

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»

(МТУСИ)

Кафедра «Математическая Кибернетика и Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №4

по дисциплине

«Информационные технологии и программирование»

на тему

“Исключения и их обработка”

Выполнил:

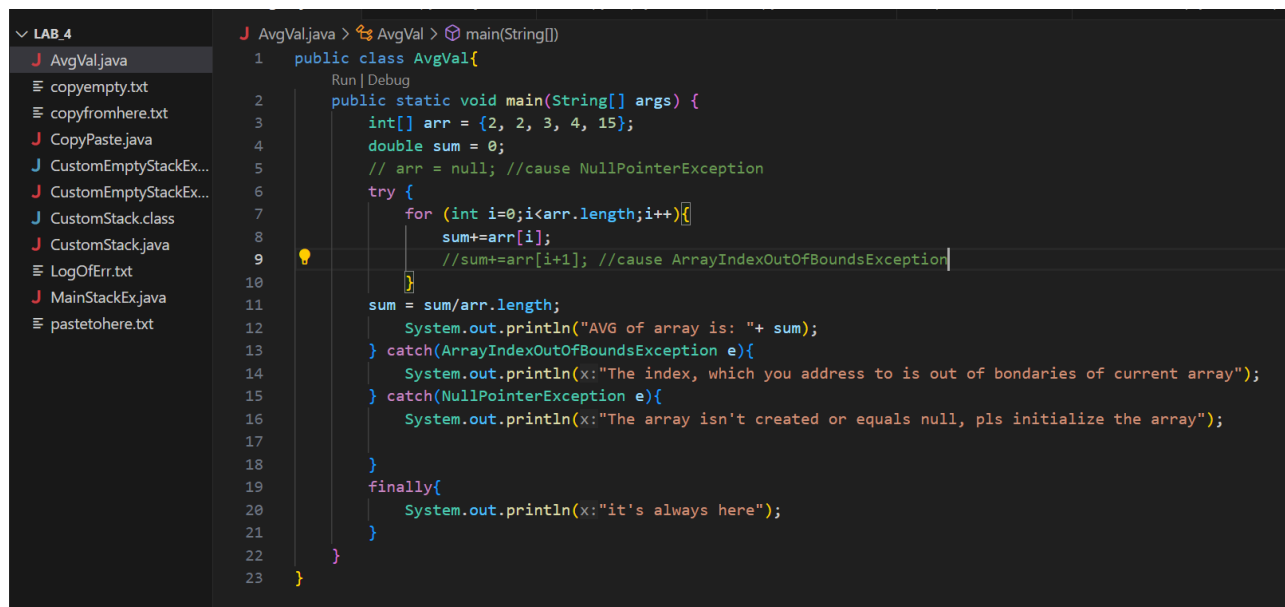
студент группы БВТ2302

Миронов А. А.

Москва, 2024 г.

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.



```
LAB_4
J AvgVal.java
  copyempty.txt
  copyfromhere.txt
  CopyPaste.java
  CustomEmptyStackEx...
  CustomEmptyStackEx...
  CustomStack.class
  CustomStack.java
  LogOfErr.txt
  MainStackEx.java
  pastetohere.txt

J AvgVal.java > AvgVal > main(String[])
1 public class AvgVal{
2     Run | Debug
3     public static void main(String[] args) {
4         int[] arr = {2, 2, 3, 4, 15};
5         double sum = 0;
6         // arr = null; //cause NullPointerException
7         try {
8             for (int i=0;i<arr.length;i++){
9                 sum+=arr[i];
10                //sum+=arr[i+1]; //cause ArrayIndexOutOfBoundsException
11            }
12            sum = sum/arr.length;
13            System.out.println("AVG of array is: "+ sum);
14        } catch(ArrayIndexOutOfBoundsException e){
15            System.out.println(x:"The index, which you address to is out of bondaries of current array");
16        } catch(NullPointerException e){
17            System.out.println(x:"The array isn't created or equals null, pls initialize the array");
18        }
19        finally{
20            System.out.println(x:"it's always here");
21        }
22    }
23 }
```

Создадим метод мейн, который считает среднее арифметическое аргументов массива. Облачим часть кода, которая может вызывать исключения в блок try, а далее обработаем исключения, при которых мы выходим за границы массива, или мы пытаемся работать с массивом, который является неинициализированным (=null). Блок finally выполняется, даже если было выброшено исключение.

