Міністерство освіти і науки України

Національний університет «Львівська політехніка»

Кафедра «Електронних обчислювальних машин»



Звіт

з лабораторної роботи № 5

з дисципліни: «Кросплатформенні засоби програмування»

на тему: «Файли у java»

|  |
| --- |
| **Виконав:** |
| студент групи КІ-306 |
| Ширий Б. І. |
| **Прийняв:** |
| доцент кафедри ЕОМ |
| Іванов Ю. С. |

Львів – 2023

# Методичні відомості роботи

## Мета

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 №4. Написати програму для тестування коректності роботи розробленого класу.

### №2

Для розробленої програми згенерувати документацію

### №3

Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

### №4

Скласти звіт про виконану роботу з приведенням тексту програми, результату її виконання та фрагменту згенерованої документації та завантажити його у ВНС.

### №5

Дати відповідь на контрольні запитання.

# Виконання лабораторної роботи

## Вихідний код

Написав програму, що реалізує метод обчислення виразу , код якої наведено у лістингу 2.1. Відповідно до завдання лабораторної роботи створив клас, що реалізує методи читання/запису у текстовому і двійковому форматах результатів роботи обчислювального класу, та навів його у лістингу 2.2.

*Лістинг 2.1. Код основної програми.*

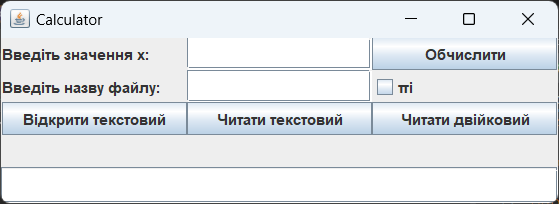
|  |
| --- |
| **package** CI\_306.Shyryi.Lab5;    **import** javax.swing.\*;  **import** java.awt.\*;  **import** java.awt.event.ActionEvent;  **import** java.awt.event.ActionListener;    ***/\*\****  ***\* A GUI application for an expression calculator.***  ***\*/***  **public** **class** ExpressionCalculatorGUI {  **private** **JTextField** xTextField;  **private** **JTextField** fileNameTextField;  **private** **JButton** calculateButton;  **private** **JButton** openTextButton;  **private** **JButton** readTextButton;  **private** **JButton** readBinaryButton;  **private** **JTextArea** resultTextArea;  **JCheckBox** checkBox = **new** **JCheckBox**("π");    ***/\*\****  ***\* Constructs the GUI for the expression calculator.***  ***\*/***  **public** ExpressionCalculatorGUI() {  **JFrame** frame = **new** **JFrame**("Calculator");  frame.setDefaultCloseOperation(**JFrame**.EXIT\_ON\_CLOSE);  frame.setLayout(**new** **BorderLayout**());  frame.setSize(460, 170);    **JPanel** inputPanel = **new** **JPanel**(**new** **GridLayout**(4, 2));  **JLabel** xLabel = **new** **JLabel**("Enter the value of x:");  xTextField = **new** **JTextField**();  **JLabel** fileNameLabel = **new** **JLabel**("Enter the file name:");  fileNameTextField = **new** **JTextField**();  calculateButton = **new** **JButton**("Calculate");  openTextButton = **new** **JButton**("Open Text");  readTextButton = **new** **JButton**("Read Text");  readBinaryButton = **new** **JButton**("Read Binary");    inputPanel.add(xLabel);  inputPanel.add(xTextField);  inputPanel.add(calculateButton);  inputPanel.add(fileNameLabel);  inputPanel.add(fileNameTextField);  inputPanel.add(checkBox);  inputPanel.add(openTextButton);  inputPanel.add(readTextButton);  inputPanel.add(readBinaryButton);    resultTextArea = **new** **JTextArea**();  resultTextArea.setEditable(**false**);    frame.add(inputPanel, **BorderLayout**.NORTH);  frame.add(**new** **JScrollPane**(resultTextArea), **BorderLayout**.CENTER);    calculateButton.addActionListener(**new** **ActionListener**() {  @**Override**  **public** **void** actionPerformed(**ActionEvent** e) {  calculateExpression();  }  });    openTextButton.addActionListener(**new** **ActionListener**() {  @**Override**  **public** **void** actionPerformed(**ActionEvent** e) {  **String** fileName = fileNameTextField.getText();  resultTextArea.setText(!fileName.isEmpty() ?  FileManager.openFile(fileName + ".txt") :  "Specify the name of the text file to open.");  }  });    readTextButton.addActionListener(**new** **ActionListener**() {  @**Override**  **public** **void** actionPerformed(**ActionEvent** e) {  **String** fileName = fileNameTextField.getText();  resultTextArea.setText(!fileName.isEmpty() ?  FileManager.readText(fileName + ".txt") :  "Specify the name of the text file to read.");  }  });    readBinaryButton.addActionListener(**new** **ActionListener**() {  @**Override**  **public** **void** actionPerformed(**ActionEvent** e) {  **String** fileName = fileNameTextField.getText();  resultTextArea.setText(!fileName.isEmpty() ?  FileManager.readBinary(fileName + ".dat") :  "Specify the name of the binary file to read.");  }  });    frame.setVisible(**true**);  }    ***/\*\****  ***\* Calculates the expression based on user input and writes the result to a file.***  ***\*/***  **private** **void** calculateExpression() {  **try** {  **double** x = **Double**.parseDouble(xTextField.getText());  **double** result = 1 / ((checkBox.isSelected() && ((**Math**.abs(4 \* x) == 0.5) ||  (((4 \* x - 0.5) % 1) == 0))) ?  0 : **Math**.cos(4 \* x));    **if** (result == **Double**.NaN  || result == **Double**.NEGATIVE\_INFINITY  || result == **Double**.POSITIVE\_INFINITY)  { **throw** **new** **ArithmeticException**(); }      **String** fileName = fileNameTextField.getText();  **if** (fileName.isEmpty()) {  fileName = "result";  }    FileManager.writeText(fileName + ".txt", x, result);  FileManager.writeBinary(fileName + ".dat", x, result);    resultTextArea.setText("Calculation result written to file '" + fileName + "'");      } **catch** (**NumberFormatException** e) {  resultTextArea.setText("Error: Invalid input format");  } **catch** (**ArithmeticException** e) {  resultTextArea.setText("Division by zero: cos(4x) equals zero.");  }  }    ***/\*\****  ***\* Entry point for the application.***  ***\****  ***\* @param args Command-line arguments (not used).***  ***\*/***  **public** **static** **void** main(**String**[] args) {  **SwingUtilities**.invokeLater(**new** **Runnable**() {  @**Override**  **public** **void** run() {  **new** ExpressionCalculatorGUI();  }  });  }  } |

