**МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Информационные технологии»

Лабораторная работа по ИТП №4

Выполнил: Студент группы

БВТ2204

Сарыков Эрбол

Москва

2023

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).

Пример реализации метода:

public class ArrayAverage {

public static void main(String[] args) {

int[] arr = {1, 2, 3, 4, 5};

int sum = 0;

try {

//Обработка массива

}

//Вывод массива

} catch (\* исключение\*) {

//Вывод исключения

} catch (\*исключение\*) {

//Вывод исключения

}

}

}

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 класс CustomUnsupportedOperationException,

который будет использоваться для обработки исключения

UnsupportedOperationException. Реализуйте программу, которая

выполняет математические операции (сложение, вычитание, умножение,

деление) с помощью со бственного класса и выбрасывайте исключение

CustomUnsupportedOperationException, если операция не

поддерживается.

Задание 1

public class ArrayAverage {

    public static void main(String[] args) {

        Object[] array = {4, -1, 6, "abc", null, true};

        try {

            System.out.println("Среднее: " + calculateAverage(array));

            System.out.println(array[array.length + 1]);

        } catch (ArrayIndexOutOfBoundsException e) {

            System.out.println("Произошла ошибка индексации: " + e.getMessage());

        } catch (Exception e) {

            System.out.println("Произошла ошибка: " + e.getMessage());

        }

    }

    static double calculateAverage(Object[] array) throws Exception {

        if (array == null || array.length == 0) {

            throw new Exception("Массив пуст или = 0");

        }

        double sum = 0;

        int validNumberCount = 0;

        for (Object obj : array) {

            try {

                if (obj != null) {

                    if (obj instanceof Number) {

                        sum += ((Number) obj).doubleValue();

                        validNumberCount++;

                    } else {

                        sum += Double.parseDouble(obj.toString());

                        validNumberCount++;

                    }

                }

            } catch (NumberFormatException e) {

                System.out.println("Пропуск нечисленных значений:" + obj);

            }

        }

        if (validNumberCount == 0) {

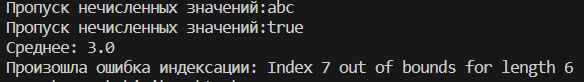
            throw new Exception("В массиве не найдено допустимых чисел");

        }

        return sum / validNumberCount;

    }

}



1. Метод calculateAverage(Object[] array)

Этот метод принимает массив объектов array, в котором содержатся числа и другие объекты.

Если массив пуст или его длина равна 0, метод генерирует исключение Exception с сообщением "Массив пуст или равен 0".

Метод выполняет следующие действия:

Инициализирует переменную sum для хранения суммы числовых значений.

Инициализирует переменную validNumberCount для подсчета допустимых чисел в массиве.

Затем метод выполняет итерацию по элементам массива:

Если элемент не равен null, метод проверяет, является ли он числовым (проверка с использованием instanceof Number). Если да, то число добавляется к сумме sum, и значение validNumberCount увеличивается.

Если элемент не является числом, метод пытается преобразовать его в число с помощью Double.parseDouble(). Если преобразование успешно, число также добавляется к сумме sum, и значение validNumberCount увеличивается.

Если преобразование не удалось (генерируется исключение NumberFormatException), метод выводит сообщение о пропуске нечислового значения.

После завершения итерации, если не было найдено допустимых чисел, метод генерирует исключение Exception с сообщением "В массиве не найдено допустимых чисел".

В конечном итоге метод возвращает среднее значение чисел в массиве, вычисляя сумму всех чисел и деля ее на validNumberCount.

2. Метод main

Метод main представляет точку входа в программу.

Создается массив array, содержащий элементы разных типов, включая числа, строки и другие объекты.

В блоке try, программа вызывает метод calculateAverage(array) для вычисления среднего значения в массиве и выводит результат на экран.

Затем программа пытается обратиться к элементу массива array с индексом, превышающим его размер (вызывается исключение ArrayIndexOutOfBoundsException), и выводит сообщение об ошибке индексации.

В блоке catch обрабатываются два типа исключений: ArrayIndexOutOfBoundsException и Exception. В случае ArrayIndexOutOfBoundsException, программа выводит сообщение о произошедшей ошибке индексации. В случае Exception, программа выводит сообщение о другой общей ошибке.

Задание 2

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.IOException;

public class Filecopy {

    public static void main(String[] args) {

        //& пример

        //\* String f\_file = "C:\\Users\\chiri\\Desktop\\Labi\\ИТИП\\Labs\\4\\1.txt";

        //\* String s\_file = "C:\\Users\\chiri\\Desktop\\Labi\\ИТИП\\Labs\\4\\2.txt";

        String f\_file = "вставь сюда путь файла";

        String s\_file = "вставь сюда путь файла";

        try {

            copyFile(f\_file, s\_file);

            if (new File(f\_file).length() > 0) {

                System.out.println("Содержимое файла 1 успешно вскопировано во 2-ой!");

            }

        } catch (IOException e) {

            System.out.println("При копировании файла произошла ошибка: " + e.getMessage());

        }

    }

    static void copyFile(String copy\_from\_here, String paste\_here) throws IOException {

        File sourceFile = new File(copy\_from\_here);

        if (!sourceFile.exists()) {

            throw new IOException("Исходный файл не существует.");

        }

        if (sourceFile.length() == 0) {

            System.out.println("Копируемый файл пуст.");

            return;

