



**COLLEGE OF ARTS AND SCIENCES
SCHOOL OF COMPUTING
SKIP1103 INTERMEDIATE PROGRAMMING
LAB EXERCISE 7**

GRAPHICAL USER INTERFACE & EVENT-DRIVEN PROGRAMMING

PART 1: NETBEANS IDE GUI BUILDER

In the IDE's GUI Builder, you can build your forms by simply putting components where you want them as if you were using absolute positioning. The GUI Builder figures out which layout attributes are required and then generates the code for you automatically. You need not concern yourself the layout managers. Follow these instructions:

Create a Netbeans project

1. In Netbeans, choose **File > New Project**. In the Categories pane, select the Java node and in the Projects pane, choose **Java Application**. Click Next.
2. Enter **RectangleApp** in the **Project Name** field and specify the project location. Leave the Use Dedicated Folder for Storing Libraries checkbox unselected.
3. Ensure that the Create Main Class checkbox is selected and clear the Create Main Class field. Click Finish.

New Java Application

Steps

1. Choose Project
2. **Name and Location**

Name and Location

Project Name:

Project Location:

Project Folder:

☐ Use Dedicated Folder for Storing Libraries

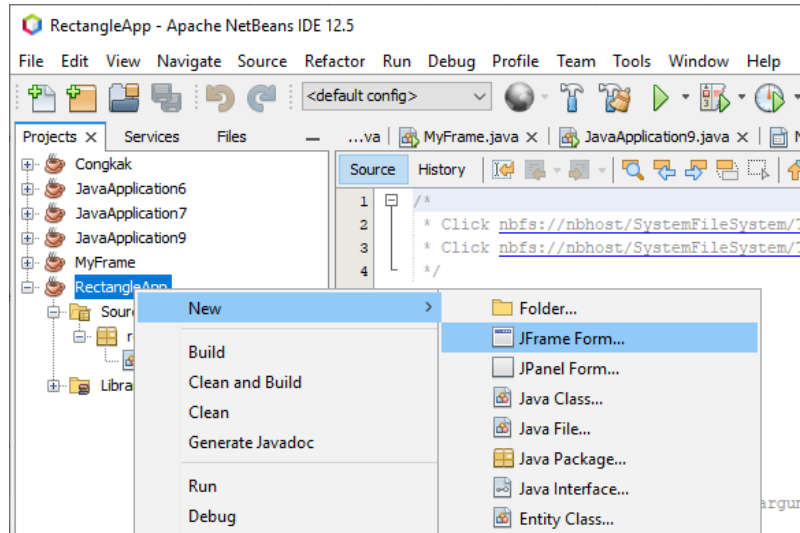
Libraries Folder:

Different users and projects can share the same compilation libraries (see Help for details).

☒ Create Main Class

Creating a JFrame container

1. To add a JFrame container, in the Projects window, right-click the **RectangleApp** node and choose **New > JFrame Form**.

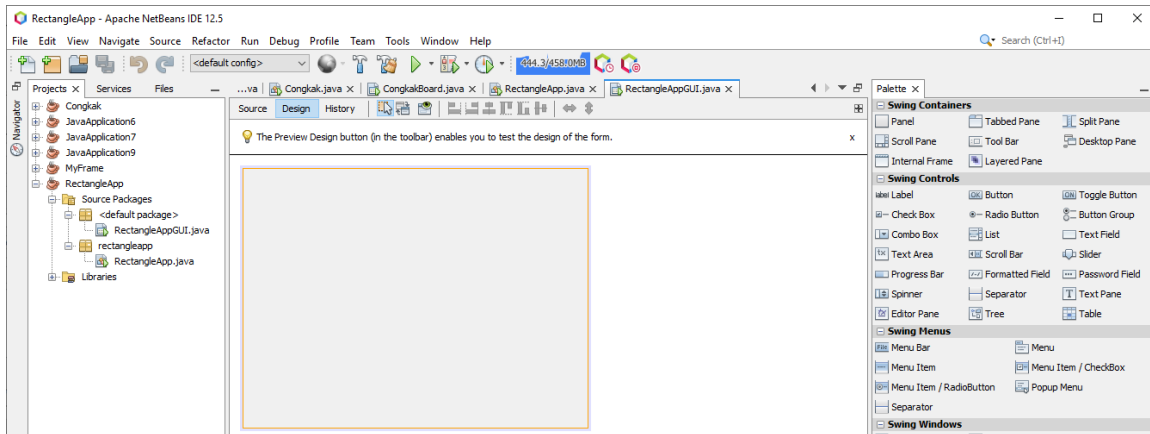


2. Enter **RectangleAppGUI** as the Class Name.
3. Click Finish.

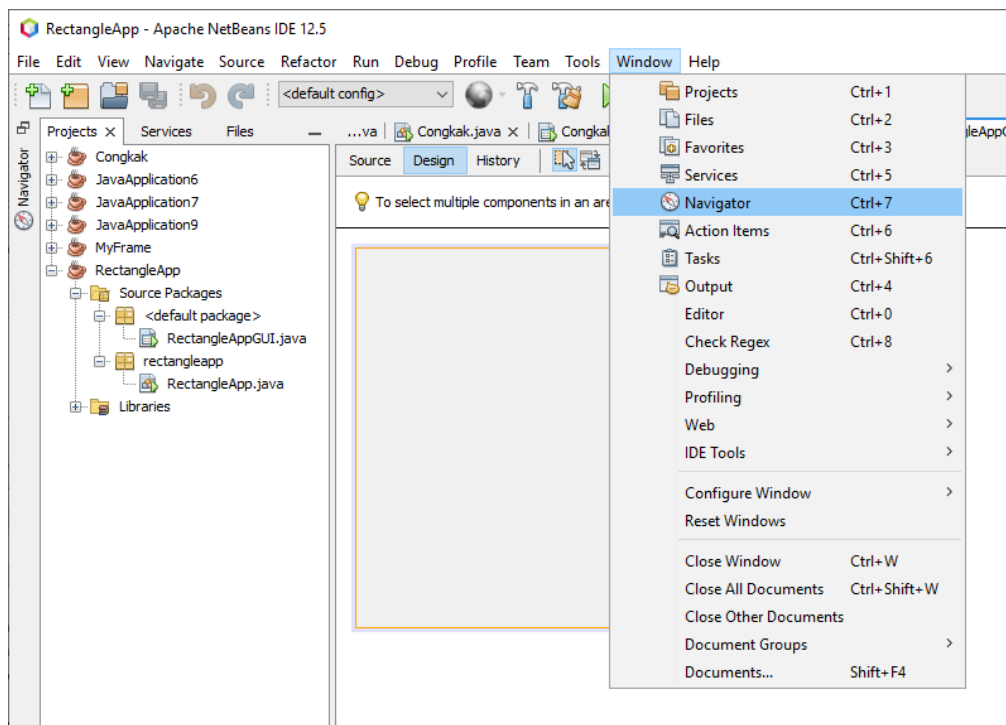
A screenshot of the 'New JFrame Form' dialog box in NetBeans. The dialog has a 'Steps' section on the left with '1. Choose File Type' and '2. Name and Location'. The 'Name and Location' section contains several fields: 'Class Name' (RectangleAppGUI), 'Project' (RectangleApp), 'Location' (Source Packages), 'Package' (empty), and 'Created File' (intermediate Programming\codes\Topic 7 - GUI\RectangleApp\src\RectangleAppGUI.java). There are also fields for 'Superclass' and 'Interfaces' with 'Browse...' buttons. At the bottom, there are buttons for '< Back', 'Next >', 'Finish' (highlighted), 'Cancel', and 'Help'. A warning message at the bottom states: 'Warning: It is highly recommended that you do not place Java classes in the default package.'