*Лістинг 2.2. Клас "Файловий менеджер".*

|  |
| --- |
| **package** CI\_306.Shyryi.Lab5;    **import** java.io.BufferedReader;  **import** java.io.DataInputStream;  **import** java.io.DataOutputStream;  **import** java.io.File;  **import** java.io.FileInputStream;  **import** java.io.FileOutputStream;  **import** java.io.FileReader;  **import** java.io.FileWriter;  **import** java.io.IOException;  **import** java.io.PrintWriter;  **import** java.util.regex.Matcher;  **import** java.util.regex.Pattern;    ***/\*\****  ***\* A utility class for managing files and performing read/write operations.***  ***\*/***  **public** **class** FileManager {    ***/\*\****  ***\* Writes text data to a text file.***  ***\****  ***\* @param fileName The name of the file to write to.***  ***\* @param x The value of x.***  ***\* @param result The result to be written.***  ***\*/***  **public** **static** **void** writeText(**String** fileName, **double** x, **double** result) {  **try** (**PrintWriter** writer = **new** **PrintWriter**(**new** **FileWriter**(fileName))) {  writer.println("Значення y при x = " + x + " дорівнює " + result);  } **catch** (**IOException** e) {  **System**.err.println("Помилка запису в текстовий файл: " + e.getMessage());  }  }    ***/\*\****  ***\* Writes binary data to a binary file.***  ***\****  ***\* @param fileName The name of the file to write to.***  ***\* @param x The value of x.***  ***\* @param result The result to be written.***  ***\*/***  **public** **static** **void** writeBinary(**String** fileName, **double** x, **double** result) {  **try** (**DataOutputStream** outputStream = **new** **DataOutputStream**(**new** **FileOutputStream**(fileName))) {  outputStream.writeDouble(x);  outputStream.writeDouble(result);  } **catch** (**IOException** e) {  **System**.err.println("Помилка запису в двійковий файл: " + e.getMessage());  }  }    ***/\*\****  ***\* Reads text data from a text file.***  ***\****  ***\* @param fileName The name of the file to read from.***  ***\* @return A string containing the read data.***  ***\*/***  **public** **static** **String** readText(**String** fileName) {  **File** file = **new** **File**(fileName);  **if** (!file.exists()) {  **return** "Помилка: Файл не існує.";  }    **try** (**BufferedReader** reader = **new** **BufferedReader**(**new** **FileReader**(file))) {  **Pattern** pattern = **Pattern**.compile("[+-]?([0-9]\*[.])?[0-9]+");  **Matcher** matcher = pattern.matcher(reader.readLine());    **if** (matcher.find()) {  **String** numberStr1 = matcher.group();  **if** (matcher.find()) {  **String** numberStr2 = matcher.group();  **double** x = **Double**.parseDouble(numberStr1);  **double** result = **Double**.parseDouble(numberStr2);  **return** "У текстовому файлі: у = " + result + ", a x =" + x;  } **else** {  **return** "Помилка: Немає другого числа в першому рядку файла.";  }  } **else** {  **return** "Помилка: Немає першого числа в першому рядку файла.";  }  } **catch** (**IOException** | **NumberFormatException** e) {  **return** "Помилка читання з текстового файлу: " + e.getMessage();  }  }    ***/\*\****  ***\* Reads binary data from a binary file.***  ***\****  ***\* @param fileName The name of the file to read from.***  ***\* @return A string containing the read data.***  ***\*/***  **public** **static** **String** readBinary(**String** fileName) {  **File** file = **new** **File**(fileName);  **if** (!file.exists()) {  **return** "Помилка: Файл не існує.";  }    **try** (**DataInputStream** inputStream = **new** **DataInputStream**(**new** **FileInputStream**(file))) {  **double** x = inputStream.readDouble();  **double** result = inputStream.readDouble();  **return** "У бінарному файлі: у = " + result + ", a x =" + x;  } **catch** (**IOException** | **NumberFormatException** e) {  **return** "Помилка читання з двійкового файлу: " + e.getMessage();  }  }    ***/\*\****  ***\* Opens a file using the default system application.***  ***\****  ***\* @param fileName The name of the file to open.***  ***\* @return A string indicating the result of the operation.***  ***\*/***  **public** **static** **String** openFile(**String** fileName) {  **File** file = **new** **File**(fileName);  **if** (!file.exists()) {  **return** "Помилка: Файл не існує.";  }    **try** {  java.awt.Desktop.getDesktop().open(file);  } **catch** (**IOException** e) {  **return** "Помилка відкриття файлу: " + e.getMessage();  }  **return** "Файл " + fileName + " відкрито";  }  } |

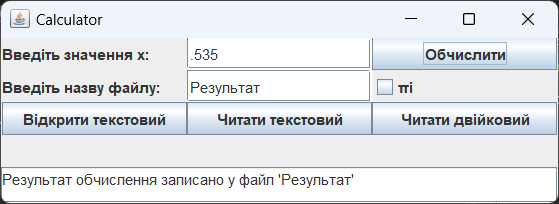
## Результати виконання

Початковий вигляд програми наведено на рисунку 2.1.



*Рисунок 2.1. Початковий вигляд програми.*

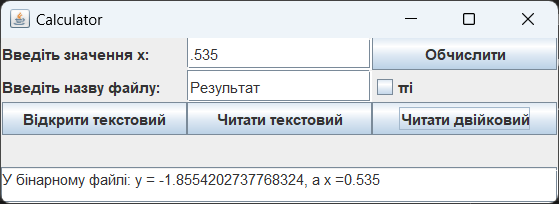
Якщо ми обрахуємо певне значення (рисунок 2.2), то через клас «Файловий менеджер» відбудеться запис у текстовий та бінарний файли.



