

**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ  
КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение  
высшего образования**

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

Кафедра «Математическая Кибернетика и Информационные технологии»

Дисциплина «Информационные технологии и программирование»

Лабораторная работа №4

«Обработка исключений в Java»

Выполнила:

Студентка группы БВТ2303

Морозова Ольга

## Цель работы:

Изучение способов обработки исключений и их логирования, и применение полученных знаний на практике на языке программирования Java.

## Ход работы:

### Задание 1.

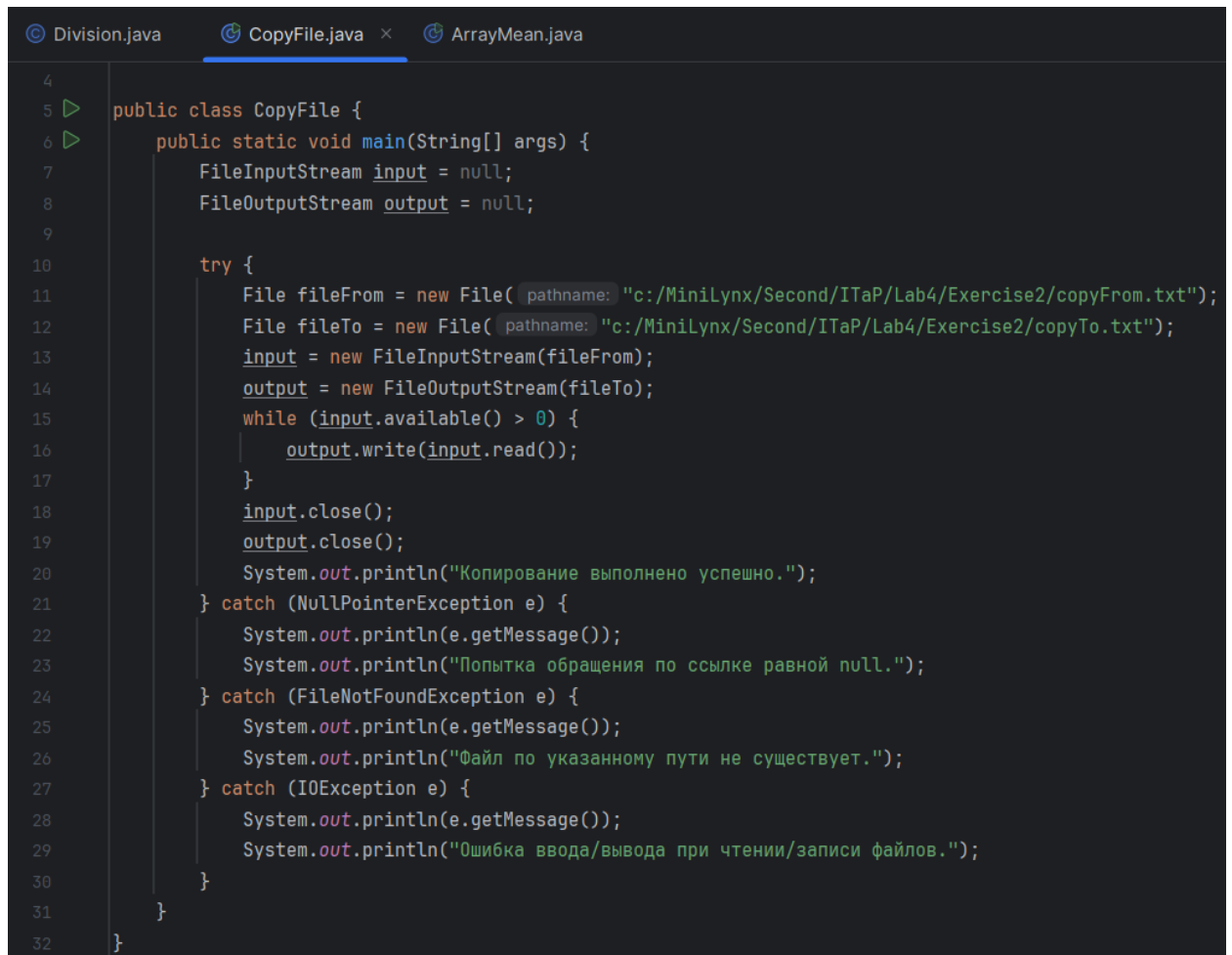
Создадим простой класс нахождения среднего арифметического чисел массива и на его основе сделаем проверку на различные исключения, которые могут возникнуть при выполнении, как то: переполнение/выход за границы массива; попытка приведения к типу Integer букв или иных входных данных неподходящего типа; попытка деления на 0 или иные ошибки при выполнении операции деления и другие. Всё это мы делаем через блоки try-catch, используя имеющиеся в Java исключения.



```
3 public class ArrayMean {
4     public static void main(String[] args) {
5         int[] mas;
6         int sum = 0;
7
8         try {
9             mas = new int[args.length];
10            for (int i = 0; i < args.length; i++) {
11                mas[i] = Integer.parseInt(args[i]);
12            }
13            for (int m : mas) {
14                sum += m;
15            }
16            sum /= mas.length;
17        } catch (ArrayIndexOutOfBoundsException e) {
18            System.out.println(e.getMessage());
19            System.out.println("Выход за пределы массива.");
20        } catch (ArrayStoreException e) {
21            System.out.println(e.getMessage());
22            System.out.println("Попытка записи что-то кроме чисел типа int в массив.");
23        } catch (NumberFormatException e) {
24            System.out.println(e.getMessage());
25            System.out.println("Ошибка при попытке приведения введенных данных к int.");
26        } catch (NullPointerException e) {
27            System.out.println(e.getMessage());
28            System.out.println("Попытка записи пустого значения в массив.");
29        } catch (ArithmeticException e) {
30            System.out.println(e.getMessage());
31            System.out.println("Ошибка при арифметических вычислениях.");
32        } finally {
33            System.out.println("Среднее арифметическое элементов массива: " + sum);
34        }
35    }
36 }
```