Getting familiar with the GUI builder

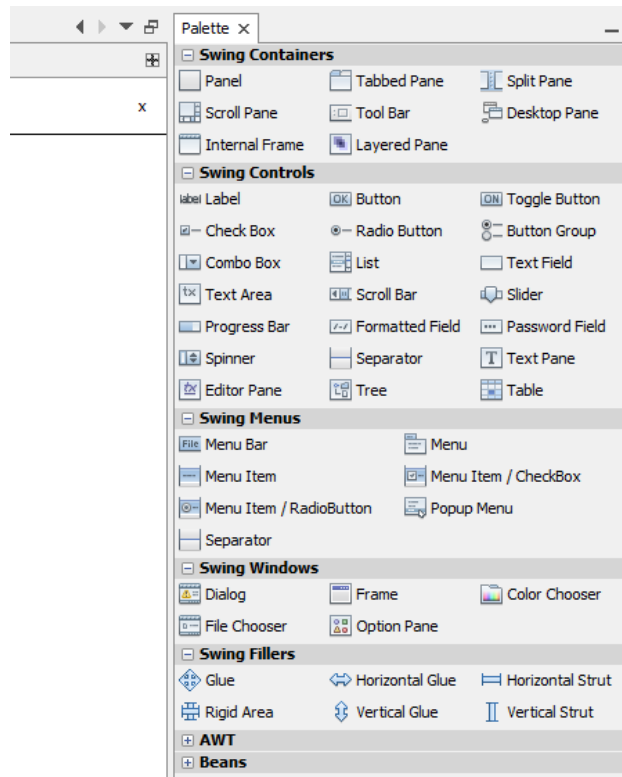
1. When we added the JFrame container, the newly-created RectangleAppGUI form is opened in the GUI Builder's Design Area and three additional windows appeared automatically, enabling you to navigate, organize, and edit GUI forms.
2. The windows are as follows:
 - i. **Design Area.** The GUI Builder's primary window (middle) for creating and editing Java GUI forms.



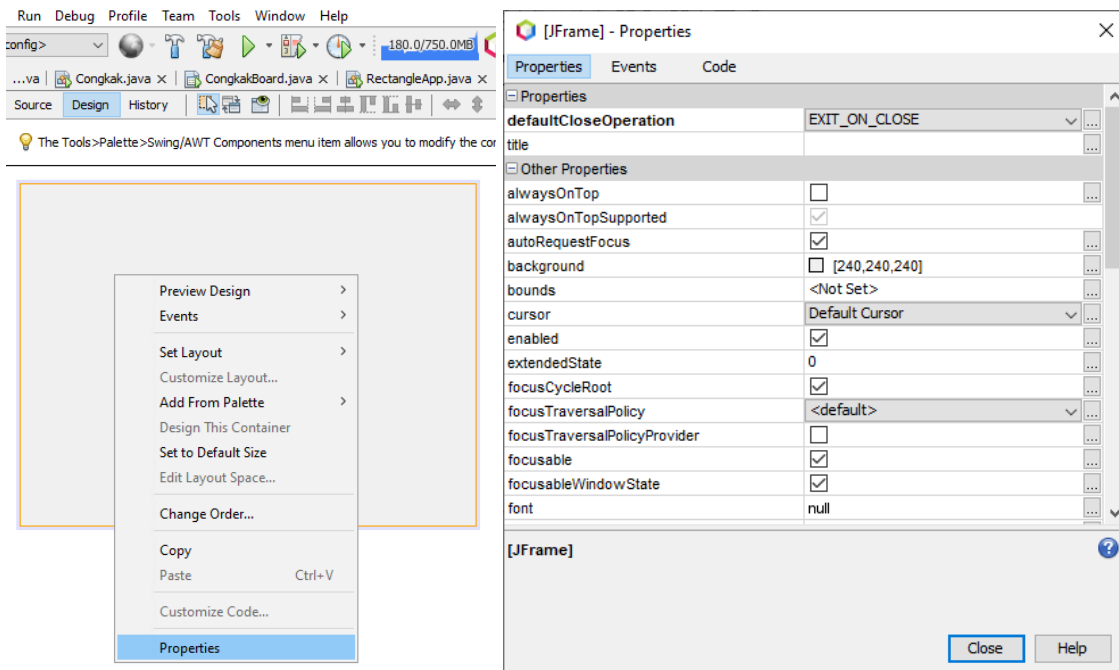
- ii. **Navigator.** Provides a representation of all the components, both visual and non-visual, in your application as a tree hierarchy.



- iii. **Palette.** A customizable list of available components containing tabs for JFC/Swing, AWT, and JavaBeans components, as well as layout managers.

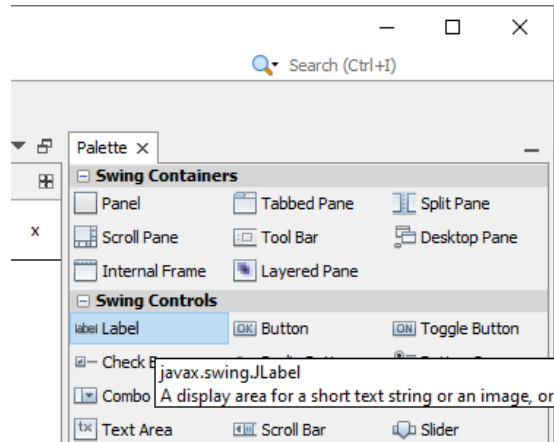


- iv. **Properties Window.** Displays the properties of the component currently selected in the GUI Builder, Navigator window, Projects window, or Files window.

