







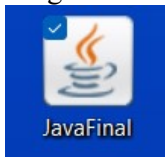


Criterion C: Development

All classes

- >  AdderMethods.java
- >  CalcMethod.java
- >  EndGui.java
- >  GasGui.java
- >  GuiIB.java
- >  Introgui.java
- >  SplitGui.java
- >  TipGui.java

Program file

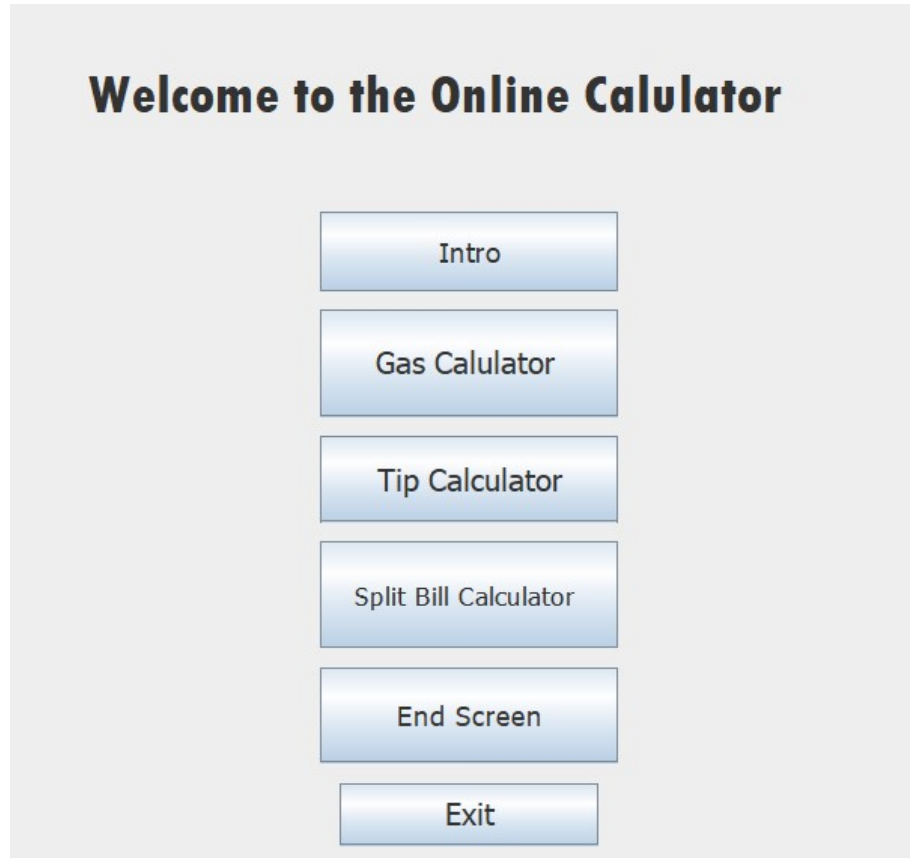


List of techniques

1. Static methods
2. Methods created by me that allow to save of the values entered the user
3. All classes use GUI and have a relation to GuiIB which allows for the classes to be connected in relations to each other
4. String Array list and double that stores the values in the end of each program when the user saves them by a button.¹

¹ PeterPefiPeterPefi 8711 silver badge66 bronze badges, AmrDeveloperAmrDeveloper 3 and Cheng ThaoCheng Thao 1 (1968) *Storing objects of multiple classes in an array list, then accessing their attributes*, *Stack Overflow*. Available at: <https://stackoverflow.com/questions/70411989/storing-objects-of-multiple-classes-in-an-array-list-then-accessing-their-attri>.

Intro Menu



```
JLabel lblNewLabel = new JLabel("Welcome to the Online Calculator");

lblNewLabel.setFont(new Font("Tw Cen MT Condensed Extra Bold", Font.PLAIN, 30));

lblNewLabel.setBounds(51, 33, 455, 57);

contentPane.add(lblNewLabel);

JButton btnNewButton_1 = new JButton("Gas Calculator ");

btnNewButton_1.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        GasGui sf = new GasGui();

        sf.show(); // show intro gui

        dispose(); // closes main frame

    }

});

btnNewButton_1.setFont(new Font("Tahoma", Font.PLAIN, 16));
```

```

btnNewButton_1.setBounds(175, 176, 158, 57);

contentPane.add(btnNewButton_1);

JButton btnNewButton_2 = new JButton("Split Bill Calculator ");

btnNewButton_2.setFont(new Font("Tahoma", Font.PLAIN, 14));

btnNewButton_2.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        SplitGui sf = new SplitGui();

        sf.show(); //

        dispose(); // closes main frame

    }

});

btnNewButton_2.setBounds(175, 299, 158, 57);

contentPane.add(btnNewButton_2);

JButton btnNewButton_3 = new JButton("Tip Calculator");

btnNewButton_3.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        TipGui sf = new TipGui();

        sf.show(); //

        dispose(); // closes main frame

    }

});

btnNewButton_3.setFont(new Font("Tahoma", Font.PLAIN, 16));

btnNewButton_3.setBounds(175, 243, 158, 46);

contentPane.add(btnNewButton_3);

JButton btnNewButton_4 = new JButton("End Screen");

btnNewButton_4.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

