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

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());
        }
    }
}
```

1. **public class ArrayAverage {**:
    - Объявляет класс с именем ArrayAverage.
    - public означает, что класс доступен из любой другой части программы.
2. **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.
3. **if (arr == null || arr.length == 0) {**:
    - Проверяет, пуст ли массив arr или равен ли null.
    - || — оператор логического “ИЛИ”.
4. **System.err.println("Массив пуст или равен нулю");**:
    - Выводит сообщение об ошибке в консоль ошибок.
5. **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.
6. **sum += num;**:
    - Прибавляет значение текущего элемента num к сумме sum.
    - += — сокращенный оператор прибавления.
7. **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.
8. **public static void main(String[] args) {**:

- Объявляет главную функцию main, точка входа в программу.

- public — означает, что функция доступна из любой другой части программы.
- static — означает, что функция принадлежит классу, а не экземпляру класса.
- void — означает, что функция ничего не возвращает.
- String[] args — массив строк, передаваемый в функцию в качестве аргументов командной строки.

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

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

12. **double sum;**:

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

13. **try {**:

- Начинает блок try, в котором может произойти исключение.

14. **sum = calculateAverage(arr);**:

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

15. **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.

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

- Выводит сообщение об ошибке в консоль ошибок.

17. **else {**:

- Выполняется, если значение sum не равно бесконечности.

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

- Выводит результат вычисления среднего в консоль.

19. **catch (Exception a) {**:

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

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

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

21. **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 = 1;

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

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

    private static void logException(Exception e) { //метод записи исключения
        try (FileWriter fileWriter = new FileWriter("error.log", true)) { //
true-данные будут дописываться в конец файла (если файл уже существует).
            StringBuilder sb = new StringBuilder();
            sb.append("Исключение: ").append(e.getClass().getSimpleName());
//Формируется строка с информацией об исключении (тип и сообщение).
            sb.append(" - ").append(e.getMessage()).append("\n");
            fileWriter.write(sb.toString());
        } catch (IOException ex) { //обрабатывается исключение IOException,
которое может возникнуть при записи в файл.
            System.err.println("Ошибка при записи в файл: " +
ex.getMessage());
```

```
        }
    }
}
```

1. Объявление пользовательского исключения:

- CustomAgeException — наш собственный класс исключения, который наследуется от класса Exception.
- Конструктор принимает строку message, которую использует для сообщения об ошибке.

2. Класс CheckAge:

- В главном методе main инициализируется переменная age со значением 1.
- Используется блок try-catch для обработки возможных исключений.
- try: Вызывает метод checkAge с переданным возрастом.
- catch (CustomAgeException e): Если метод checkAge бросает исключение CustomAgeException, оно ловится здесь. В блоке catch выводится сообщение об ошибке и вызывается метод logException для записи исключения в файл.

3. Метод checkAge:

- private static void checkAge(int age) throws CustomAgeException — метод, который проверяет возраст.
- Если возраст меньше 0 или больше 120, метод бросает исключение CustomAgeException, передавая сообщение об ошибке.

4. Метод logException:

- private static void logException(Exception e) — метод, который записывает исключение в файл error.log.
- Используется FileWriter для записи в файл.
- true во втором аргументе FileWriter означает, что данные будут дописываться в конец файла (если файл уже существует).
- Формируется строка с информацией об исключении (тип и сообщение).
- В блоке catch обрабатывается исключение IOException, которое может возникнуть при записи в файл.

Задание 3

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

```
public class readFile {

    static final int BUFFER_SIZE = 63;

    public static void main(String[] args) {

        try (BufferedReader reader = new BufferedReader(new
FileReader("src/readfile"))) {
            System.out.println("---Начало файла---");
            String line;
            while ((line = reader.readLine()) != null) {
                System.out.println(line);
            }
            System.out.println("---Конец файла---");
        } catch (FileNotFoundException e) {
            System.out.println("Файл не найден");
        } catch (IOException e) {
            System.err.println("Ошибка чтения файла: " + e.getMessage());
        }
    }
}
```

Этот код:

1. Открывает файл для чтения.
2. Читает файл по строкам и выводит их на консоль.
3. Обрабатывает ошибки, которые могут возникнуть при чтении файла.

1. Импорт необходимых классов:

- **BufferedReader**: Этот класс обеспечивает более эффективное чтение файла, буферизуя данные, чтобы не читать файл по одному символу за раз.
- **FileNotFoundException**:Это исключение, которое возникает, если файл не найден.
- **FileReader**: Класс, который используется для чтения данных из файла.
- **IOException**: Общее исключение, которое может возникнуть при работе с вводом/выводом, в том числе при чтении файлов.

2. Определение класса readFile:

```
public class readFile {

   // ... код ...

}
```

- **public**: Класс доступен из любой другой части программы.
- **class readFile**: Определяется класс с именем readFile.

3. Объявление константы BUFFER_SIZE:

```
static final int BUFFER_SIZE = 63;
```

- **static**: Переменная принадлежит классу, а не экземпляру класса.
- **final**: Значение переменной не может быть изменено после инициализации.
- BUFFER_SIZE — в этом коде она не используется, ее можно удалить.

4. Главный метод main:

```
public static void main(String[] args) {
    // ... код ...
}
```

- **public static void main(String[] args)**: Точка входа в программу.

5. Чтение файла:

```
try (BufferedReader reader = new BufferedReader(new
FileReader("src/readfile"))) {
    // ... код ...
} catch (FileNotFoundException e) {
    // ... код ...
} catch (IOException e) {
    // ... код ...
}
```

- **try (BufferedReader reader = new BufferedReader(new FileReader("src/readfile")))**: Блок кода try-with-resources, который автоматически закрывает поток reader после завершения блока try.
    - **BufferedReader reader = new BufferedReader(new FileReader("src/readfile"))**: Создается объект reader типа BufferedReader, который используется для чтения файла “src/readfile”.
- **catch (FileNotFoundException e)**: Ловит исключение, которое возникает, если файл не найден.
- **catch (IOException e)**: Ловит исключение, которое может возникнуть при чтении файла.

6. Выполнение действий при успешном чтении:

```
System.out.println("---Начало файла---");

String line;

while ((line = reader.readLine()) != null) {

   System.out.println(line);

}

System.out.println("---Конец файла---");
```

- **System.out.println("---Начало файла---");**: Выводит в консоль сообщение “—Начало файла—”.
- **String line;**: Объявляет переменную line типа String для хранения прочитанной строки.
- **while ((line = reader.readLine()) != null)**: Цикл while, который читает файл по строкам, пока не достигнет конца файла.
- **System.out.println(line);**: Выводит прочитанную строку в консоль.
- **System.out.println("---Конец файла---");**: Выводит в консоль сообщение “—Конец файла—”.

7. Обработка ошибок:

```
catch (FileNotFoundException e) {

   System.out.println("Файл не найден");

} catch (IOException e) {

   System.err.println("Ошибка чтения файла: " + e.getMessage());

}
```

- **catch (FileNotFoundException e)**: Если файл не найден, выводится сообщение “Файл не найден”.
- **catch (IOException e)**: Если возникает ошибка при чтении файла, выводится сообщение “Ошибка чтения файла: ” с текстом сообщения об ошибке.