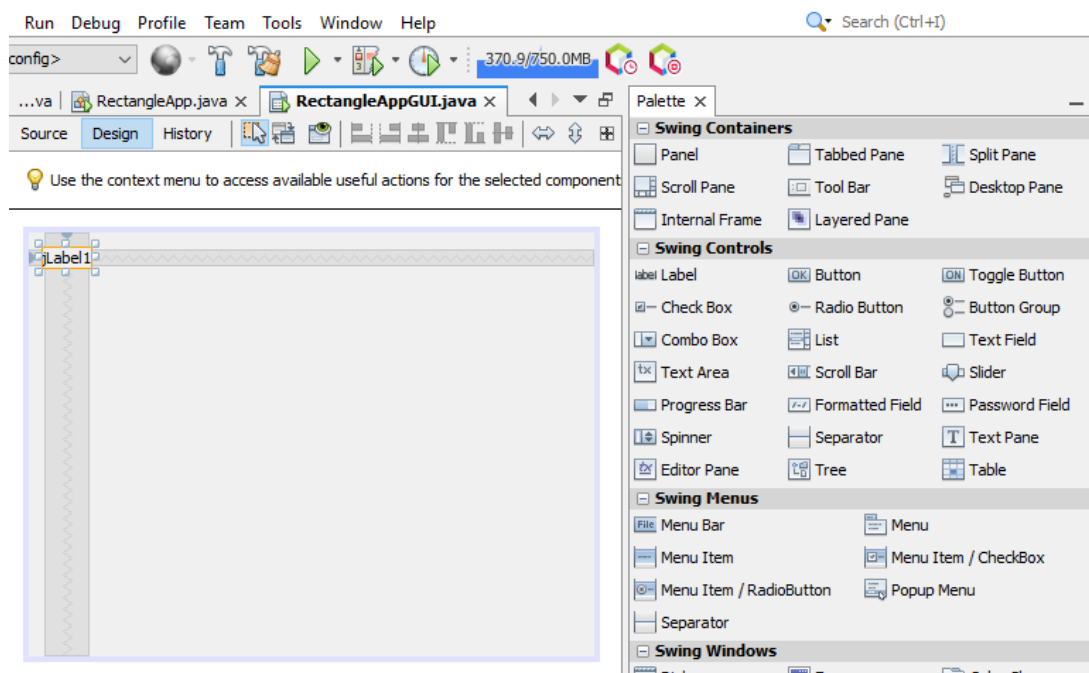


Adding components

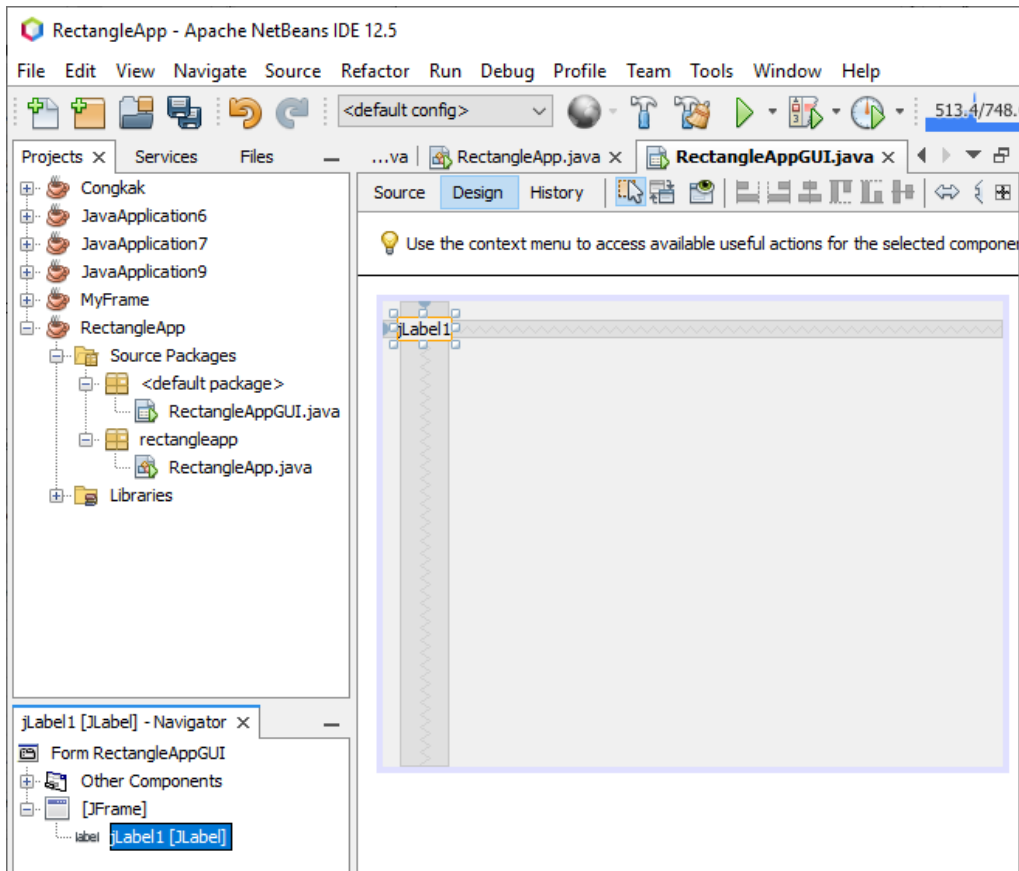
1. To add a **JLabel** to the form, in the Palette window, select (click your mouse) the Label component from the Swing Controls category.



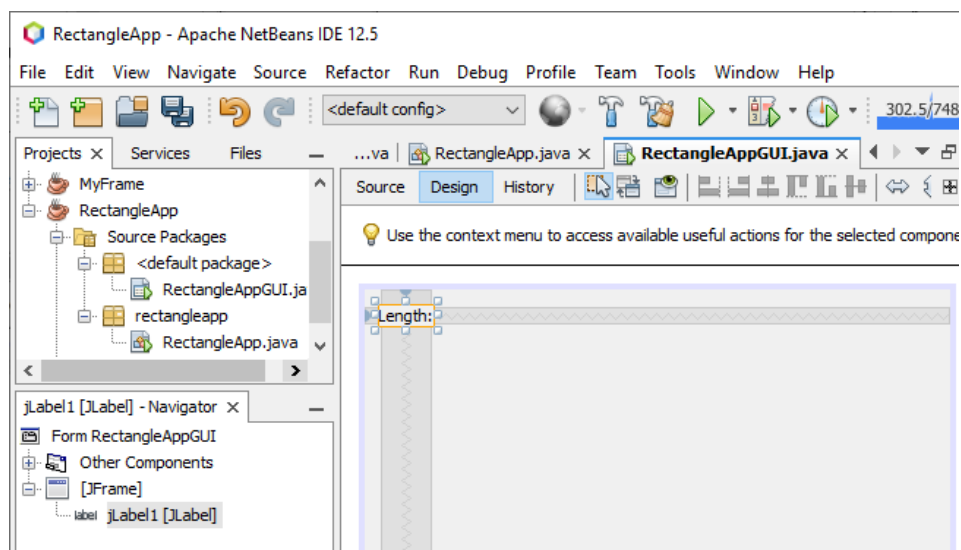
2. Move your mouse into the **Design Area**.
3. When the guidelines appear indicating that the JLabel is positioned in the top left corner with a small margin at the top and left edges, click again your mouse to place the Label.



4. The **JLabel** is added to the form and a corresponding node representing the component is added to the **Navigator window**.

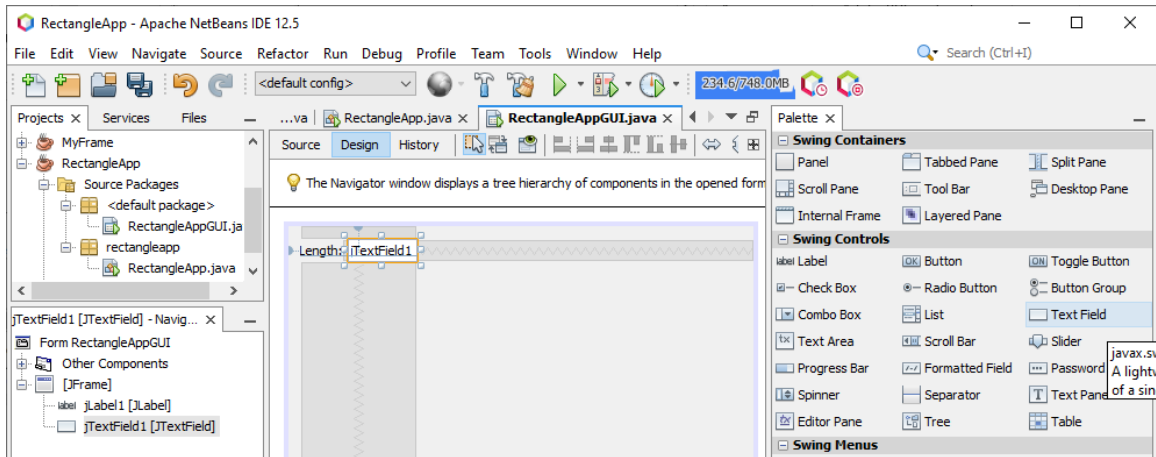


5. Before going further, we need to edit the JLabel text. To edit the display text of a JLabel:
 - i. **Double-click** the JLabel to select its display text.
 - ii. Type 'Length:' and press Enter.
 - iii. The JLabel's new name is displayed, and the component's width adjusts as a result of the edit.



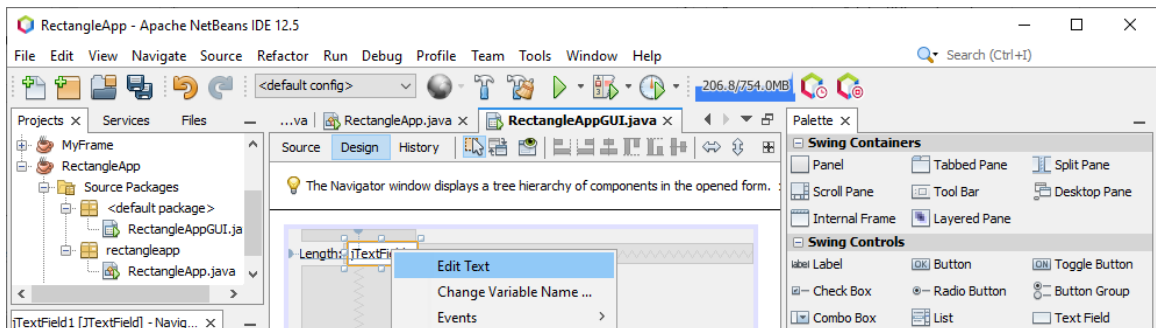
6. To add a **JTextField** to the form:

- i. In the **Palette** window, select the Text Field component.
- ii. Move the cursor immediately to the right of the 'Length:' JLabel
- iii. When the horizontal guideline appears indicating that the JTextField's baseline is aligned with the JLabel and the spacing between them is suggested with a vertical guideline, click to position the JTextField.
- iv. The JTextField snaps into position in the form aligned with the JLabel's baseline, as shown below:

