|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **Компьютерные системы и сети (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.04.01 Информатика и вычислительная техника**

**Отчет**

**по лабораторной работе № 5**

**Название:** Потоковый ввод/вывод, исключения

**Дисциплина:** Языки программирования для работы с большими данными

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ6-21М |  |  | Д.Ю. Ермошин |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | П.В. Степанов |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Задание**

Научиться работать с потоковым вводом выводом в Java, обрабатывать исключения.

**Выполнение работы**

**Вариант 1**

Выполнить задания на основе варианта 1 лабораторной работы 4, контролируя состояние потоков ввода/вывода. При возникновении ошибок, связанных с корректностью выполнения математических операций, генерировать и обрабатывать исключительные ситуации. Предусмотреть обработку исключений, возникающих при нехватке памяти, отсутствии требуемой записи (объекта) в файле, недопустимом значении поля и т.д.

|  |
| --- |
| package lab5;  import lab4.Mobile;  import java.io.IOException;  import java.io.InputStream;  public class Variant1 {  public void work() {  Mobile mobile;  try {  InputStream inputStream = System.in;  mobile = new Mobile(inputStream);  inputStream.close();  } catch (IOException e) {  System.out.println("Error by reading");  e.printStackTrace();  mobile = new Mobile();  } catch (NullPointerException e) {  System.out.println("Can't find such input stream");  e.printStackTrace();  mobile = new Mobile();  } catch (OutOfMemoryError e) {  System.out.println("Not enough memory for object creating");  e.printStackTrace();  return;  }  mobile.getMobileInfos().forEach(mobileInfo ->  mobileInfo.getMobiles()  .forEach(mobileModel-> {  try {  mobileModel.writeToOutputStream(System.out);  } catch (IOException e) {  System.out.println("Error by printing"); e.printStackTrace();  }  }));  }  } |

**Вариант 2**

Выполнить задания из варианта 2 лабораторной работы 4, реализуя собственные обработчики исключений и исключения ввода/вывода.

|  |
| --- |
| package lab5;  import lab4.Comedy;  import lab4.Film;  import java.io.BufferedWriter;  import java.io.IOException;  import java.io.OutputStreamWriter;  import java.util.NoSuchElementException;  import java.util.Scanner;  public class Variant2 {  public void work() {  Film film = new Comedy();  try {  Scanner scanner = new Scanner(System.in);  film.setRating(scanner.nextInt());  } catch (NoSuchElementException e) {  System.out.println("Input stream is exhausted");  e.printStackTrace();  film.setRating(8);  } catch (IllegalStateException e) {  System.out.println("This scanner is closed");  e.printStackTrace();  film.setRating(8);  } catch (OutOfMemoryError e) {  System.out.println("Not enough memory for object creating");  e.printStackTrace();  return;  }  try (final BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(System.out))) {  bw.write(String.valueOf(film.getRating()));  } catch (IOException e) {  System.out.println("Error by printing");  e.printStackTrace();  }  }  } |

**Вариант 3**

Требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:

- каждая строка состоит из одного слова;

- каждая строка состоит из нескольких слов.

Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.

В каждой строке найти и удалить заданную подстроку.

|  |
| --- |
| package lab5;  import java.io.BufferedReader;  import java.io.BufferedWriter;  import java.io.IOException;  import java.io.InputStreamReader;  import java.io.OutputStreamWriter;  import java.nio.file.Files;  import java.nio.file.Paths;  import java.util.ArrayList;  import java.util.List;  /\*\*  \* В следующих заданиях требуется ввести последовательность строк из текстового потока и выполнить указанные действия. При этом могут рассматриваться два варианта:  \* - каждая строка состоит из одного слова;  \* - каждая строка состоит из нескольких слов.  \* Имена входного и выходного файлов, а также абсолютный путь к ним могут быть введены как параметры командной строки или храниться в файле.  \* 1. В каждой строке найти и удалить заданную подстроку.  \*/  public class Variant3 {  public void work(final String inputFile, final String outputString, final String substring) throws IOException {  final List<String> list = new ArrayList<>();  try (final BufferedReader br = new BufferedReader(new InputStreamReader(Files.newInputStream(Paths.get(inputFile))))) {  String line = br.readLine();  while (line != null) {  list.add(line);  line = br.readLine();  }  }  for (int i = 0; i < list.size(); i++) {  list.set(i, list.get(i).replaceAll(substring, ""));  }  try (final BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(Files.newOutputStream(Paths.get(outputString))))) {  for (final String formattedLine : list) {  bw.write(formattedLine);  bw.newLine();  }  }  }  } |

**Вариант 4**

Для вывода результатов необходимо создавать новую директорию и файл средствами класса File.

Прочитать текст Java-программы и все слова public в объявлении атрибутов и методов класса заменить на слово private.

|  |
| --- |
| package lab5;  import java.io.BufferedWriter;  import java.io.File;  import java.io.FileOutputStream;  import java.io.IOException;  import java.io.OutputStreamWriter;  import java.util.Scanner;  /\*\*  \* При выполнении следующих заданий для вывода результатов создавать новую директорию и файл средствами класса File  \* 1. Прочитать текст Java-программы и все слова public в объявлении атрибутов и методов класса заменить на слово private.  \*/  public class Variant4 {  public void work() throws IOException {  final File newFile = new File("src/lab5/output/Car.java");  newFile.getParentFile().mkdir();  newFile.createNewFile();  final File currentFile = new File("src/lab3/Car.java");  try (final BufferedWriter out = new BufferedWriter(new OutputStreamWriter(new FileOutputStream(newFile)))) {  final Scanner sc = new Scanner(currentFile);  while (sc.hasNextLine()) {  out.write(sc.nextLine().replace("public", "private"));  out.newLine();  }  }  }  } |

**Заключение**

В данной лабораторной работе мы научились работать с потоковым вводом выводом в Java, обрабатывать исключения.