        }

        try (

            FileInputStream copy\_from\_hereInputStream = new FileInputStream(sourceFile);

            FileOutputStream destinationFileOutputStream = new FileOutputStream(paste\_here)

        ) {

            byte[] buffer = new byte[1024];

            int bytesRead;

            while ((bytesRead = copy\_from\_hereInputStream.read(buffer)) > 0) {

                destinationFileOutputStream.write(buffer, 0, bytesRead);

            }

        } catch (IOException e) {

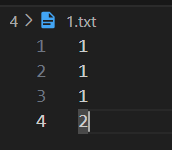
            throw new IOException("Не удалось скопировать файл: " + e.getMessage(), e);

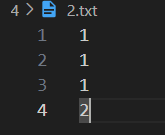
        }

    }

}







1. Метод copyFile(String copy\_from\_here, String paste\_here)

Этот метод принимает два параметра: copy\_from\_here (путь к файлу, который нужно скопировать) и paste\_here (путь к файлу, в который нужно вставить скопированное содержимое).

Метод выполняет следующие действия:

Проверяет существование исходного файла. Если исходный файл не существует, метод генерирует исключение IOException с сообщением "Исходный файл не существует."

Проверяет, что исходный файл не пуст. Если исходный файл пуст, метод выводит сообщение и завершает выполнение без копирования.

Создает FileInputStream для чтения содержимого исходного файла и FileOutputStream для записи в целевой файл.

Создает буфер байтов размером 1024 байта для копирования данных.

Читает данные из исходного файла в буфер и записывает их в целевой файл до тех пор, пока есть данные для чтения.

Закрывает потоки copy\_from\_hereInputStream и destinationFileOutputStream после завершения копирования.

2. Метод main

Метод main представляет точку входа в программу.

В этом методе определены две строки f\_file и s\_file, которые представляют пути к исходному и целевому файлам, которые должны быть скопированы. По умолчанию, пути к файлам указаны как заглушки (вставь сюда путь файла), и пользователю предлагается заменить их на фактические пути файлов.

В блоке try, программа вызывает метод copyFile(f\_file, s\_file) для выполнения копирования содержимого исходного файла в целевой файл.

После успешного выполнения операции копирования, программа проверяет, что размер исходного файла больше нуля (чтобы убедиться, что копирование произошло успешно) и выводит соответствующее сообщение.

Если во время операции копирования возникает ошибка типа IOException, программа перехватывает исключение и выводит сообщение об ошибке.

Задание 3

class CustomUnsupportedOperationException extends RuntimeException {

    public CustomUnsupportedOperationException(String message) {

        super(message);

    }

}

class MathOperations {

    public static double add(double a, double b) {

        return a + b;

    }

    public static double subtract(double a, double b) {

        return a - b;

    }

    public static double multiply(double a, double b) {

        return a \* b;

    }

    public static double divide(double a, double b) {

        if (b == 0) throw new ArithmeticException("Деление на ноль");

        return a / b;

    }

    public static double unsupportedOperation() {

        throw new CustomUnsupportedOperationException("Эта операция не поддерживается");

    }

}

// Основной класс

public class UOE {

    public static void main(String[] args) {

        try {

            System.out.println("Сложение: " + MathOperations.add(5, 3));

            System.out.println("Вычитание: " + MathOperations.subtract(5, 3));

            System.out.println("Умножение: " + MathOperations.multiply(5, 3));

            System.out.println("Деление: " + MathOperations.divide(5, 3));

            // попытка выполнить не поддерживаемую операцию

            MathOperations.unsupportedOperation();

        } catch (CustomUnsupportedOperationException e) {

            System.out.println("Ошибка: " + e.getMessage());

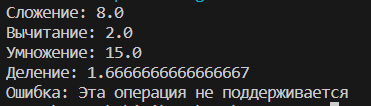
        } catch (Exception e) {

            System.out.println("Неизвестная ошибка: " + e.getMessage());

        }

    }

}



1. Класс CustomUnsupportedOperationException

Этот класс представляет пользовательское исключение, которое расширяет класс RuntimeException. Он используется для обозначения ситуаций, когда выполняется несовместимая операция.

2. Класс MathOperations

Этот класс предоставляет несколько статических методов для выполнения базовых математических операций, таких как сложение, вычитание, умножение и деление. Каждый метод возвращает результат операции.

Метод divide включает в себя проверку на деление на ноль. Если делитель равен нулю, метод генерирует исключение ArithmeticException.

Метод unsupportedOperation намеренно вызывает пользовательское исключение CustomUnsupportedOperationException, чтобы показать, как можно создавать и использовать пользовательские исключения.

3. Класс UOE (UnsupportedOperationException)

Этот класс представляет основной класс программы и содержит метод main, который выполняет следующие действия:

Выполняет базовые математические операции (сложение, вычитание, умножение и деление) с использованием методов из класса MathOperations и выводит результаты на экран.

Попытка выполнить операцию unsupportedOperation, которая вызывает пользовательское исключение CustomUnsupportedOperationException.

В блоке catch обрабатываются два типа исключений: CustomUnsupportedOperationException и общее исключение Exception. В случае исключения CustomUnsupportedOperationException, программа выводит сообщение об ошибке, указывающее на несовместимую операцию.

В случае исключения Exception, программа выводит сообщение об общей ошибке.