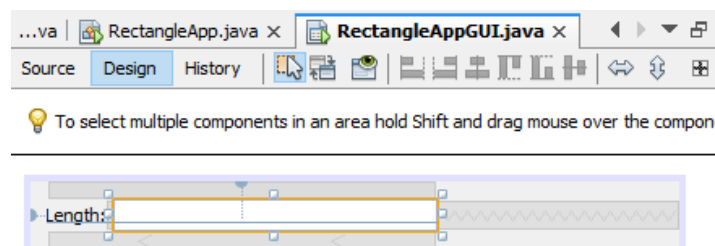


7. To remove the text inside the JTextField & resize it:

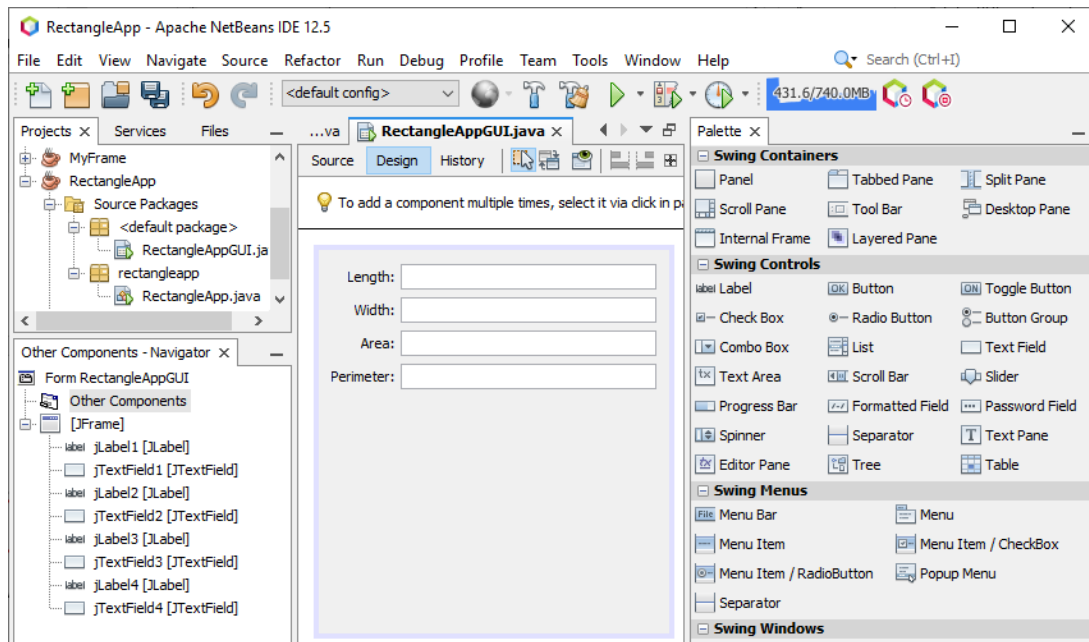
- i. Double-click the JTextField to select its display text & press Delete.



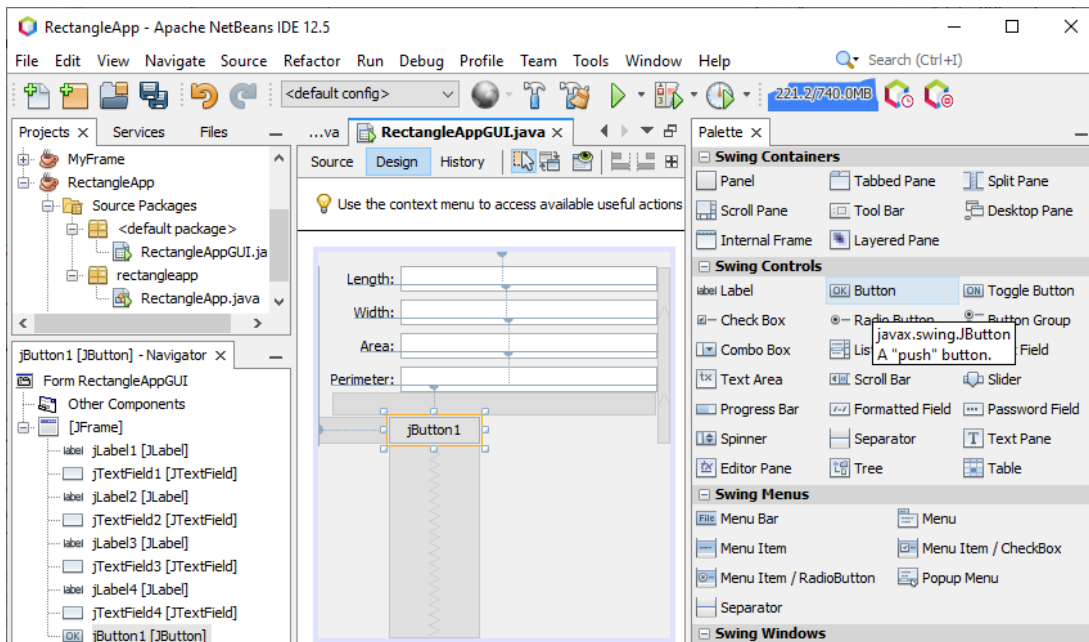
- ii. The width of the JTextField will be reduced. You need to resize (make it wider) the JTextField by dragging the JTextField's right edge resize handle toward the right edge of the enclosing JFrame.



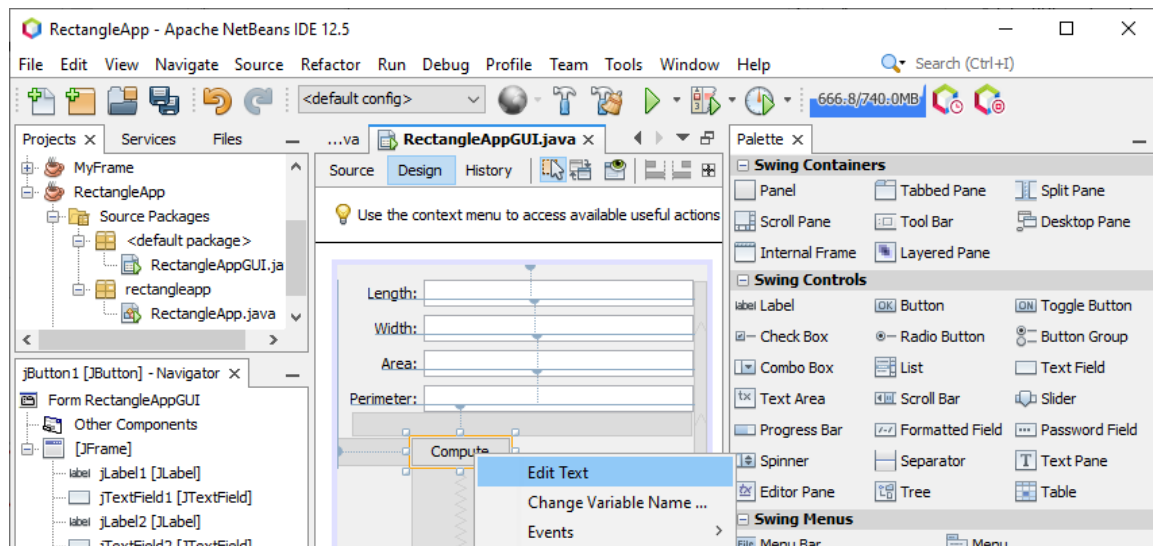
8. To add the remaining three JLabels & JTextFields, repeat all the previous steps to add JLabels & JTextFields.
9. Ensure that all the texts for the labels & textfields have been changed correctly (e.g. second JLabel's text is 'Width :' etc.)
10. Once finished, you should have your GUI as shown below:



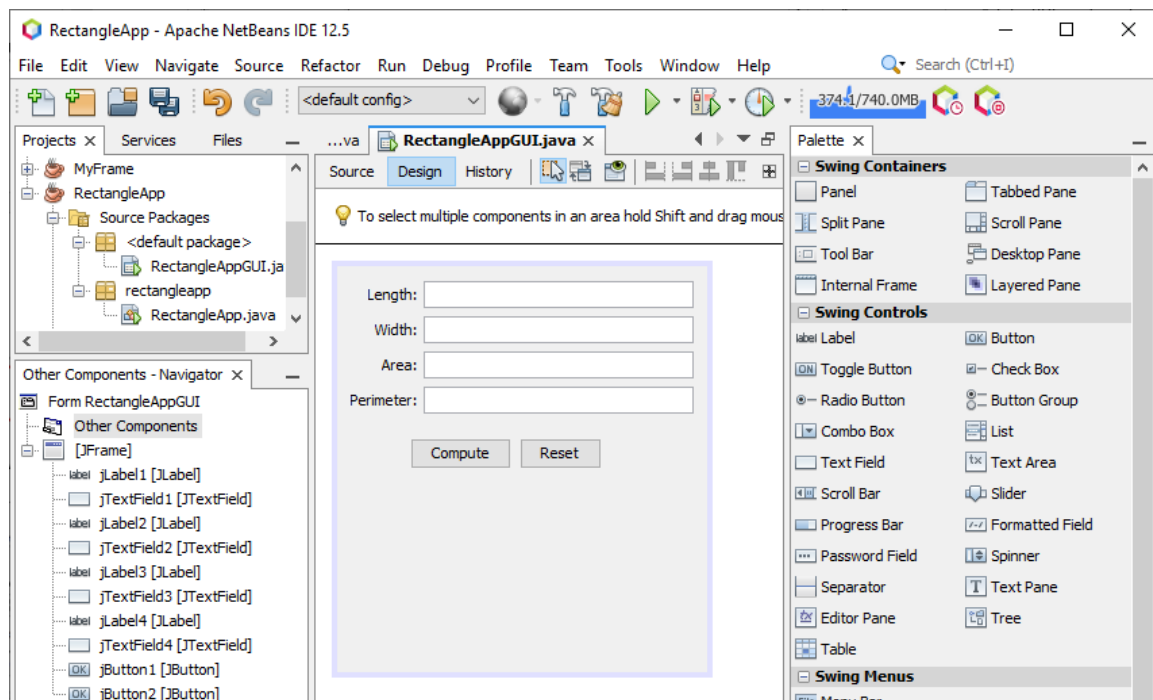
11. To add JButton to the form, in the Palette window, select the Button component from the Swing Controls category. Move the JButton below of the Perimeter: JTextField.



12. Set the display text for the JButton. You can edit the button's text by right-clicking the button and choosing Edit Text. Enter 'Compute' for the button

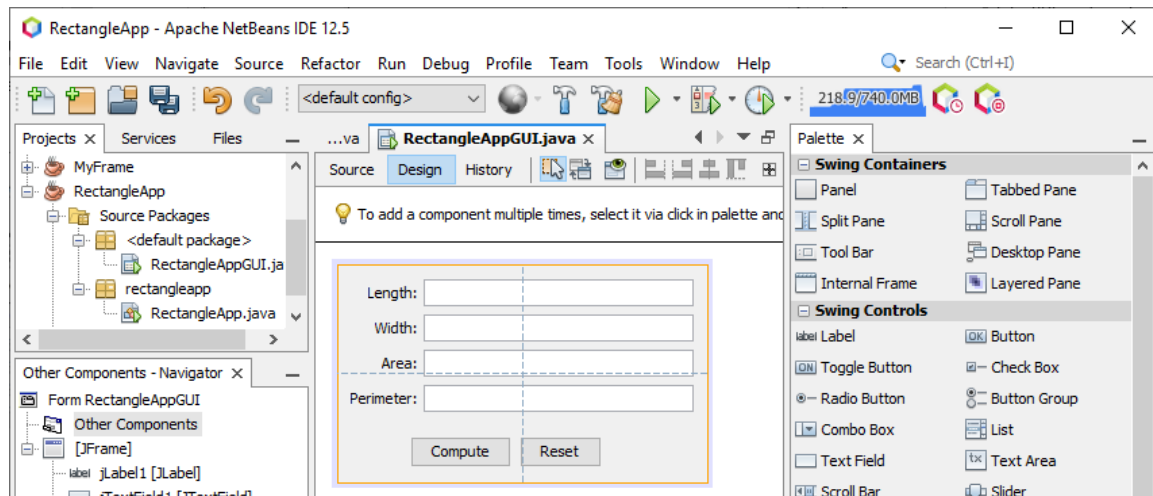


13. To add another JButton repeat all the previous steps to add the second JButton. The second JButton text must be set to 'Reset'. Once finished, you should have your GUI as shown below:

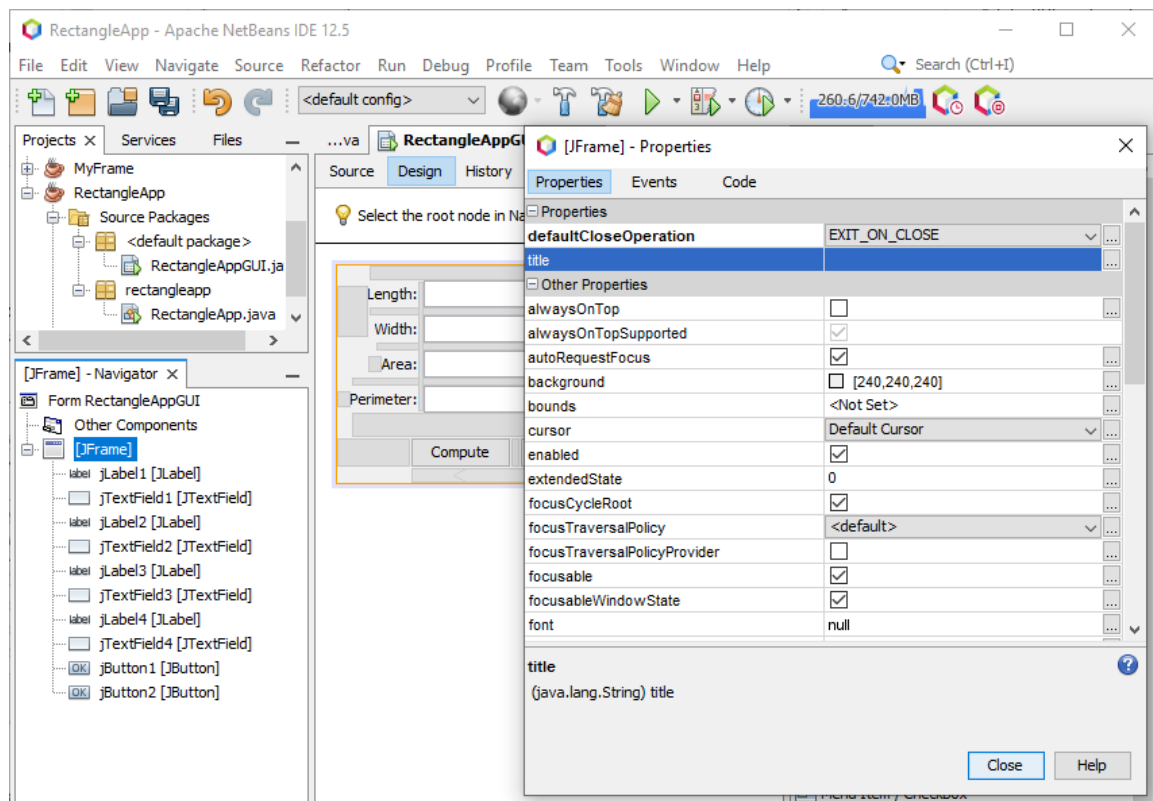


Set the JFrame container size and title

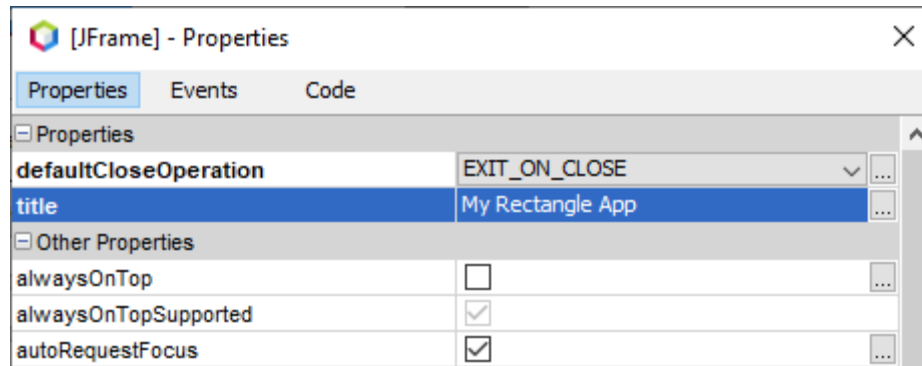
1. To resize your JFrame (main container), bring your mouse to the bottom right corner of the frame. Click & drag your mouse diagonally up or down to resize the frame according your preference.



2. To set the JFrame title, click the **JFrame** > **Properties** and ensure that it is highlighted (with orange-colored border). In the JFrame Properties Window, click the 'title' property field.

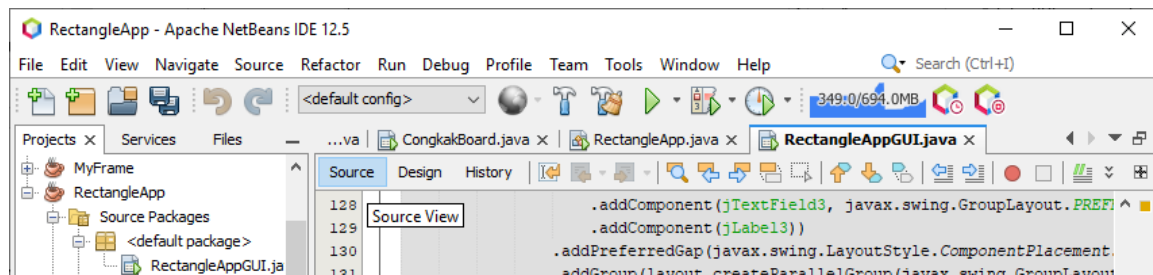


3. To set the JFrame title type the desired title (e.g. 'My Rectangle App') in the title property field.

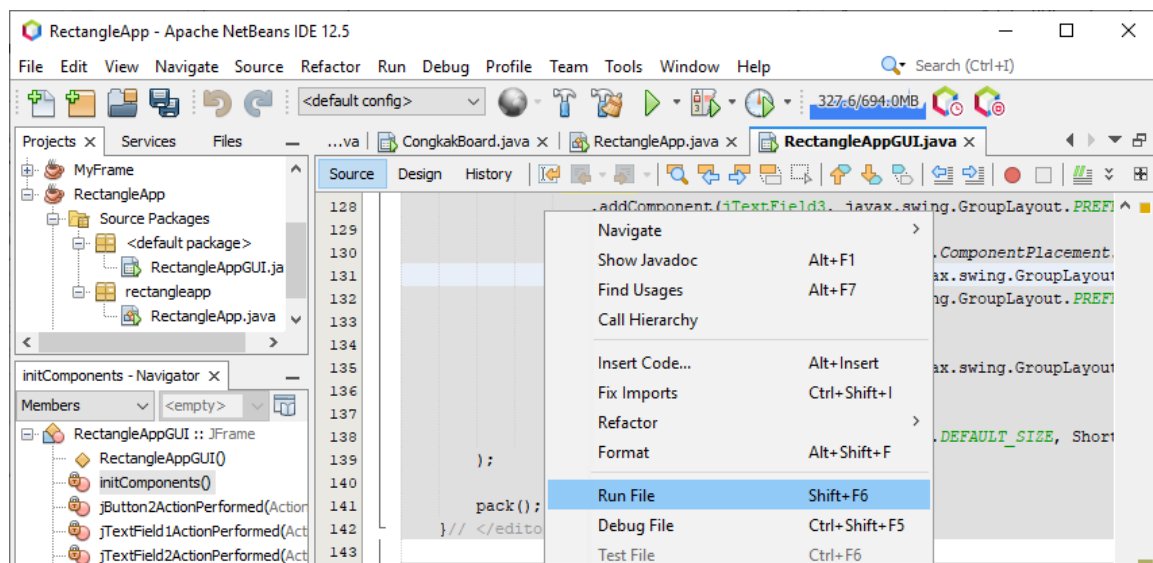


Running the app

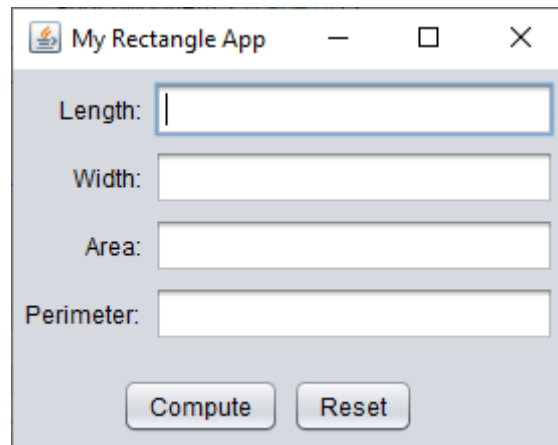
1. To run your RectangleAppGUI program, click the **Source** view tab to view the source codes that have been generated by Netbeans.



2. To run your RectangleAppGUI program, Right Click your mouse in the generated codes area & select "Run File".



3. You can see your GUI when the program runs:

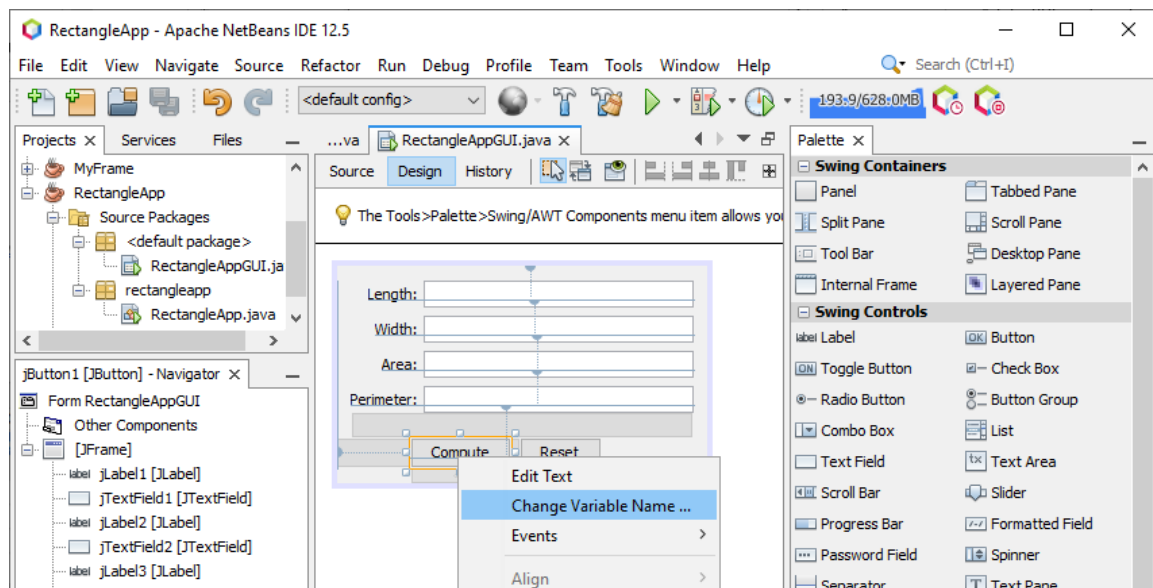


PART 2: GUI EVENT HANDLING

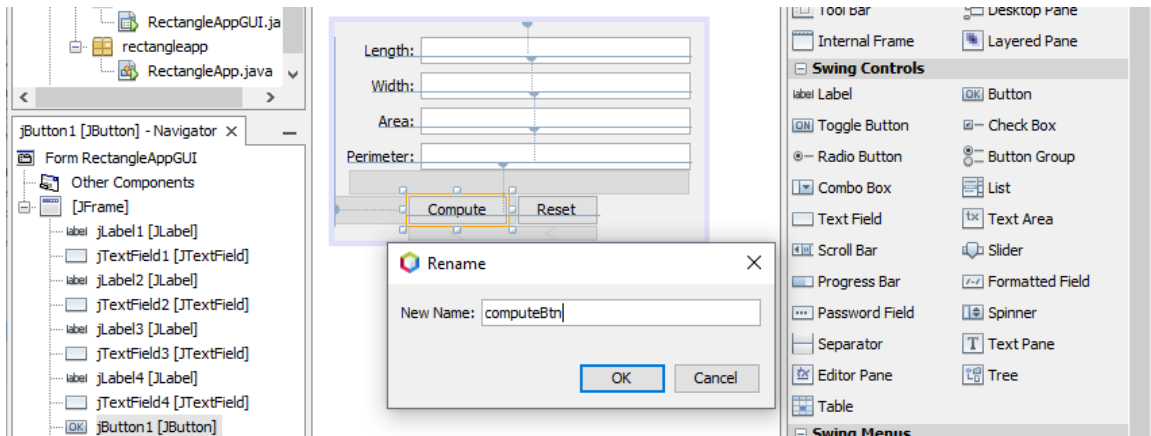
Here we will add the event handling for the GUI that we have built previously. Specifically, we'll add the functionality for the two buttons, Compute & Reset