```

```

EndGui sf = new EndGui();

sf.show(); //

dispose(); // closes main frame

}

});

btnNewButton_4.setFont(new Font("Tahoma", Font.PLAIN, 15));

btnNewButton_4.setBounds(175, 366, 158, 51);

contentPane.add(btnNewButton_4);

JButton btnNewButton = new JButton("Intro");

btnNewButton.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        Introgui sf = new Introgui();

        sf.show(); //

        dispose(); // closes main frame

    }

});

btnNewButton.setFont(new Font("Tahoma", Font.PLAIN, 15));

btnNewButton.setBounds(175, 124, 158, 42);

contentPane.add(btnNewButton);

JButton btnNewButton_5 = new JButton("Exit");

btnNewButton_5.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        dispose();

    }

});

btnNewButton_5.setFont(new Font("Tahoma", Font.PLAIN, 16));

btnNewButton_5.setBounds(185, 428, 138, 33);

```

```

contentPane.add(btnNewButton_5);

}

}

```

In these lines of code the user can switch between different classes for different calculators, they use buttons to close the current class and open a new class. This allows for the user to go in order or any order when using the app, the classes are not depended on each other, they are depended on the main gui.

The Exit button ends the program which is located on the main gui class.

The GasGui

```

public void actionPerformed(ActionEvent e) {

if (Gassbutton1.isSelected())

{

value = 1;

GassButton2.setSelected(false);

GassButton3.setSelected(false);

}
}

```

```

}

});

Gassbutton1.setFont(new Font("Tahoma", Font.PLAIN, 15));

Gassbutton1.setBounds(6, 107, 192, 21);

contentPane.add(Gassbutton1);

GassButton2.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        if (GassButton2.isSelected())

        {

            value = 2;

            Gassbutton1.setSelected(false);

            GassButton3.setSelected(false);

        }

    }

});

GassButton2.setFont(new Font("Tahoma", Font.PLAIN, 15));

GassButton2.setBounds(6, 130, 159, 21);

contentPane.add(GassButton2);

GassButton3.addActionListener(new ActionListener() {

    public void actionPerformed(ActionEvent e) {

        if (GassButton3.isSelected())

        {

            value = 3;

            GassButton2.setSelected(false);

            GassButton3.setSelected(false);

        }

    }

}

```

```
});
```

The button selection here allows for a user to not selected multiple buttons which would mess up their order because if there were not if statements then the program would have multiple values which the user did not select would affect the whole program. During clients time the calculator uses methods from the AdderMethod class to save values.

```
if (combocheck > 0)
```

```
{
```

```
    adder.setNumber(ival);
```

```
    String a = "In the Gas Calcualtor you had a membership of " + val1 + " and your price came out as" + ival;
```

```
    adder.addString(a);
```

```
}
```

```
else
```

```
{
```

```
    adder.setNumber(ival);
```

```
    adder.addString("In the Gas Calcualtor your price came out as" + ival);
```

```
}
```

In certain selections from the user the program adjust to it. By using if-else

In the TipGui class

```
try
```

```
{
```

```
    tipin = Integer.parseInt(textField_1.getText());
```

```
}
```

```
catch (Exception e1)
```

```
{
```

```
    JOptionPane.showMessageDialog(null, "You cannot enter a value with a %  
", "ERROR!", JOptionPane.INFORMATION_MESSAGE);
```

```
}
```

```
if (tipin > 500)
```

```
{
```

```
    JOptionPane.showMessageDialog(null, "You entered a value higher than 500, please
```

```
redo", "ERROR!", JOptionPane.INFORMATION_MESSAGE);
}
```



This try catch statement catches a part of button which will show a message saying that the user should not enter a \$ when entering money into a Jtextbox, this saves the program from running into syntax errors.

Along with this, it stops the user from entering a value higher than 500 because the user may have accidentally inputted a high number rather than what they wanted to do.

```
JButton btnNewButton_2_1 = new JButton("Clear\r\n");
```

```
btnNewButton_2_1.addActionListener(new ActionListener() {
```

```
public void actionPerformed(ActionEvent e) {
```

```
val1 = "";
```

```
inval = 0;
```

```
Gassbutton1.setSelected(false);
```

```
GassButton2.setSelected(false);
```

```
GassButton3.setSelected(false);
```