Задание 2:

Необходимо написать программу, которая будет копировать содержимое одного файла в другой. При этом программа должна обрабатывать возможные ошибки, связанные с:

Вариант 1

Открытием и закрытием файлов

Вариант 2

Чтением и записью файлов

```

import java.io.FileNotFoundException;
import java.util.Scanner;
import java.io.FileWriter;
import java.util.NoSuchElementException;
import java.io.IOException;
import java.io.File;

public class CopyPaste {
    Run | Debug
    public static void main(String[] args) {
        String CopFilepath = "copyfromhere.txt";
        //String CopFilepath2 = "copyempty.txt"; //cause NoSuchElementException
        //String CopFilepath = "copyfromhere1.txt"; //cause FileNotFoundException
        String InsFilepath = "pastetohere.txt";
        String tempData = "";

        try{
            File Copfile = new File(CopFilepath);
            //File Copfile = new File(CopFilepath2); //cause NoSuchElementException
            Scanner readman = new Scanner(Copfile);
            //readman.nextLine(); //cause NoSuchElementException
            while (readman.hasNextLine()){
                tempData+=readman.nextLine()+"\n";
            }
            readman.close();

        } catch(FileNotFoundException e) {
            System.out.println(x:"No such file in current directory, please correct the name of the path.");
            e.printStackTrace();
        }
    }
}

```

Для начала импортируем все необходимые элементы, а это исключения, которые мы будем обрабатывать и классы и для копирования текста из одного файла и записи в другой. `FileNotFoundException` выбрасывается в случае, когда наш файл не найден по указанному пути. Метод `printStackTrace()` выводит в консоль обширную информацию о выброшенном исключении.

```

        e.printStackTrace();
    } catch(NoSuchElementException e){
        System.out.println(x:"You try to copy a line from an empty file");
        e.printStackTrace();
    } catch(Exception e){
        System.out.println(x:"Smtb went wrong :(");
        e.printStackTrace();
    }
}

try{
    FileWriter writeman = new FileWriter(InsFilepath);
    writeman.write(tempData);
    writeman.close();
    //writeman.write(tempData); //cause IOException
}

catch(FileNotFoundException e){
    System.out.println(x:"No such file with given path or name");
    e.printStackTrace();
} catch(IOException i) {
    System.out.println(x:"Maybe you are trying to read with readman after closing it?...");
    i.printStackTrace();
} catch(Exception e){
    System.out.println(x:"We've got the problem(");
    e.printStackTrace();
}
}
}

```

NoSuchElementException появляется, когда мы пытаемся скопировать строку из пустого файла с помощью сканнера. IOException же возникает, когда мы после закрытия вайтера пытаемся записать что то в другой файл. Стоит отметить, хпj в конце конструкции try-catch мы располагаем исключение Exception, которое стоит в классификации исключений выше двух предыдущих и соответствует любому исключению при его обработке. Чем выше иерархия исключения, тем ниже оно стоит в структуре try-catch.

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 5:

Создайте класс CustomEmptyStackException, который будет использоваться для обработки исключения EmptyStackException.

Напишите класс CustomStack, имитирующий стек, и, если происходит попытка извлечь элемент из пустого стека, выбрасывайте исключение CustomEmptyStackException.

```
CustomEmptyStackException.java > ...  
1 public class CustomEmptyStackException extends Exception {  
2     public CustomEmptyStackException(String message_1){  
3         super(message_1);  
4     }  
5 }
```

Создадим свой собственный класс исключений, который наследует класс Exception (все исключения, которые мы создаём, должны наследовать его) и создадим для него лишь конструктор с параметром, которым будет являться сообщение об ошибке. Конструктор нашего класса-исключения вызывает родительский конструктор класса Exception.

```
public class CustomStack {
    String[] elems;

    public CustomStack(int stackSize){
        elems = new String[stackSize];
    }

    public CustomStack(){
        this(stackSize:10);
    }

    public void SetEl(int index, String val){
        elems[index] = val;
    }

    public String GetEl(int index) throws CustomEmptyStackException{
        if (elems[index]==null){
            throw new CustomEmptyStackException(message_1:"You try to fetch the element from an empty stack");
        }
        return elems[index];
    }
}
```

