In this project you will build a Java GUI application for evaluating post-fix arithmetic expressions. This application uses a stack data structure to hold the intermediate results of the RPN calculations.

Introduction - An RPN Expression is also sometimes called a post-fix expression. RPN expression are evaluated by scanning them in a left-to-right order. The operands are placed to the left of the operation being performed on them. So the infix expression

\[ X + Y \]

would become

\[ XY+ \]

and the expression

\[ (X+Y)/Z - (X-Y)*Z \]

would become

\[ XY+Z/XY-Z*- . \]

We can use a stack data structure to hold the intermediate values as we evaluate an RPN expression. The rules for RPN evaluation are:

1. Scan the expression one character (or token) at a time from the left to the right.
2. When an operand is encountered, push the corresponding value onto the stack.
3. When an operator is encountered, pop the top two values off the stack, perform the indicated operation, and push the result back onto the stack.
4. Repeat steps (1) through (3) above until expression is completely scanned. The value on the top of the stack is the result.

Before we attempt to build our program, let's try out this procedure by hand, using the previous example postfix expression. The problem statement says that we can assume that the operands (variables) will be represented by single letters A through Z. This will simplify our expression scanning procedure. Let's assume that we have values associated with our operands as we practice. Assume that \( X=1, Y=2, \) and \( Z=-1 \) as you evaluate the RPN expression, \( XY+Z/XY-Z*- . \)

<table>
<thead>
<tr>
<th>( XY+Z/XY-Z*- )</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>2</td>
</tr>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>3</td>
</tr>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>-1</td>
</tr>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>-3</td>
</tr>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>1/3</td>
</tr>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>2/1</td>
</tr>
<tr>
<td>( XY+Z/XY-Z*- )</td>
<td>-1/3</td>
</tr>
<tr>
<td>push the value of X onto the stack</td>
<td></td>
</tr>
<tr>
<td>push the value of Y onto the stack</td>
<td></td>
</tr>
<tr>
<td>pop the top two values off the stack and push the sum (1+2)</td>
<td></td>
</tr>
<tr>
<td>push the value of Z onto the stack</td>
<td></td>
</tr>
<tr>
<td>pop the top two values and push the ratio (3/[-1])</td>
<td></td>
</tr>
<tr>
<td>push the value of X onto the stack</td>
<td></td>
</tr>
<tr>
<td>push the value of Y onto the stack</td>
<td></td>
</tr>
<tr>
<td>pop the top two values and push the difference (1-2)</td>
<td></td>
</tr>
</tbody>
</table>
Therefore, the result of this expression is the value remaining on the top of the stack (i.e. -4). Besides performing these operations properly, we may want to include error checking in our program. For example, there always must be at least one more operand encountered than operators as we scan an RPN expression from left to right (why?).

Creating a Netbeans Project - When building a GUI application we first create the "look and feel" of the application. This is the part that the user sees and interacts with. Choose an appropriate name for your project (we will use RPNCalculator here) and then complete the following steps:

(1) Choose File > New Project. Alternatively, you can click the New Project icon in the IDE toolbar.

(2) In the Categories pane, select the Java node. In the Projects pane, choose Java Application. Click Next.

(3) Type RPNCalculator in the Project Name field and specify a path, for example, in your home directory, as the project location.

(4) Deselect the Create Main Class checkbox if it is selected.

(5) Click Finish.

Building the GUI - To proceed with building our interface, we need to create a Java container within which we will place the other required GUI components. In this step we’ll create a container using the JFrame component. We will place the container in a new package, which will appear within the Source Packages node.

Create a JFrame container

(1) In the Projects window, right-click the RPNCalculator node and choose New > Other.

(2) In the New File dialog box, choose the Swing GUI Forms category and the JFrame Form file type. Click Next.

(3) Enter RPNCalculatorUI as the class name.

(4) Enter my.rpncalculator as the package.

(5) Click Finish.

The IDE creates the RPNCalculatorUI form and the RPNCalculatorUI class within the RPNCalculator application, and opens the RPNCalculatorUI form in the GUI Builder. The my. RPNCalculator package replaces the default package.

Adding Components: Making the Front End
Next we will use the Palette to populate our application’s front end with a JPanel. Then we will add JLabels and JTextFields for five variables A, through E, a JLabel and JTextField to display the Result, and a JLabel and JTextField to hold the RPN expression. We will add JButtons to Run the calculator, to Clear the Results and to Exit the program.

If you do not see the *Palette* window in the upper right corner of the IDE, choose *Window > Palette*.

1. Start by selecting a Panel from the *Swing Containers* category on Palette and drop it onto the JFrame.

2. While the JPanel is highlighted, go to the Properties window and click the ellipsis (…) button next to Border to choose a border style.

3. In the Border dialog, select *TitledBorder* from the list, and type in *RPN Calculator* in the Title field. Click OK to save the changes and exit the dialog.

4. You should now see an empty titled JFrame that says *RPN Calculator* like in the screenshot. Look at the screenshot and add JLabels, JTextFields and JButtons as you see above.

**Renaming the Components**

In this step we are going to rename the display text of the components that were just added to the JFrame.

Double-click each jLabel1 and change the text property to a more appropriate name such as A, B, Run, etc.

Delete the sample text from each jTextField. You can make the display text editable by right-clicking the text field and choosing Edit Text from the popup menu. You may have to resize the jTextField to its original size.

Rename the display text of the JButtons (Run, Clear, and Exit) by choosing Edit Text. Or you can click the button, pause, and then click again.

**Adding Functionality** - In this exercise we are going to give functionality to the buttons and make use of the TextFields to access input values and display results.
Making the Exit Button Work

In order to give function to the buttons, we have to assign an event handler to each to respond
to events. In our case we want to know when the button is pressed, either by mouse click or via
keyboard. So we will use ActionListener responding to ActionEvent.

Right click the Exit button. From the pop-up menu choose Events > Action > actionPerformed.
Note that the menu contains many more events you can respond to! When you select the
actionPerformed event, the IDE will automatically add an ActionListener to the Exit button and
generate a handler method for handling the listener's actionPerformed method.

The IDE will open up the Source Code window and scroll to where you implement the action
you want the button to do when the button is pressed (either by mouse click or via keyboard).
Your Source Code window should contain the following lines:

```java
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    //TODO add your handling code here:
}
```

We are now going to add code for what we want the Exit Button to do. Replace the TODO line
with System.exit(0);. Your finished Exit button code should look like this:

```java
private void jButton3ActionPerformed(java.awt.event.ActionEvent evt) {
    System.exit(0);
}
```

Making the Clear Button Work

Click the Design tab at the top of your work area to go back to the Form Design.

Right click the Clear button (jButton1). From the pop-up menu select Events > Action >
actionPerformed.

We are going to have the Clear button erase all text from the jTextFields. To do this, you will
add some code like above. Your finished source code should look like this:

```java
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt){
    JTextField1.setText("\n");
    JTextField2.setText("\n");
    JTextField3.setText("\n");
    :
}
```

The above code changes the text in all of our JTextFields to nothing, in essence it is overwriting
the existing Text with a blank.

Adding Functionality to the Run Button is left an exercise for the student. Refer to NetBeans
Tutorial Introduction to Building a GUI (on which this project guide is based) for more guidance.