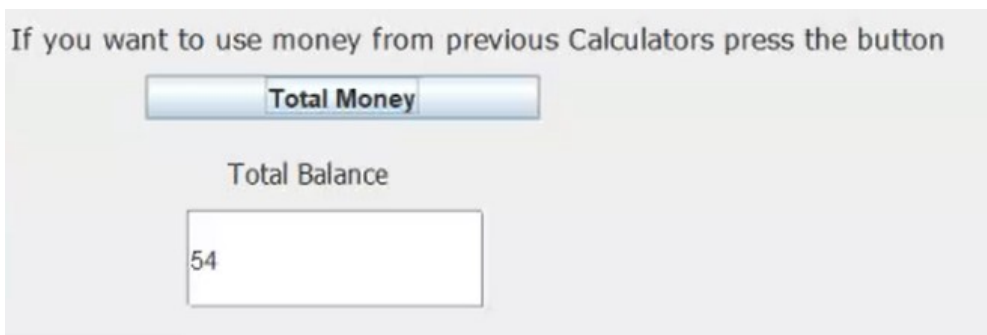
```
textField_1.setText("");
```

```
comboBox.setSelectedIndex(0);
```

```
textField.setText("");
```

```
}
```

The clear button respective in each class menu allows for the user to clear the options that they have selected if they had made a mistake or want to redo what they have completed, this allows for the user to go fix any mistakes they had during their first run of the program.



```
JButton btnNewButton = new JButton("Total Money");
```

```
btnNewButton.addActionListener(new ActionListener() {
```



```

public void actionPerformed(ActionEvent e) {

    invalnew = adder.getNumber();

    String strNumber = Double.toString(invalnew);

    textField.setText((strNumber));

}

```

When the user presses the Total money button inside the split Gui this allows for the user to call a method from the AdderMethods, which saved double from previous values and allows for them to not have to worry about a value being lost and can properly input the value in. Else the user can enter their own value if they did not use the other calculators

```

public class AdderMethods {

    private ArrayList<String> list;

    public AdderMethods() {

        list = new ArrayList<>();

    }

    public ArrayList<String> getList() {

        return list;

    }

    public void addString(String str) {

        list.add(str);

    }

    public double number;

    public void setNumber(double newNumber) {

        number = newNumber;

    }

    public double getNumber() {

        return number;

    }

    public void addToNumber(double valueToAdd) {

        number += valueToAdd;

    }

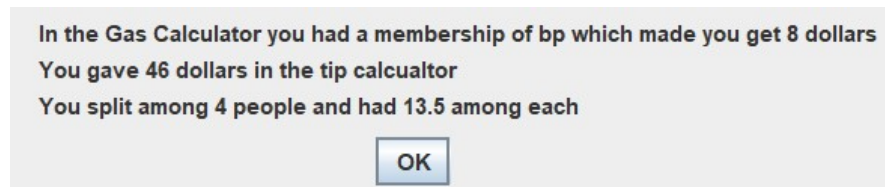
}

```

These methods are used to save the values that the user inputted previously inside the program. Due to this the user does not have to continually remember the values that they have gotten if they were going to use all parts of the program or only use one part. This also allows for the user to see the methods he/she used during her time in the program by saving what they selected.²

```
for (String Olda : list) {  
    Stral = Stral + "\n" + Olda;  
}  
  
JOptionPane.showMessageDialog(null, Stral, "", JOptionPane.INFORMATION_MESSAGE);  
}
```

This calls the Array storing method and then runs it through a loop to produce the final output of the code. By this, it properly gets the values that the user selected and tells them a brief recap of what they have selected.



```
import java.text.DecimalFormat;  
  
public class CalcMethod {  
  
    public static double GasGalin (int check, double numbers)  
  
    {  
  
        double newnum1 =0;  
  
        double newnum = 0;  
  
        if (check == 1)  
  
        {  
  
            newnum1 = 3.40 * numbers;  
  
            DecimalFormat df = new DecimalFormat("#.##");
```

² PeterPefiPeterPefi 8711 silver badge66 bronze badges, AmrDeveloperAmrDeveloper 3 and Cheng ThaoCheng Thao 1 (1968) *Storing objects of multiple classes in an array list, then accessing their attributes*, Stack Overflow. Available at: <https://stackoverflow.com/questions/70411989/storing-objects-of-multiple-classes-in-an-array-list-then-accessing-their-attri>.

```

String formatted = df.format(newnum1);

double newnum11 = Double.parseDouble(formatted);

return newnum11;

}

else if (check ==2)

{

newnum1 = 3.60 * numbers;

DecimalFormat df = new DecimalFormat("#.##");

String formatted = df.format(newnum1);

double newnum11 = Double.parseDouble(formatted);

return newnum11;

}

else if (check ==3)

{

newnum1 = 3.90 * numbers;

DecimalFormat df = new DecimalFormat("#.##");

String formatted = df.format(newnum1);

double newnum11 = Double.parseDouble(formatted);

return newnum11;

}

return newnum;

}

public static double moneygive (double tipin, double moneyin)

{

double newmoney = 0;

newmoney = (tipin * moneyin) +moneyin;

DecimalFormat df = new DecimalFormat("#.##");

String formatted = df.format(newmoney);

```

```
double result = Double.parseDouble(formatted);  
  
return result;  
  
}  
  
}
```

The method here is the critical part of the program as it takes in values that the user has entered from parameters then gives them back with 2 decimal places. This allows for the client to be satisfied with her need of having 2 decimal places.

Words 1,187