NetBeans IDE GUI How-To

Building Graphical User Interfaces is easy in an IDE such as NetBeans. Just follow these steps to create an event driven application with a GUI front-end. These instructions will walk you through the creation of the VowelCounterProject creation. Follow them carefully.

1. Create the Project:
   a. Choose File | New Project from the menu bar.

   b. In the Categories Pane, select the Java node. In the Projects pane choose Java Application. Click Next.

   c. Type **VowelCounterProject** in the Project Name field and accept the path to the project location.

   d. Leave *Use Dedicated Folder for Storing Libraries* deselected.

   e. **Deselect *Create Main Class*** (this is very important!!!)

   f. Click [Finish]

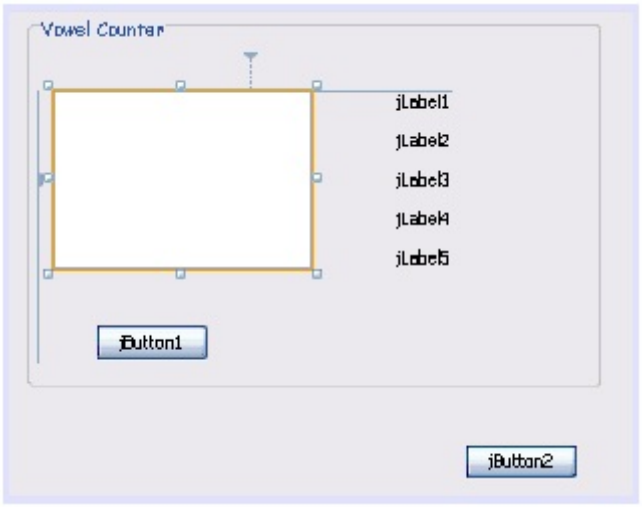
2. Build the GUI front end:
   a. **Right-click** the **VowelCounterProject** node in the Projects window on the left of the workspace and choose **New | JFrame Form**

   b. Enter **VowelCounterUI** as the **class** name.

   c. Enter my.vowelcounter as the **package** name.

   d. Click [Finish]

The IDE creates the VowelCounterUI form and the VowelCounterUI class within the VowelCounterProject application, and opens the VowelCounterUI form in the GUI Builder. The my.vowelcounter package replaces the default package.



3. **Adding components**: Making the Front End

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

a. Start by selecting a **JPanel** from the Palette and dropping it onto the **JFrame**. Stretch it to a size similar to the picture above by pulling on the handles.

b. While the JPanel is selected, go to the **properties** window and click the **ellipsis** (...) button next to **Border** to choose a border style.

c. In the **Border dialog**, select **TitledBorder** from the list, and type in **Vowel Counter** in the **Title field**. (You may opt to choose a different font here if you wish.) Click OK to save the changes and exit the dialog.

d. Looking at the screen shot above, add the remaining components from the Swing Controls palette: a Text Area, five Labels, and two Buttons.

4. Naming the components: In this step we will rename the components and change the text that is displayed on them.

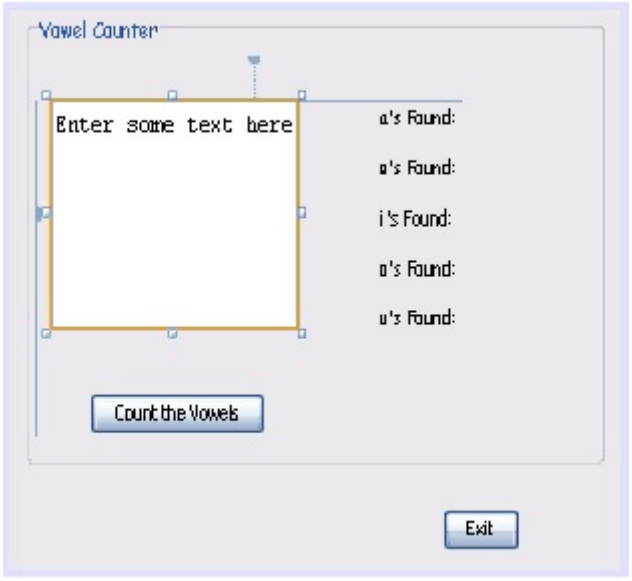
a. right-click the JTextArea. Choose **Change Variable Name** from the context menu and type ***`inputBox`*** in the text field and click [OK]

b. In the properties pane (with the text area still selected), check the boxes next to **editable** and **lineWrap**. Change the **text** property to *Enter some text here*.

c. right-click jLabel1. Choose **Edit Text** from the context menu and type *a's Found:* in the text area and Click [OK]

d. right-click the label again. This time Choose **Change Variable Name** type **`aLabel`** in the text area and click [OK]

e. Continue this procedure naming the remaining labels: eLabel, iLabel, oLabel, uLabel, and the buttons: countButton, and exitButton. Change the text properties to reflect those in the screen shot:



Adding functionality: In this step we will add event handlers to the button components and add code to the handlers

5. **Making the exit button work**:

a. right click the **exit button**. Choose Events | Mouse | Mouse Clicked.

b. 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 by a mouse click.

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

c. replace the //TODO comment with the code:

```
System.exit(0);
```

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

6. **Making the countButton work**:

a. click the [Design] option to return to the GUI layout and the palette.

b. right click the **countButton.** Choose Events | Mouse | Mouse Clicked.

c. Again, 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 by a mouse click.

d. Using the design below, code the method so that it counts the vowels in the input string - comment your code so that you understand what each line does. You might want to look at the String API for an explanation of getChars()

Local vars: aCount, eCount, iCount, oCount, uCount (all initialized to 0)
userInput (a String that is initialized to the contents of the text in the inputBox
Code: `String userInput = inputBox.getText();`
charArray (an array of char long enough to hold the userInputString)
Code: `char[] charArray = new char[userInput.length()];`
Convert the userInput String to an array of char and put it into charArray
Code: `userInput.getChars(0, userInput.length(), charArray, 0);`
For each value of i starting at 0, while i < the length of the userInput String, i++
if the current character is equivalent to one of the 5 vowels
increment the count variable for that vowel (nested if)

Set the **text** of the **aLabel** to "a's Found: " + aCount
etc…

e. Test it using several phrases