Renaming the GUI components variables

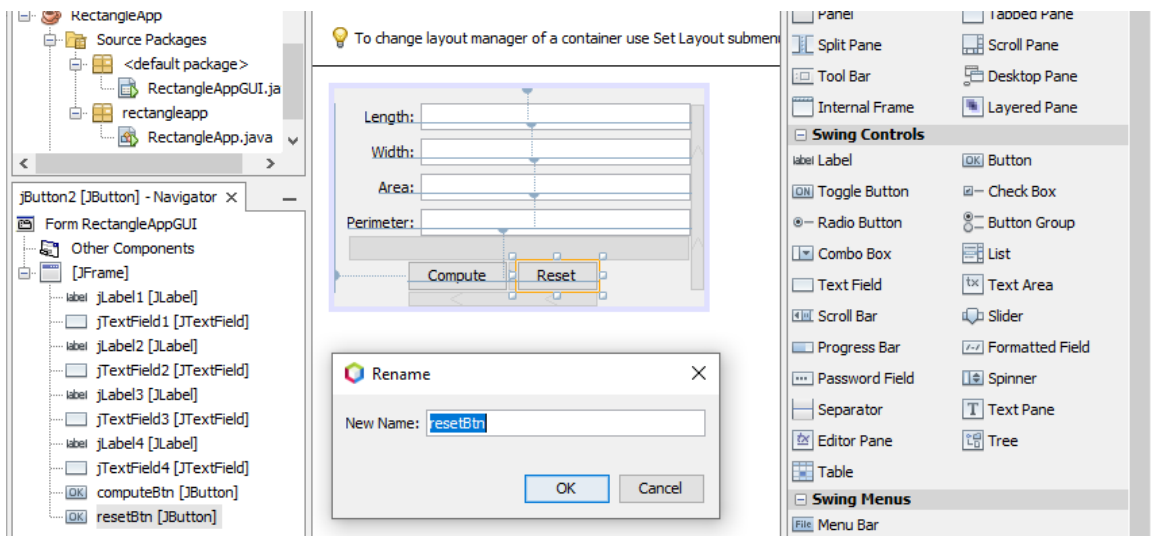
1. The reference variables for GUI objects (such as buttons & textfields) should be renamed so that we can easily identify them in the codes generated by Netbeans GUI Builder.
2. Right-click the Compute button and select **Change Variable Name**.



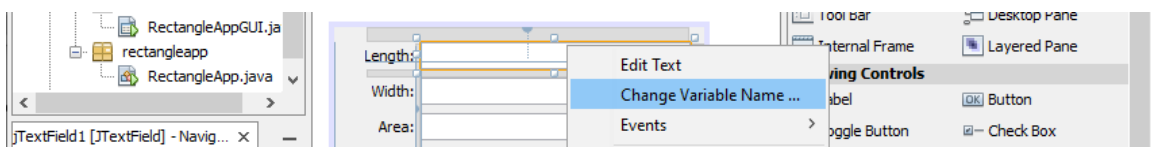
3. In the Rename dialog box, type '**computeBtn**'. (The variable name for the Compute button has been changed from **jButton1** to **computeBtn**).



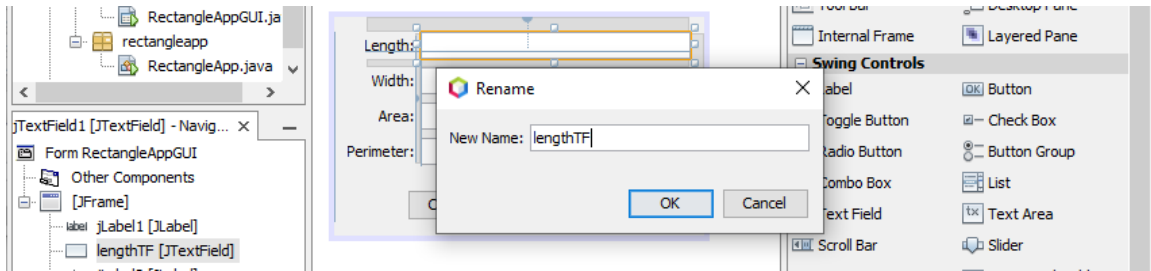
4. Repeat the previous steps to rename the variable for Reset button from **jButton2** to **resetBtn**.



5. Next we are going to rename all the variables for the four **textfields**.
6. Right-click the textfield beside label '**Length:**' and select **Change Variable Name**.



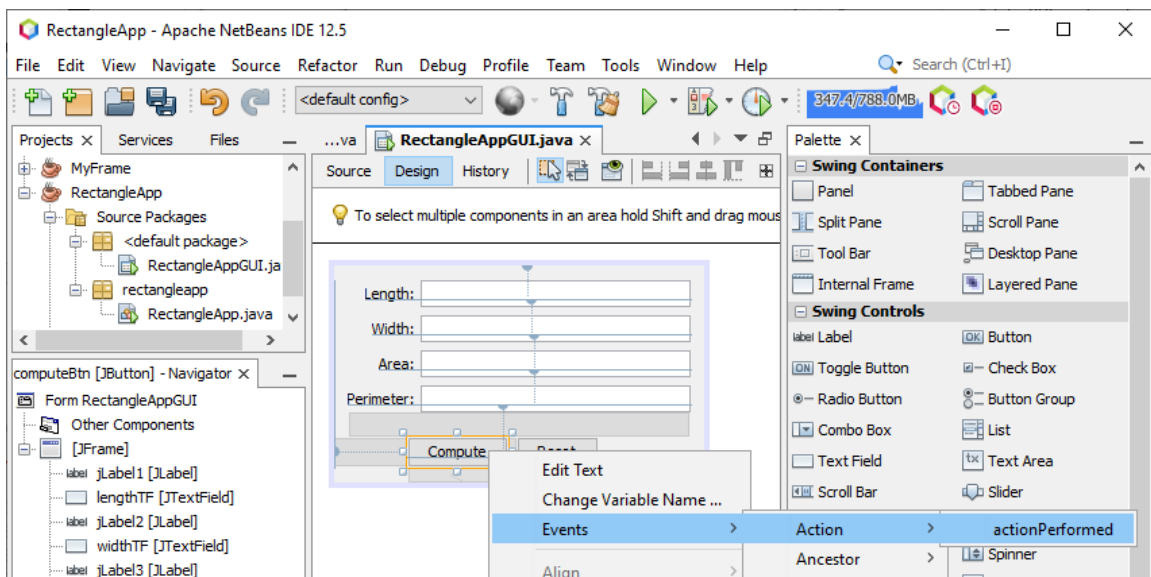
7. In the Rename dialog box, type '**lengthTF**'. (The variable name for the Length textfield has been changed from **jTextField1** to **lengthTF**).



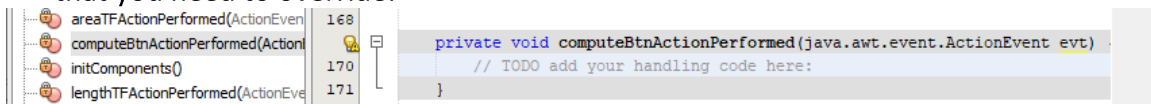
8. Repeat the previous steps to rename the remaining three textfields:
 - i. the variable for **Width** textfield from **jTextField2** to **widthTF**
 - ii. the variable for **Area** textfield from **jTextField3** to **areaTF**
 - iii. the variable for **Perimeter** textfield from **jTextField4** to **perimeterTF**

Adding the event handlers

1. First, we will make the Compute button works by assigning the handler to the button, specifically, we will use ActionListener responding to ActionEvent
2. Right Click on the Compute button. From the pop-up menu choose **Events > Action > actionPerformed**.



