**МОЛДАВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ**

**Факультет Математики и Информатики**

**Департамент Информатики**

Лабораторная работа № 2

Программирование на JAVA

Создание пользовательского исключения и обработка ошибок при работе с файлами

Проверил: профессор, др. Gheorghe Latu

Выполнил: Mamaliga Artur grupa I2302-1

Кишинев, 2024

## Цель работы:

Создать программу на языке Java, которая:  
1. Определяет собственное исключение.  
2. Реализует метод, который может возбуждать это исключение.  
3. Использует механизм обработки исключений try-catch для корректной обработки ошибок.  
4. Работает с файлами, читая строку из файла и обрабатывая ситуацию, когда файл отсутствует.

## Задание:

В данной лабораторной работе требуется:  
- Создать пользовательское исключение (класс).  
- Написать метод, который возбуждает это исключение.  
- Реализовать метод, который перехватывает и обрабатывает исключение с использованием блока try-catch.  
- Организовать ввод пути к файлу с консоли и обрабатывать ситуацию отсутствия файла. В случае ошибки программа должна выбрасывать и обрабатывать исключение, связанное с отсутствием файла.

## Ход работы:

### 1. Создание пользовательского исключения:

Был разработан класс FileNotFoundExceptionCustom, который наследуется от стандартного класса FileNotFoundException. Этот класс включает поле filePath, которое хранит путь к файлу, и метод для его получения. Пользовательское исключение выбрасывается в случае, если указанный файл не найден.

Код класса:  
  
import java.io.FileNotFoundException;  
  
public class FileNotFoundExceptionCustom extends FileNotFoundException {  
 private String filePath;  
  
 public String getFilePath() {  
 return filePath;  
 }  
  
 public FileNotFoundExceptionCustom(String message, String filePath) {  
 super(message);  
 this.filePath = filePath;  
 }  
}

### 2. Метод для возбуждения исключения:

Был реализован метод readLineFile(String filePath), который читает первую строку из файла. В случае отсутствия файла метод выбрасывает пользовательское исключение FileNotFoundExceptionCustom.

Код метода:  
  
public static String readLineFile(String filePath) throws FileNotFoundExceptionCustom {  
 File file = new File(filePath);  
 try {  
 Scanner fileScanner = new Scanner(file);  
 return fileScanner.nextLine();  
 } catch (FileNotFoundException e) {  
 throw new FileNotFoundExceptionCustom("Couldn't find the file at the path: ", filePath);  
 }  
}

### 3. Метод для обработки исключения:

В методе main реализован ввод пути к файлу с консоли и обработка возможного исключения с использованием конструкции try-catch. Если файл не найден, выбрасывается исключение и выводится сообщение об ошибке с указанием пути к файлу.

Код метода main:  
  
public static void main(String[] args){  
 Scanner sc = new Scanner(System.in);  
  
 System.out.println("Enter the file path: ");  
 String path = sc.nextLine();  
  
 try {  
 String firstLine = readLineFile(path);  
 System.out.println("First line of the file: " + firstLine);  
 } catch (FileNotFoundExceptionCustom e) {  
 System.out.println(e.getMessage() + e.getFilePath());  
 }  
}

### 4. Обработка ошибки отсутствия файла:

Программа запрашивает путь к файлу у пользователя и пытается открыть файл для чтения. Если файл отсутствует, срабатывает блок catch, который перехватывает исключение FileNotFoundExceptionCustom и выводит на экран соответствующее сообщение, содержащее информацию о пути к файлу.

## Результаты:

1. Программа успешно компилируется и выполняет следующие задачи:  
- Чтение строки из файла по указанному пути.  
- Обработка исключения в случае отсутствия файла с выводом информативного сообщения.  
2. При отсутствии файла на указанном пути программа корректно выбрасывает и обрабатывает пользовательское исключение.

## Заключение:

В ходе выполнения лабораторной работы была изучена работа с исключениями в Java, включая создание пользовательского исключения и его обработку. Программа корректно обрабатывает ситуацию отсутствия файла и выбрасывает соответствующее исключение, что позволяет улучшить устойчивость и предсказуемость работы программы при ошибках ввода-вывода.

## Исходный код программы:

Program.java:  
  
import java.io.File;  
import java.io.FileNotFoundException;  
import java.util.Scanner;  
  
public class Program {  
  
 public static String readLineFile(String filePath) throws FileNotFoundExceptionCustom {  
 File file = new File(filePath);  
 try {  
 Scanner fileScanner = new Scanner(file);  
 return fileScanner.nextLine();  
 } catch (FileNotFoundException e) {  
 throw new FileNotFoundExceptionCustom("Couldn't find the file at the path: ", filePath);  
 }  
 }  
  
 public static void main(String[] args){  
 Scanner sc = new Scanner(System.in);  
  
 System.out.println("Enter the file path: ");  
 String path = sc.nextLine();  
  
 try {  
 String firstLine = readLineFile(path);  
 System.out.println("First line of the file: " + firstLine);  
 } catch (FileNotFoundExceptionCustom e) {  
 System.out.println(e.getMessage() + e.getFilePath());  
 }  
 }  
}

FileNotFoundExceptionCustom.java:  
  
import java.io.FileNotFoundException;  
  
public class FileNotFoundExceptionCustom extends FileNotFoundException {  
 private String filePath;  
  
 public String getFilePath() {  
 return filePath;  
 }  
  
 public FileNotFoundExceptionCustom(String message, String filePath) {  
 super(message);  
 this.filePath = filePath;  
 }  
}