Experiment No.___3 *Date___/___/2020*

TITLE OF EXPERIMENT: - A Program to Develop a GUI which accepts the information regarding the marks for all the subjects of a student in the examination. Display the result for a student in a separate window.

DIVISION:_____ **BRANCH:** _____

BATCH:_____ **ROLL NO.:** _____

PERFORMED ON DATE: _____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 3

Aim: Develop a GUI which accepts the information regarding the marks for all the subjects of a student in the examination. Display the result for a student in a separate window.

Software:

1. Command prompt
2. JDK 8
3. Applet viewer

Theory:

To understand this Program you should have the knowledge of following concepts of Java:

- Java For Loop
- Arrays in Java
- if..else-if in Java

Swing is a part of the **JFC (Java Foundation Classes)**. Building Graphical User Interface in Java requires the use of Swings. Swing Framework contains a large set of components which allow a high level of customization and provide rich functionalities, and is used to create window-based applications. Java swing components are lightweight, platform-independent, provide powerful components like tables, scroll panels, buttons, list, colour chooser, etc.
In this article, we will see how to write the students information in a Jframe and store it in a file.

Approach: To write this program, we need to follow some steps, that are as follow:

1. First, we need to create a frame using **JFrame.**
2. Next create **JLabels, JTextFields, JComboBoxes, JButtons** and set their bounds respectively.
3. Name these components accordingly and set their bounds.
4. Now, in order to **save the data into the text file** on button click, we need to add Event Handlers. In this case, we will **add ActionListener** to perform an action method known as actionPerformed in which first we need to get the values from the text fields which is default as a "string".
5. Finally, the **Jbuttons, JLabels, JTextFields and JComboBoxes** are added to the JFrame and the text is stored in a text file.

Java Integer parseInt (CharSequence s, int beginText, int endText, int radix)

This method parses the **CharSequence** argument as a **signed** integer in the specified **radix** argument, beginning at the specified **beginIndex** and extending to **endIndex - 1**. This method does not take steps to guard against the CharSequence being mutated while parsing.

Syntax:

Following are the declarations of **parseInt ()** method:

1. **public static int** parseInt (String s)
2. **public static int** parseInt (String s, **int** radix)
3. **public static int** parseInt (CharSequence s, **int** beginIndex, **int** endIndex, **int** radix)

Parameter:

DataType	Parameter	Description	Required/Optional
String	s	It is a String which needs to be converted into the Integer equivalent.	Required
int	radix	The radix to be used while parsing the String	Required
int	beginIndex	The beginning index, inclusive.	Required

int	endIndex	The ending index, exclusive.	Required
CharSequence	s	It is the CharSequence which needs to be converted into the Integer equivalent.	Required

Returns:

Method	Returns
parseInt (String s)	This method returns the integer value which is represented by the argument in decimal equivalent.
parseInt (String s, int radix)	This method returns the integer value which is represented by the string argument in the specified radix.
parseInt (String s, int radix)	This method returns the integer value which is represented by the string argument in the specified radix.

Exceptions:

NullPointerException: If s is null.

IndexOutOfBoundsException: If beginIndex is negative, or if beginIndex is greater than endIndex or if endIndex is greater than s.length ().

NumberFormatException: If the CharSequence does not contain a parsable int in the specified radix, or if radix is either smaller than Character.MIN_RADIX or larger than Character.MAX_RADIX.

Compatibility Version:

Java 1.2 and above:

- Java Integer parseInt (String s)
- Java Integer parseInt (String s, int radix)

Java 9:

- Java Integer parseInt (CharSequence s, int beginText, int endText, int radix)

Example

```
public class IntegerParseIntExample1 {
   public static void main(String[] args) {
      int decimalExample = Integer.parseInt("20");
                  int signedPositiveExample = Integer.parseInt("+20");
                  int signedNegativeExample = Integer.parseInt("-20");

               System.out.println("Value = "+decimalExample);
               System.out.println("Value = "+signedPositiveExample);
               System.out.println("Value = "+signedNegativeExample);
      }
  }
```

