**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»**

Кафедра Математическая кибернетика и информационные технологии

**Лабораторная работа №4**

**Обработка исключений**

Выполнил: студент группы \_\_\_БВТ2402\_\_\_

Голиков Михаил Вячеславович

Руководитель: \_Мосева Марина Сергеевна\_

Москва, 2077

## Цель работы

Целью лабораторной работы является изучение механизмов обработки исключений в языке программирования Java, освоение принципов создания собственных классов исключений, а также реализация безопасных и устойчивых программ с использованием блоков try, catch, finally и пользовательских исключений.  
Кроме того, работа направлена на формирование навыков логирования ошибок и обеспечения корректного завершения программ при возникновении нештатных ситуаций.

## Индивидуальное задание

В рамках лабораторной работы необходимо реализовать следующие задачи:

1. **Задача**   
   Написать программу, которая вычисляет среднее арифметическое элементов массива. Программа должна обрабатывать ошибки, связанные с:
   * выходом за границы массива;
   * некорректными данными (например, если элемент массива не является числом).  
     Для этого требуется использовать конструкцию try-catch-finally.
2. **Задача**   
   Написать программу, которая копирует содержимое одного файла в другой.  
   Необходимо обработать возможные исключения, связанные с:

* отсутствием исходного файла;
* корректным закрытием потоков после завершения работы (вариант 1).

1. **Задача**   
   Создать Java-проект для работы с пользовательскими исключениями.  
   Для каждой из восьми задач в будущем будет добавлен собственный класс обработки ошибок.  
   В рамках текущего варианта реализовать класс CustomInputMismatchException для обработки исключения InputMismatchException.

Программа должна:

* считывать целое число с клавиатуры;
* при вводе некорректных данных выбрасывать пользовательское исключение CustomInputMismatchException;
* логировать информацию обо всех исключениях в текстовый файл exceptions.txt.

(Вариант 6).

## Основная часть

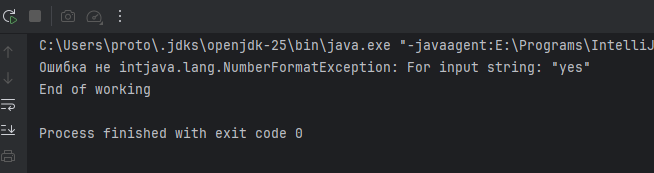
В процессе выполнения лабораторной работы были реализованы три отдельных класса, каждый из которых решает свою задачу, связанную с обработкой исключений:

1. **ArrayChecker** — демонстрирует работу с массивами и отлавливает исключения типа IndexOutOfBoundsException и NumberFormatException.  
   Программа корректно завершает выполнение, даже если пользователь передал неверные данные или количество аргументов превышает размер массива.
2. **FileOpener** — выполняет копирование данных между файлами и отлавливает ошибки NoSuchFileException и IOException.  
   Реализовано уведомление пользователя о каждой проблеме и корректное завершение программы с выводом сообщений об ошибках.
3. **CustomException** и **CustomInputMismatchException** — демонстрируют создание собственного исключения и ведение журнала ошибок.  
   Все выброшенные исключения записываются в файл exceptions.txt, что позволяет анализировать их без вывода в консоль.

Таким образом, каждая программа демонстрирует различные аспекты работы с исключениями в Java: системные ошибки, ошибки ввода-вывода и пользовательские исключения.

package Lab\_4;  
  
import java.util.Arrays;  
import java.util.Objects;  
  
public class ArrayChecker {  
  
 public static void main(String[] args) {  
 int[] arr = new int[5];  
 int summ = 0;  
  
 try {  
 for (int i = 0; i < arr.length; i++) {  
 arr[i] = Integer.*parseInt*(args[i]);  
 summ += arr[i];  
 }  
 if (args.length > arr.length) {  
 throw new IndexOutOfBoundsException("args.length > arr.length");  
 }  
  
 System.*out*.println(Arrays.*toString*(arr));  
 double avg = (double) summ /5;  
 System.*out*.println(avg);  
  