Теперь создадим наш класс, для создания стэков, где будем хранить строки и назначать количество элементов в массиве (стэке) при инициализации. Стоит обратить внимание на метод, который может вызывать исключение. В нём мы вначале прописываем с помощью ключевого слова `throws`, какое исключение он выкидывает, а затем условие при котором это происходит. В данном случае — когда мы обращаемся к пустому элементу массива (само исключение себя не выкинет и нам нужно указать, когда это должно случаться).

```

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

public class MainStackEx {
    Run | Debug
    public static void main(String[] args) {
        String Logging_path = "LogOfErr.txt";
        CustomStack OurStack = new CustomStack();
        OurStack.SetEl(index:0, val:"mysterious thing");
        OurStack.SetEl(index:1, val:"spooky journal");
        OurStack.SetEl(index:2, val:"grappling hook");
        OurStack.SetEl(index:3, val:"beavers<3");
        OurStack.SetEl(index:4, val:"Gooblewonker");
        OurStack.SetEl(index:5, val:"cursed wax figure");
        OurStack.SetEl(index:6, val:"magic flashlight");

        try {
            System.out.println(OurStack.GetEl(index:0));
            System.out.println(OurStack.GetEl(index:2));
            System.out.println(OurStack.GetEl(index:4));
            System.out.println(OurStack.GetEl(index:6));
            System.out.println(OurStack.GetEl(index:7)); //cause CustomEmptyStackException
        } catch (CustomEmptyStackException e){
            System.out.println(x:"This stack do not consist anything, even a mystery :0");
            e.printStackTrace();
            try{
                FileWriter writeman = new FileWriter(Logging_path, append:true);
                writeman.write(e.toString()+"\n");
            }
        }
    }
}

```

```

        try{
            FileWriter writeman = new FileWriter(Logging_path, append:true);
            writeman.write(e.toString()+"\n");
            writeman.close();
            //writeman.write(e.toString()); //cause IOException
        } catch (IOException i){
            System.out.println(x:"Apparently here are more mysteries, than you capable to comprehend");
            i.printStackTrace();
        }
    }
}

```

И наконец напишем файл мэйн, в котором заполним объект нашего стека значениями, а затем попытаемся их вывести и при возникновении исключений обработаем известным нам способом.

Помимо прочего мы записываем исключение, касающееся попытки извлечения из стека пустого значения с помощью известного нам из второго задания этой лабораторной класса `FileWriter`, в отдельный текстовый файл. Таким образом мы реализуем логгирование.

```
LogOfErr.txt
1 CustomEmptyStackException: You try to fetch the element from an empty stack
2 CustomEmptyStackException: You try to fetch the element from an empty stack
3
```

Для наглядности продемонстрируем, как выглядят мои исключения в консоли, когда они возникают при выполнении программы.

```
PS C:\Users\User\Desktop\MTUCI\2_course\java_subject\just_labs\j_labzzz\lab_4> java MainStackEx.java
mysterious thing
grappling hook
Gooblewonker
magic flashlight
This stack do not consist anything, even a mystery :O
CustomEmptyStackException: You try to fetch the element from an empty stack
    at CustomStack.GetEl(CustomStack.java:18)
    at MainStackEx.main(MainStackEx.java:22)
    at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:103)
    at java.base/java.lang.reflect.Method.invoke(Method.java:580)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.execute(Main.java:484)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.run(Main.java:208)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.main(Main.java:135)
Apparently here are more mysteries, than you capable to comprehend
java.io.IOException: Stream closed
    at java.base/sun.nio.cs.StreamEncoder.ensureOpen(StreamEncoder.java:52)
    at java.base/sun.nio.cs.StreamEncoder.lockedWrite(StreamEncoder.java:151)
    at java.base/sun.nio.cs.StreamEncoder.write(StreamEncoder.java:139)
    at java.base/sun.nio.cs.StreamEncoder.write(StreamEncoder.java:167)
    at java.base/java.io.OutputStreamWriter.write(OutputStreamWriter.java:237)
    at java.base/java.io.Writer.write(Writer.java:278)
    at MainStackEx.main(MainStackEx.java:31)
    at java.base/jdk.internal.reflect.DirectMethodHandleAccessor.invoke(DirectMethodHandleAccessor.java:103)
    at java.base/java.lang.reflect.Method.invoke(Method.java:580)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.execute(Main.java:484)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.run(Main.java:208)
    at jdk.compiler/com.sun.tools.javac.launcher.Main.main(Main.java:135)
```

Вывод: Мы ознакомились со структурой и иерархией исключений в джава и с их практическим применением.

А также поняли как они работают и научились взаимодействовать с ними, обрабатывая.

Более того, мы научились создавать собственные исключения и контролировать их поведение.