения об ошибке.
    }
}

class CheckAge {

    public static void main(String[] args) {
        int age = 1100;

        try {
            checkAge(age);
            System.out.println("Возраст корректен");
        } catch (CustomAgeException e) {
            System.err.println("Ошибка: " + e.getMessage());
            logException(e); // пишет исключения в файл
        }
    }

    private static void checkAge(int age) throws CustomAgeException {
//метод, который проверяет возраст throw-бросить
        if (age < 0 || age > 120) {
            throw new CustomAgeException("Недопустимый возраст: " + age);
        }
    }

    private static void logException(Exception e) {
        try (FileWriter writer = new FileWriter("error.log", true)) { //
открываем файл для записи
            writer.write(e.getMessage() + "\n"); // записываем сообщение об
ошибке
        } catch (IOException ex) {
            System.err.println("Ошибка при записи в файл: " +
ex.getMessage());
        }
    }
}
```

Этот код демонстрирует обработку пользовательского исключения (CustomAgeException) в Java.

1. CustomAgeException класс:

Это пользовательский класс исключения, который наследуется от класса Exception. Он используется для обозначения ситуации, когда возраст находится вне допустимого диапазона. Конструктор принимает строку с

сообщением об ошибке, которая затем передается в конструктор базового класса Exception.

2. CheckAge класс:

Этот класс содержит основной метод main и вспомогательные методы для проверки возраста и логирования исключений.

- **main метод:** В этом методе задается возраст (age), и вызывается метод checkAge() для проверки корректности возраста. Блок try-catch обрабатывает возможное исключение CustomAgeException. Если исключение возникает, выводится сообщение об ошибке, и вызывается метод logException() для записи информации об ошибке в файл.

- **checkAge(int age) метод:** Этот метод проверяет возраст. Если возраст меньше 0 или больше 120, он бросает (throw) исключение CustomAgeException с соответствующим сообщением.

- **logException(Exception e) метод:** Этот метод записывает информацию об исключении в файл “error.log”. Он использует FileWriter для записи в файл, режим true добавляет информацию в конец файла (append), а не перезаписывает его. Блок try-catch обрабатывает возможные ошибки ввода-вывода (IOException) при записи в файл.

В целом:

Код демонстрирует правильный подход к обработке исключений в Java:

- **Пользовательское исключение:** Создается собственный класс исключения для конкретной ситуации (некорректный возраст).
- **Обработка исключений:** Используются блоки try-catch для обработки исключений.
- **Логирование:** Информация об исключении записывается в файл для дальнейшего анализа.
- **try-with-resources:** FileWriter автоматически закрывается в try-with-resources, предотвращая утечки ресурсов.

Задание 3

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

public class readFile {
```

```
    public static void main(String[] args) {

        try (BufferedReader reader = new BufferedReader(new
FileReader("src\\input.txt"))) {

            String line;

            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }

        } catch (FileNotFoundException e) {
            System.out.println("Файл не найден");
        } catch (IOException e) {
            System.err.println("Ошибка: " + e.getMessage());
        }

    }

}
```

Этот код на Java читает и выводит содержимое текстового файла построчно.

1. Импорт библиотек:

- java.io.BufferedReader: Класс для эффективного построчного чтения из потока данных.
- java.io.FileNotFoundException: Класс исключения, которое возникает, если файл не найден.
- java.io.FileReader: Класс для чтения из файла.
- java.io.IOException: Общий класс исключения для ошибок ввода-вывода.

2. main метод:

- **Блок try-with-resources:** try (BufferedReader reader = new BufferedReader(new FileReader("src\\input.txt"))) { ... } Этот блок создает объект BufferedReader, обернутый вокруг FileReader, который открывает файл “src\input.txt”. Важно: try-with-resources автоматически закрывает reader после завершения блока, даже если возникнут исключения. Это предотвращает утечку ресурсов.

- **Чтение файла построчно:** String line; while ((line = reader.readLine()) != null) { ... } Цикл while читает файл построчно. reader.readLine() читает одну строку из файла. Если строка прочитана (line != null), она

выводится на консоль с помощью System.out.println(line). Цикл продолжается до тех пор, пока не будут прочитаны все строки.

- **Обработка исключений:** catch (FileNotFoundException e) { ... } catch (IOException e) { ... } Два блока catch обрабатывают возможные исключения:

    - FileNotFoundException: Если файл “src\input.txt” не найден, выводится сообщение “Файл не найден”.
    - IOException: Если возникают другие ошибки ввода-вывода (например, ошибка доступа к файлу), выводится сообщение “Ошибка: ” с подробным описанием ошибки из e.getMessage().

В целом: Код представляет собой простой, но надежный способ чтения и вывода содержимого текстового файла. Использование try-with-resources является хорошей практикой, обеспечивающей автоматическое закрытие ресурсов и предотвращение утечек. Обработка исключений делает код более устойчивым к ошибкам.