 } catch (IndexOutOfBoundsException e) {  
 System.*out*.println("Ошибка индекса"+ e.toString());  
 } catch (NumberFormatException e) {  
 System.*out*.println("Ошибка не int" + e.toString());  
 } catch (Exception e) {  
 System.*out*.println("Неопознанная ошибка" + e.toString());  
 } finally {  
 System.*out*.println("End of working");  
 }  
 }  
}

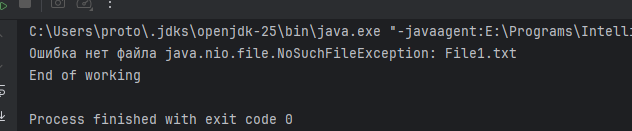
Элемент 1 ― Код программы 1



Элемент 2 ― Вывод работы программы 1

package Lab\_4;  
  
import java.io.\*;  
import java.nio.file.\*;  
  
public class FileOpener {  
 public static void main(String[] args){  
 Path fileOne = Paths.*get*("File.txt");  
 Path fileTwo = Paths.*get*("FileTwo.txt");  
  
 try{  
 Files.*copy*(fileOne, fileTwo, StandardCopyOption.*REPLACE\_EXISTING*);  
 System.*out*.println("Данные из файла 1 скопированы в файл 2");  
 } catch (NoSuchFileException e){  
 System.*out*.println("Ошибка нет файла " + e.toString());  
 } catch (IOException e){  
 System.*out*.println("Ошибка IO " + e.toString());  
 } catch (Exception e) {  
 System.*out*.println("Неопознанная ошибка " + e.toString());  
 } finally {  
 System.*out*.println("End of working");  
 }  
 }  
}

Элемент 3 ― Код программы 2



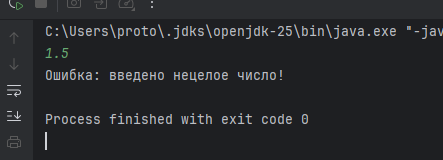
Элемент 4 ― Вывод работы программы 2

package Lab\_4;  
  
import java.io.IOException;  
import java.nio.file.Files;  
import java.nio.file.StandardOpenOption;  
import java.util.Collections;  
import java.util.InputMismatchException;  
import java.util.Scanner;  
import java.io.\*;  
import java.nio.file.\*;  
  
public class CustomException {  
  
 public static void logException(Exception e) {  
 try {  
 Path path = Path.*of*("exceptions.txt");  
 Files.*write*(path, Collections.*singletonList*("Исключение:" + e.toString()+ "\n"), StandardOpenOption.*CREATE*, StandardOpenOption.*APPEND*);  
  
 } catch (IOException ioEx) {  
 System.*out*.println("Ошибка при записи в лог: " + ioEx.getMessage());  
 }  
 }  
  
 public static void main(String[] args) {  
 Scanner scanner = new Scanner(System.*in*);  
  
 try {  
 int number = scanner.nextInt();  
 System.*out*.println("Число: " + number);  
 } catch (InputMismatchException e) {  
 CustomInputMismatchException customException = new CustomInputMismatchException("Ошибка: введено нецелое число!");  
 *logException*(customException);  
 System.*out*.println(customException.getMessage());  
 } catch (Exception e) {  
 *logException*(e);  
 System.*out*.println("Неопознанная ошибка: " + e.getMessage());  
 } finally {  
 scanner.close();  
 }  
 }  
}

Элемент 5 ― Код программы 3

package Lab\_4;  
  
  
class CustomInputMismatchException extends Exception {  
 public CustomInputMismatchException(String message) {  
 super(message);  
 }  
  
}

Элемент 6 ― Код программы CustomInputMismatchException



Элемент 7 ― Вывод работы программы 3

## Заключение

В ходе выполнения лабораторной работы были изучены и применены на практике основные механизмы **обработки исключений** в языке Java.  
Были созданы программы, демонстрирующие:

* корректную обработку стандартных исключений (IndexOutOfBoundsException, IOException, InputMismatchException);
* работу с пользовательскими исключениями;
* применение блоков try, catch и finally для обеспечения безопасного завершения программ;
* ведение журнала ошибок (логирование) в текстовый файл.

Работа позволила закрепить навыки проектирования надёжных программ и повысить понимание принципов устойчивого программирования на Java.

**Github:** [**https://github.com/Prototype721/Java**](https://github.com/Prototype721/Java)