Output:

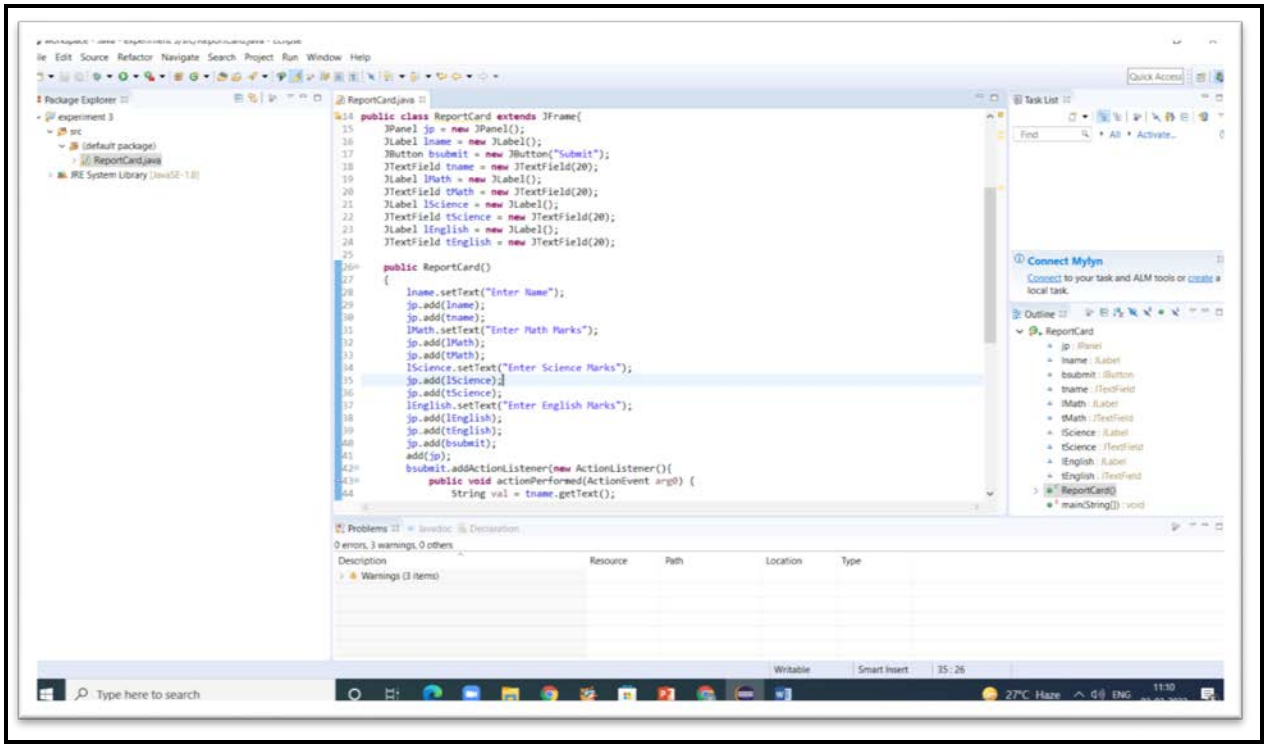
```
Value = 20
Value = 20
Value = -20
```

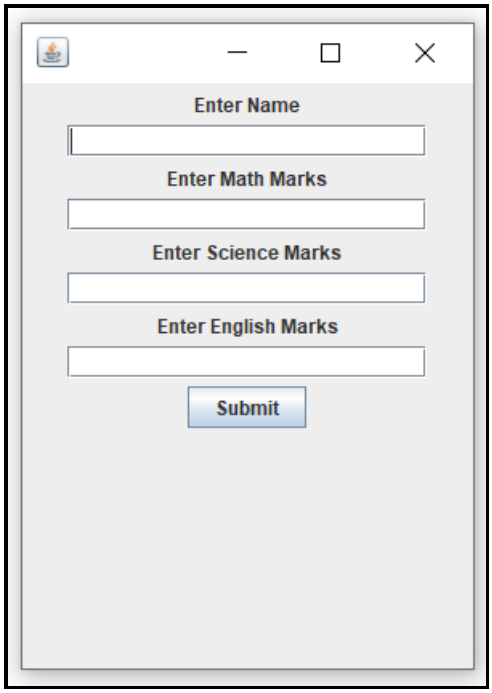
Program:

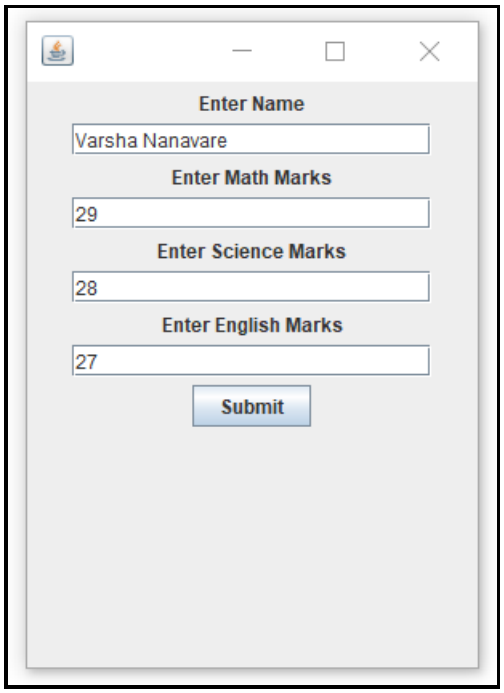
```
import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JTextField;

public class ReportCard extends JFrame{
       JPanel jp = new JPanel();
       JLabel lname = new JLabel();
```

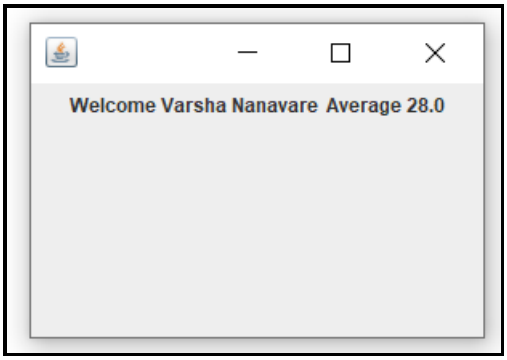
```
	JButton bsubmit = new JButton("Submit");
	JTextField tname = new JTextField(20);
	JLabel lMath = new JLabel();
	JTextField tMath = new JTextField(20);
	JLabel lScience = new JLabel();
	JTextField tScience = new JTextField(20);
	JLabel lEnglish = new JLabel();
	JTextField tEnglish = new JTextField(20);

	public ReportCard()
	{
		lname.setText("Enter Name");
		jp.add(lname);
		jp.add(tname);
		lMath.setText("Enter Math Marks");
		jp.add(lMath);
		jp.add(tMath);
		lScience.setText("Enter Science Marks");
		jp.add(lScience);
		jp.add(tScience);
		lEnglish.setText("Enter English Marks");
		jp.add(lEnglish);
		jp.add(tEnglish);
		jp.add(bsubmit);
		add(jp);
		bsubmit.addActionListener(new ActionListener(){
			public void actionPerformed(ActionEvent arg0) {
				String val = tname.getText();
				JLabel l1 = new JLabel("Welcome "+val);
				int sub1 = Integer.parseInt(tMath.getText());
				int sub2 = Integer.parseInt(tScience.getText());
				int sub3 = Integer.parseInt(tEnglish.getText());
				int sum = sub1+sub2+sub3;
				float average = sum/3;
				JLabel l2 = new JLabel("Average "+ average);
				JPanel jip = new JPanel();
				jip.add(l1);
				jip.add(l2);
				JFrame inf = new JFrame();
				inf.setVisible(true);
				inf.add(jip);
				inf.setSize(300, 100);
			}

		});

	}

	public static void main(String[] args) {
		ReportCard rc = new ReportCard();
		rc.setSize(300, 200);
		rc.setVisible(true);

	}

}
```

Screenshot's of Output:









Conclusion: