Міністерство освіти та науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



**ЗВІТ**

**з лабораторної роботи № 6**

**з дисципліни:** «Кросплатформенні засоби програмування»

**на тему:** «ФАЙЛИ»

Виконав: ст. гр. КІ-35

Рибець О.В.

Прийняв:

доцент кафедри ЕОМ

Іванов Ю.С

Львів-2022

**Мета роботи:** оволодіти навиками використання засобів мови Java для роботи з потоками і файлами.

**ЗАВДАННЯ**

1. Створити клас, що реалізує методи читання/запису у текстовому і двійковому форматах результатів роботи класу, що розроблений у лабораторній роботі №5. Написати програму для тестування коректності роботи розробленого класу.
2. Для розробленої програми згенерувати документацію.
3. Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагмент згенерованої документації.
4. Дати відповідь на контрольні запитання.

**Варіант 7:**



**Текст програми:**

|  |
| --- |
| ***Лістинг CalcException***  package lab6;  public class CalcException extends ArithmeticException {  public CalcException() {  }   */\*\*  \* Constructor with string  \** ***@param*** *cause  \*/* public CalcException(String cause)  {  super(cause);  } }  }  ***Лістинг AccountFio***  package lab6;  import java.io.\*; import java.util.Scanner;  public class AccountantFio {  private double result;  private String BINARY\_FILE\_PATH;  private String TXT\_FILE\_PATH;  public void writeResTxt(String fName) throws FileNotFoundException  {  PrintWriter f = new PrintWriter(fName);  f.printf("%f ",result);  f.close();  }   */\*\*  \* Method implements read from file  \** ***@param*** *fName  \*/* public void readResTxt(String fName)  {  try  {  File f = new File (fName);  if (f.exists())  {  Scanner s = new Scanner(f);  result = s.nextDouble();  s.close();  }  else  throw new FileNotFoundException("File " + fName + "not found");  }  catch (FileNotFoundException ex)  {  System.*out*.print(ex.getMessage());  }  }   */\*\*  \* Method implements write to bin file  \** ***@param*** *fName  \** ***@throws*** *FileNotFoundException  \** ***@throws*** *IOException  \*/* public void writeResBin(String fName) throws FileNotFoundException, IOException  {  DataOutputStream f = new DataOutputStream(new FileOutputStream(fName));  f.writeDouble(result);  f.close();   }   */\*\*  \* Method implements read from bin file  \** ***@param*** *fName  \** ***@throws*** *FileNotFoundException  \** ***@throws*** *IOException  \*/* public void readResBin(String fName) throws FileNotFoundException, IOException  {  DataInputStream f = new DataInputStream(new FileInputStream(fName));  result = f.readDouble();  f.close();  }   */\*\*  \* Method calculates the tg(x)/ctg(x) expression  \** ***@param*** *x - angle in degrees  \** ***@throws*** *CalcException  \*/* public double calculator(double x) throws lab5.CalcException {  double rad = 0;  double rad2 = 0;  try {  rad = Math.*toRadians*(7\*x-1);  rad2 = Math.*toRadians*(x);  result = (1/Math.*tan*(rad2))/Math.*sin*(rad);  if (result == Double.*POSITIVE\_INFINITY* || result == Double.*NEGATIVE\_INFINITY*) {  throw new ArithmeticException();  }  } catch (ArithmeticException e) {   if (rad==Math.*PI*/2.0 || rad==-Math.*PI*/2.0)  throw new lab5.CalcException("Exception reason: Illegal value of X for tangent calculation");  else if (x==0)  throw new lab5.CalcException("Exception reason: X = 0");  else  throw new lab5.CalcException("Unknown reason of the exception during exception calculation");  }  return result;  }   public double getResult()  {  return result;  }  }  ***Лістинг EquotionsApp***  package lab6;  import java.io.\*; import java.util.RandomAccess; import java.util.Scanner;  public class EquationsApp {  public static void main(String[] args) throws IOException {  boolean IsCatched = true;  AccountantFio obj = new AccountantFio();  Scanner s = new Scanner(System.*in*);  System.*out*.print("Enter data: ");  double data = s.nextDouble();  RandomAccessFile randomAccessFile = new RandomAccessFile(new File("Texxxxt.txt"), "rw");  randomAccessFile.writeUTF("dgdfgf");  randomAccessFile.close();  FileOutputStream fileInputStream = new FileOutputStream(new File("Texxt.txt"));  fileInputStream.write('K');  fileInputStream.close();   try  {  obj.calculator(data);  }  catch (CalcException ex)  {  IsCatched = false;  System.out.println(ex.getMessage());  }   if(IsCatched) {  System.out.println("Result is: " + obj.getResult());  obj.writeResTxt("textRes.txt");  obj.writeResBin("BinRes.bin");   obj.readResBin("BinRes.bin");  System.out.println("Result is(read from bin): " + obj.getResult());  obj.readResTxt("textRes.txt");  System.out.println("Result is(read from txt): " + obj.getResult());  }   } } |

