Name: Sarwan Heer

CodeHS

Engineering Project U1

Experiment.java

```
/*************************************************************
* File name:
* Experiment.java
*
* Description:
* This Java code creates a Swing GUI application named "Lab Calculator"
* that allows users to
* read data from a file,
* analyze it,
* clear the input fields,
* and exit the application.
*
* Author: Sarwan Heer
*
* Date: March 8, 2024
*
* Concepts:
1. **Swing GUI Components**: The code utilizes various Swing components such as `JFrame`, `JLabel`, `JTextField`, `JButton`, and `JTextArea` to create a graphical user interface.

2. **Layout Management**: The layout of the `JFrame` is set to null (`setLayout(null)`) to allow manual positioning of components.

3. **Event Handling**: Action listeners are used to handle user interactions with buttons. Lambda expressions are employed to provide concise implementations of action listener interfaces.

4. **File Input/Output**: The code reads data from a file using the `Scanner` class (`Scanner sc = new Scanner(Experiment.class.getResourceAsStream(txtFile.getText()))`) and displays the content in the GUI.

5. **Resource Loading**: Files are loaded as resource streams using `Experiment.class.getResourceAsStream(txtFile.getText())`.

6. **String Manipulation**: String concatenation and manipulation are used to construct and display data read from the file.

7. **Control Flow**: The code utilizes conditional statements (`if` and `else`) to determine the appropriate action based on user input.

8. **Class Instantiation**: Instances of Swing components (`JFrame`, `JLabel`, `JTextField`, `JButton`, `JTextArea`) are created and added to the GUI.

9. **GUI Interaction**: The code provides functionality for users to interact with the GUI by entering text, clicking buttons, and viewing output.

10. **Resource Management**: The `Scanner` instances are properly closed using the `close()` method to release system resources.

These concepts collectively enable the creation of a functional GUI application for reading, analyzing, and displaying data from files within a graphical environment.

*************************************************************/
import javax.swing.*;
import java.util.Scanner;

public class Experiment {

    public static void main(String[] args) {

        JFrame frmMain = new JFrame();
        frmMain.setSize(500, 500);
        frmMain.setLayout(null);


        JLabel lblTitle = new JLabel("Lab Calculator");
        lblTitle.setBounds(200, 30, 200, 30);
        frmMain.add(lblTitle);

        JTextField txtFile = new JTextField();
        txtFile.setBounds(90, 100, 270, 40);
        frmMain.add(txtFile);

        JLabel lblFile = new JLabel("File Name:");
        lblFile.setBounds(10, 70, 100, 100);
        frmMain.add(lblFile);

        JButton btnRead = new JButton("Read File");
        btnRead.setBounds(380, 105, 100, 30);
        frmMain.add(btnRead);
```

```
68.
69.         JTextArea txtInfo = new JTextArea();
70.         txtInfo.setBounds(90, 150, 270, 140);
71.         frmMain.add(txtInfo);
72.
73.         JButton btnAnalyze = new JButton("Analyze");
74.         btnAnalyze.setBounds(180, 300, 100, 30);
75.         frmMain.add(btnAnalyze);
76.
77.         JTextArea txtOutput = new JTextArea();
78.         txtOutput.setBounds(90, 350, 270, 60);
79.         frmMain.add(txtOutput);
80.
81.         JButton btnClear = new JButton("Clear");
82.         btnClear.setBounds(10, 430, 90, 30);
83.         frmMain.add(btnClear);
84.
85.         JButton btnExit = new JButton("Exit");
86.         btnExit.setBounds(390, 430, 90, 30);
87.         frmMain.add(btnExit);
88.
89.         btnRead.addActionListener(e -> {
90.             String material, quantity1, variable1, unit1, quantity2, variable2, unit2;
91.             double value1 = 0, value2 = 0, value3 = 0, value4 = 0;
92.
93.             Scanner sc = new Scanner(
94.             Experiment.class.getResourceAsStream(txtFile.getText())
95.             );
96.
97.             sc.nextLine();
98.             sc.findInLine("material");
99.             material = sc.nextLine();
100.            sc.findInLine("quantity1");
101.            quantity1 = sc.next();
102.            sc.findInLine("variable1");
103.            variable1 = sc.next();
104.            sc.findInLine("unit1");
105.            unit1 = sc.next();
106.
107.            sc.nextLine();
108.            value1 = sc.nextDouble();
109.            value2 = sc.nextDouble();
110.
111.            sc.nextLine();
112.            sc.findInLine("quantity2");
113.            quantity2 = sc.next();
114.            sc.findInLine("variable2");
115.            variable2 = sc.next();
116.            sc.findInLine("unit2");
117.            unit2 = sc.next();
118.
119.            sc.nextLine();
120.            value3 = sc.nextDouble();
121.            value4 = sc.nextDouble();
122.
123.            String data = material + "\n" + quantity1 + " " + variable1 + " " + "(" + unit1 + ")"
124.                                   + "   " + quantity2 + " " + variable2 + " " + "(" + unit2 + ")" + "\n"
125.                                   + value1 + "                      " + value3 + "\n"
126.                                   + value2 + "                   " + value4;
127.
128.            txtInfo.setText(data);
129.
130.            sc.close();
131.
132.        });
133.
134.        btnAnalyze.addActionListener(e -> {
135.            String quantity, variable, unit1, unit2;
136.
137.              Scanner sc = new Scanner(
138.            Experiment.class.getResourceAsStream(txtFile.getText())
139.                   );
140.
141.            double firstValue = 0, secondValue = 0;
142.
143.            sc.findInLine("quantity");
144.            quantity = sc.next();
145.            sc.findInLine("variable");
146.            variable = sc.next();
147.
148.            sc.nextLine();
149.            sc.nextLine();
150.            sc.findInLine("unit1");
151.            unit1 = sc.next();
152.            sc.nextLine();
153.            firstValue = sc.nextDouble();
154.            firstValue += sc.nextDouble();
155.            firstValue /= 2;
156.
157.            sc.nextLine();
158.            sc.findInLine("unit2");
159.            unit2 = sc.next();
```

```
160.             sc.nextLine();
161.             secondValue = sc.nextDouble();
162.             secondValue += sc.nextDouble();
163.             secondValue /= 2;
164.
165.             String calculations = quantity + " " + variable + " = "
166.                                 + (firstValue / secondValue)
167.                                 + " " + unit1 + "/" + unit2;
168.
169.             txtOutput.setText(calculations);
170.
171.             sc.close();
172.         });
173.
174.         btnExit.addActionListener(e -> {
175.             System.exit(0);
176.         });
177.
178.         btnClear.addActionListener(e -> {
179.             txtFile.setText("");
180.             txtOutput.setText("");
181.             txtInfo.setText("");
182.         });
183.
184.         frmMain.setVisible(true);
185.     }
186. }
```

data/.

```
1.
```

data/experiment1.txt

```
1. ratio quantity density ratio variable d
2. material ice
3. quantity1 mass variable1 m unit1 g
4. 2.3 2.45
5. quantity2 volume variable2 v unit2 cm^3
6. 1.9 1.98
```

data/experiment2.txt

```
1. ratio quantity speed ratio variable v
2. material car speed
3. quantity1 distance variable1 d unit1 m
4. 2.3 2.45
5. quantity2 time variable2 t unit2 s
6. 1.9 1.98
```

data/experiment3.txt

```
1. ratio quantity Resistance ratio variable R
2. material Ohm's law
3. quantity1 Voltage variable1 V unit1 V
4. 4.0 6.0
5. quantity2 Current variable2 I unit2 A
6. 1.7 1.3
```