## Задание 2.

Создаём простой класс, в задачи которого входит копирование данных из одного файла в другой. В данном случае мы при помощи блоков try-catch выявляем ошибки, как то: отсутствие одного из файлов, с которыми предстоит работать; попытка работы со значением null; ошибка ввода/вывода при открытии/закрытии файлов или в процессе копирования (чтения/записи данных).

The image shows a screenshot of an IDE with three tabs: Division.java, CopyFile.java (active), and ArrayMean.java. The CopyFile.java file contains the following Java code:

```
4
5 public class CopyFile {
6     public static void main(String[] args) {
7         FileInputStream input = null;
8         FileOutputStream output = null;
9
10        try {
11            File fileFrom = new File( pathname: "c:/MiniLynx/Second/ITaP/Lab4/Exercise2/copyFrom.txt");
12            File fileTo = new File( pathname: "c:/MiniLynx/Second/ITaP/Lab4/Exercise2/copyTo.txt");
13            input = new FileInputStream(fileFrom);
14            output = new FileOutputStream(fileTo);
15            while (input.available() > 0) {
16                output.write(input.read());
17            }
18            input.close();
19            output.close();
20            System.out.println("Копирование выполнено успешно.");
21        } catch (NullPointerException e) {
22            System.out.println(e.getMessage());
23            System.out.println("Попытка обращения по ссылке равной null.");
24        } catch (FileNotFoundException e) {
25            System.out.println(e.getMessage());
26            System.out.println("Файл по указанному пути не существует.");
27        } catch (IOException e) {
28            System.out.println(e.getMessage());
29            System.out.println("Ошибка ввода/вывода при чтении/записи файлов.");
30        }
31    }
32 }
```

## Задание 3.

На сей раз мы создаём простой класс для деления двух чисел. Для проверки исключения при делении на 0 мы создаём свой собственный класс-исключение. Его можно прописать в одном файле с основным классом, в котором и выполняем данную арифметическую операцию. Для «выбрасывания» своего исключения используем throw, также при помощи throws следует указывать некоторые исключения, которые может при определённых обстоятельствах «выбросить» класс или метод.

```

Division.java x CopyFile.java ArrayMean.java
1 package Lab4.Exercise3;
2
3 import java.io.IOException;
4 import java.util.logging.*;
5
6 public class Division {
7     static Logger log = Logger.getLogger("Logs"); 4 usages
8
9     public static void main(String[] args) throws IOException {
10         FileHandler file = new FileHandler("c:/MiniLynx/Second/ITaP/Lab4/Exercise3/Logs.log", append: true);
11         log.addHandler(file);
12         file.setFormatter(new SimpleFormatter());
13         try {
14             int a = Integer.parseInt(args[0]);
15             int b = Integer.parseInt(args[1]);
16             System.out.println(DivineTwoInt(a, b));
17         } catch (ArrayIndexOutOfBoundsException e) {
18             log.log(Level.SEVERE, msg: "Выход за пределы массива или работа с пустым массивом.");
19         } catch (NumberFormatException e) {
20             log.log(Level.SEVERE, msg: "Ошибка при попытке приведения введённых данных к int.");
21         } catch (CustomDivisionException e) {
22             log.log(Level.SEVERE, msg: "Ошибка при попытке деления на 0.");
23         }
24     }
25
26     public static int DivineTwoInt(int a, int b) throws CustomDivisionException { 1 usage
27         if (b == 0) {
28             throw new CustomDivisionException();
29         } else {
30             return a / b;
31         }
32     }
33 }
34
35 class CustomDivisionException extends Exception { 3 usages
36     public CustomDivisionException() { 1 usage
37         super("Ошибка при попытке деления на 0.");
38     }
39 }

```

## Вывод:

Мы изучили способы обработки исключений и их логирование, а также написали программы на языке программирования Java с использованием полученных знаний.

GitHub - [https://github.com/MiniLynx13/ITaP\\_Lab4](https://github.com/MiniLynx13/ITaP_Lab4)