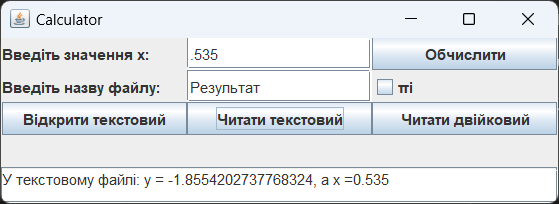
*Рисунок 2.2. Запис у файли.*

Відповідно, ми можемо:

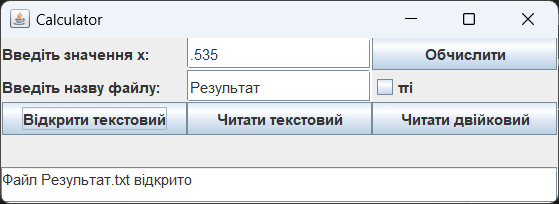
* Прочитати бінарний файл - рисунок 2.3,
* Прочитати текстовий файл - рисунок 2.4,
* Відкрити текстовий файл - рисунки 2.5 та 2.6.



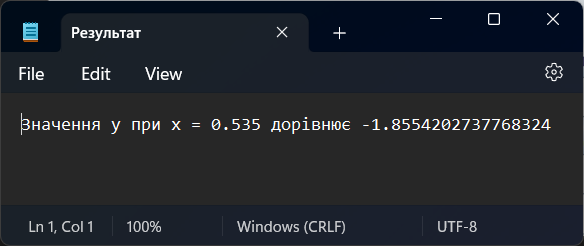
*Рисунок 2.3. Читання з бінарного файлу.*



*Рисунок 2.4. Читання з текстового файлу.*



*Рисунок 2.5. Відкриття текстового файлу.*



*Рисунок 2.6. Відкритий текстовий файл.*

## Документація

Згенерував документацію, фрагмент якої навів на рисунку 2.7.

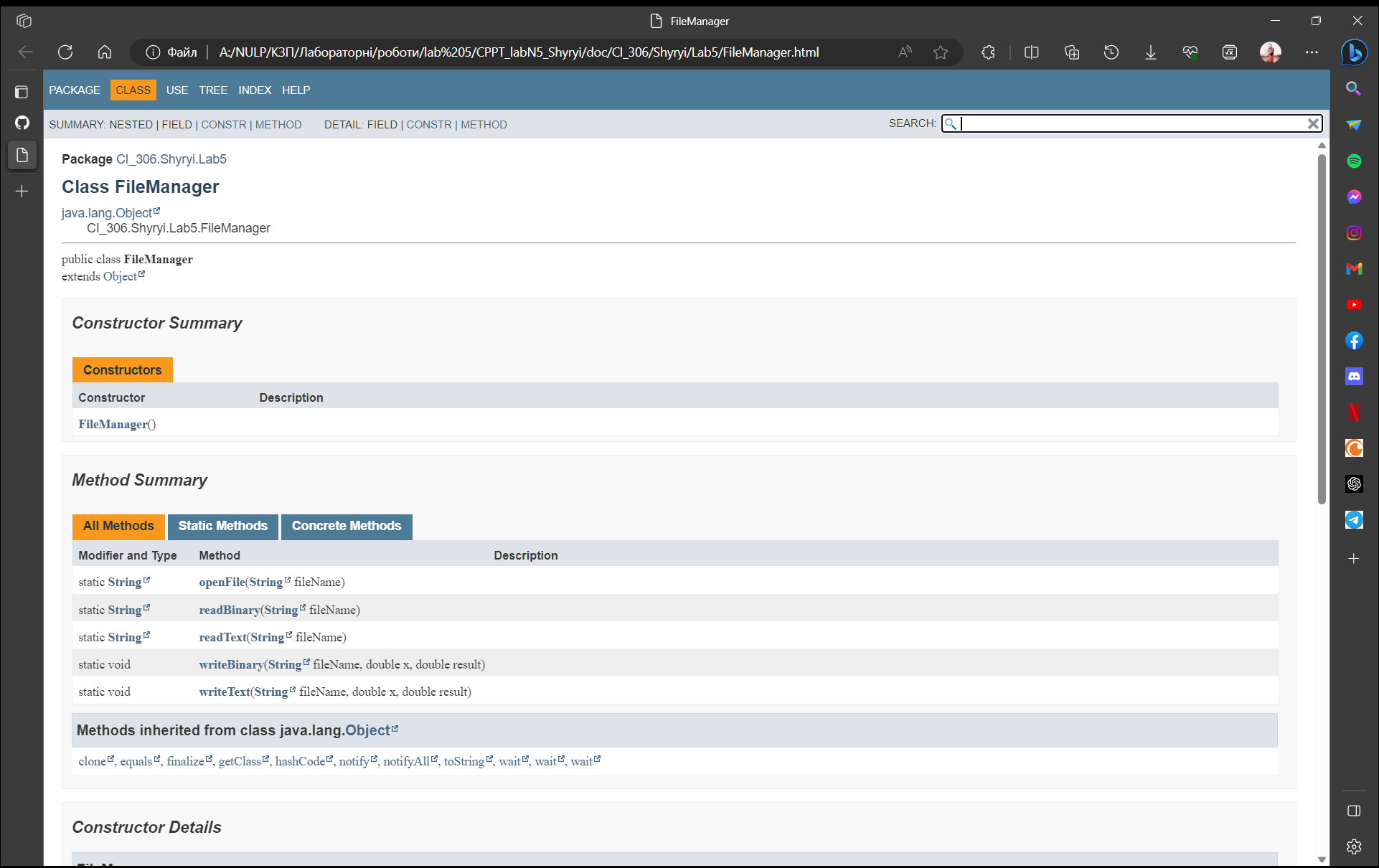


Рисунок 2.7. Фрагмент згенерованої документації.

## Відповіді на контрольні питання

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

Принципи роботи з файловою системою в мові Java реалізуються через класи, такі як File, FileInputStream, FileOutputStream, FileReader, і FileWriter. За допомогою цих класів можна створювати, читати, записувати і видаляти файли та директорії.

### Охарактеризуйте клас Scanner.

Клас Scanner використовується для зчитування даних з різних вхідних джерел, таких як клавіатура або файли. Він надає методи для зчитування примітивних типів даних та рядків.

### Наведіть приклад використання класу Scanner.

Приклад використання класу Scanner для зчитування цілого числа з консолі наведено у лістингу 2.3.

*Лістинг 2.3.*

|  |
| --- |
| **import** java.util.Scanner;    **public** **class** ScannerExample {  **public** **static** **void** main(**String**[] args) {  **Scanner** scanner = **new** **Scanner**(**System**.in);  **System**.out.print("Введіть ціле число: ");  **int** number = scanner.nextInt();  **System**.out.println("Ви ввели: " + number);  scanner.close();  }  } |

### За допомогою якого класу можна здійснити запис у текстовий потік?

Запис у текстовий потік можна здійснити за допомогою класу FileWriter.

### Охарактеризуйте клас PrintWriter.

Клас PrintWriter є підкласом Writer і використовується для запису даних у текстовий потік з більш зручними методами, які автоматично перетворюють різні типи даних в рядки.