**Результат роботи програми:**

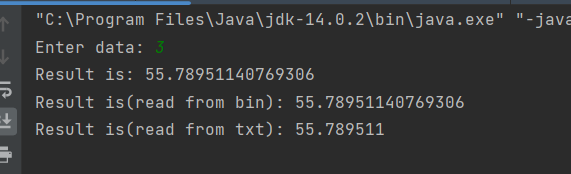
****

Рис.1.1 Результат роботи програми



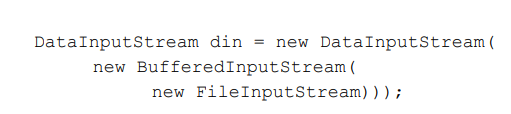
Рис.1.2 Результат роботи програми у файлі textRes.txt

Рис.2.2 Фрагмент згенерованої документації файлу Calculation

**Відповіді на контрольні запитання:**

1. Розкрийте принципи роботи з файловою системою засобами мови Java.

Для створення файлових потоків і роботи з ними у Java є 2 класи, що успадковані від InputStream і OutputStream це - FileInputStream і FileOutputStream. Як і їх суперкласи вони мають методи лише для байтового небуферизованого блокуючого читання/запису даних та керуванням потоками. На відміну від, наприклад, мови програмування С, де для виконання усіх можливих операцій з файлами необхідно мати один вказівник на FILE у мові Java реалізовано інший набагато складніший і гнучкіший підхід, який дозволяє формувати такі властивості потоку, які найкраще відповідають потребам рішення конкретної задачі. Так у Java розділено окремі функціональні можливості потоків на різні класи. Компонуючи ці класи між собою і досягається необхідна кінцева функціональність потоку. Так одні класи, як FileInputStream, забезпечують елементарний доступ до файлів, інші, як PrintWriter, надають додаткової функціональності по високорівневій обробці даних, що пишуться у файл. Ще інші, наприклад, BufferedInputStream забезпечують буферизацію. Таким чином, наприклад, щоб отримати буферизований файловий потік для читання інформації у форматі примітивних типів (char, int, double,…) слід створити потік з одночасним сумісним використанням функціональності класів FileInputStream, BufferedInputStream і DataInputStream. Для цього слід здійснити наступний виклик:



Класи типу BufferedInputStream, DataInputStream, PushbackInputStream (дозволяє читати з потоку дані і повертати їх назад у потік) успадковані від класу FilterInputStream. Вони виступають так званими фільтрами, що своїм комбінуванням забезпечують додаткову лише необхідну функціональність при читанні даних з файлу. Аналогічний підхід застосовано і при реалізації класів для обробки текстових даних, що успадковані від Reader і Writer.

1. Охарактеризуйте клас Scanner

Для читання текстових потоків найкраще підходить клас Scanner. На відміну від InputStreamReader і FileReader, що дозволяють лише читати текст, він має велику кількість методів, які здатні читати як рядки, так і окремі примітивні типи з подальшим їх перекодуванням до цих типів, робити шаблонний аналіз текстового потоку, здатний працювати без потоку даних та ще багато іншого. Приклад читання даних за допомогою класу Scanner з стандартного потоку вводу:



**Висновок:**

Під час виконання роботи я оволодів навиками використання засобів мови Java для роботи з потоками і файлами.