### Розкрийте методи читання/запису двійкових даних засобами мови Java.

Для читання/запису двійкових даних можна використовувати класи FileInputStream, FileOutputStream, DataInputStream, і DataOutputStream.

### Призначення класів DataInputStream і DataOutputStream.

Класи DataInputStream і DataOutputStream призначені для читання і запису примітивних типів даних, а також рядків у двійковому форматі.

### Який клас мови Java використовується для здійснення довільного доступу до файлів.

Для здійснення довільного доступу до файлів можна використовувати клас RandomAccessFile.

### Охарактеризуйте клас RandomAccessFile.

Клас RandomAccessFile дозволяє зчитувати і записувати дані в файлі на випадкових позиціях, а не послідовно.

### Який зв’язок між інтерфейсом DataOutput і класом DataOutputStream?

Інтерфейс DataOutput визначає методи для запису примітивних типів даних у двійковому форматі, а клас DataOutputStream реалізує цей інтерфейс і додає методи для запису рядків.

## Висновок

Протягом лабораторної роботи, було реалізовано клас для читання/запису результатів роботи класу, який був розроблений під час попередньої лабораторної роботи. Для цього були використані відповідні клас та методи, зазначені в завданні.

Далі була згенерована документація для коду за допомогою Javadoc, і весь код разом із згенерованою документацією був завантажений на GitHub, де його можна знайти відповідно до методичних вказівок щодо роботи з цим сервісом.

Звіт про виконану роботу містить текст програми, результати її виконання (зразки введених та виведених даних), фрагмент згенерованої документації та іншу інформацію, яка була вказана у завданні.