3. Netbeans will automatically add an ActionListener to the Compute button and generate a handler method for handling the listener's actionPerformed method.
4. You will be brought into the **Source** Code window where you implement the action you want the Compute button to do when the button is pressed. Your Source Code window should contain the following lines which is the actionPerformed method that you need to override:



5. Next, we add the codes to implement what we want the **Compute** Button to do (replacing the `// TODO add your handling code here:` line)
6. The action performed by the **Compute** button can be divided into 3 steps:
 - i. It is going to accept user inputs, length & width, from the textfields **lengthTF** and **widthTF** and convert the inputs from type `String` to `double`.
 - ii. It will then perform calculation of rectangle area & perimeter based on the inputs length & width,
 - iii. It will convert the area & perimeter to type `String` and place them in the **areaTF** & **perimeterTF** respectively.
7. The finished source code shall look like this:

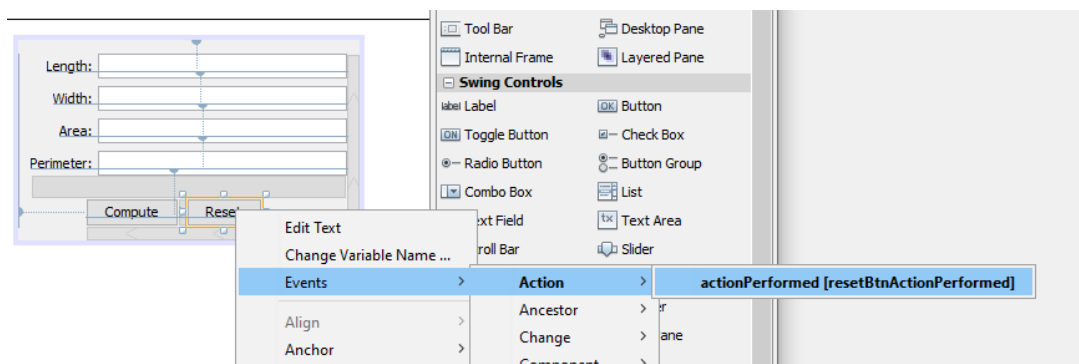
```
private void computeBtnActionPerformed(java.awt.event.ActionEvent evt)
{
    //i. Read the text input from the textfields
    String lengthText = lengthTF.getText();
    String widthText = widthTF.getText();

    //ii. Convert the text input to type double
    double length = Double.parseDouble(lengthText);
    double width = Double.parseDouble(widthText);

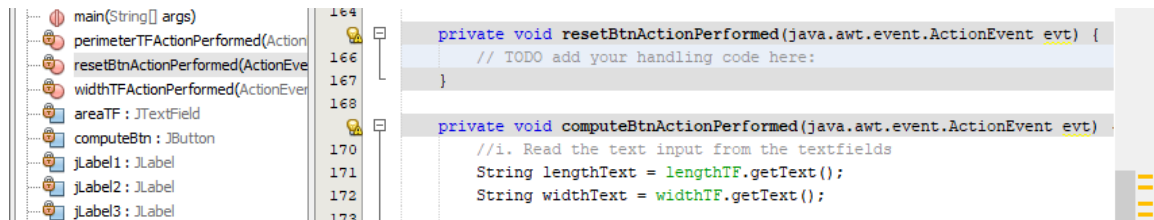
    //Perform the area & perimeter calculations
    double area = length * width;
    double perimeter = 2 * (length + width);

    //iii. Convert the results from double to String
    //and send them to be displayed in the textfields
    areaTF.setText(String.valueOf(area));
    perimeterTF.setText(String.valueOf(perimeter));
}
```

8. Next, we will make the **Reset** button works by assigning the handler to the button. Again, we will use `ActionListener` responding to `ActionEvent`. Go back to **Design**.
9. Right Click on the **Reset** button. From the pop-up menu choose **Events > Action > actionPerformed**.



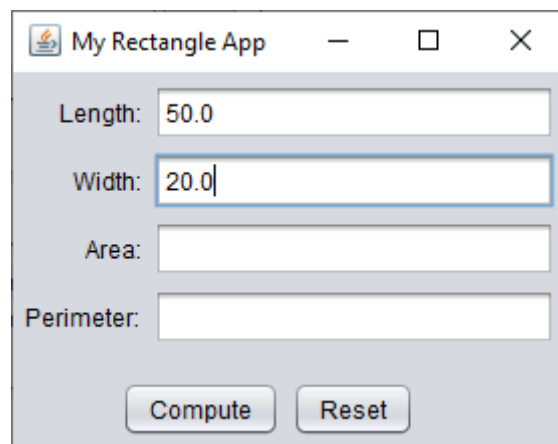
10. Your **Source** Code window should contain the following lines which is the actionPerformed method for the Reset button that you need to override:



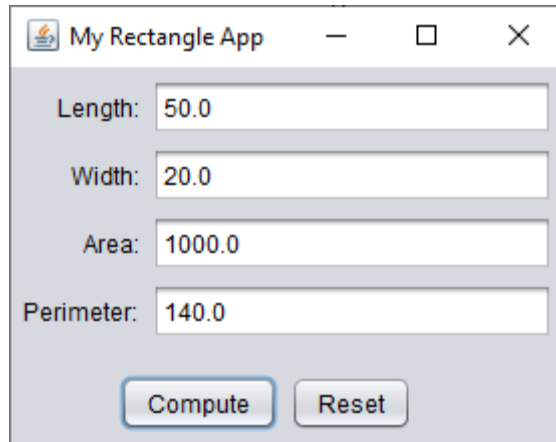
11. Next, we add the codes to implement what we want the **Reset** Button to do (replacing the // TODO add your handling code here: line)
12. The action performed by the Reset button is to simply erase all texts in the textfields. To do this, you will set the texts to all of the four textfields to empty String.
13. Your finished source code should look like this:

```
private void resetBtnActionPerformed(java.awt.event.ActionEvent evt) {  
    lengthTF.setText("");  
    widthTF.setText("");  
    areaTF.setText("");  
    perimeterTF.setText("");  
}
```

14. The final step is to run the program. Right click in the **Source** Code window and select **Run File**.
15. Test the **Compute** button by entering values in the Length & Width textfields and click the **Compute** button.

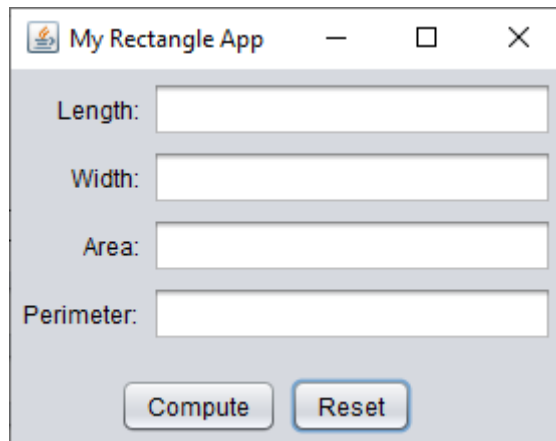


16. A sample of the output when you click **Compute** is as follows:



A screenshot of a Java Swing window titled "My Rectangle App". The window has a light gray background and standard window controls (minimize, maximize, close) in the title bar. It contains four text input fields stacked vertically, each with a label to its left: "Length:" with value "50.0", "Width:" with value "20.0", "Area:" with value "1000.0", and "Perimeter:" with value "140.0". At the bottom of the window are two buttons: "Compute" and "Reset". The "Compute" button is highlighted with a blue border.

17. Test the **Reset** button by clicking it and all the textfields will be cleared. A sample of the output is as follows:



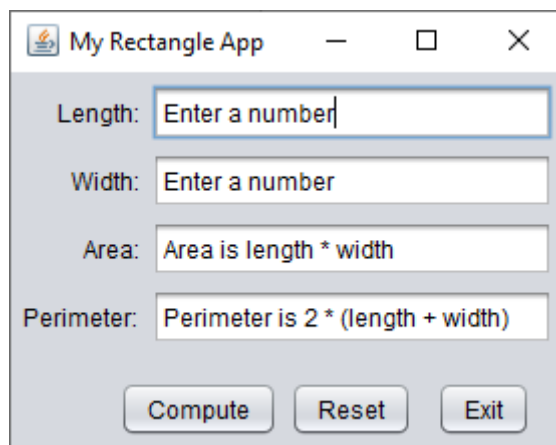
A screenshot of the same "My Rectangle App" window, but after the "Reset" button has been clicked. All four text input fields (Length, Width, Area, and Perimeter) are now empty. The "Reset" button at the bottom is now highlighted with a blue border, while the "Compute" button is no longer highlighted.

Exercise 1

1. Add another button object to the GUI which is an **Exit** button.
 2. Add the event handler for the **Exit** button so that when it is clicked, the application will close and terminate (you can use the **System.exit(0)** command to terminate a program).
 3. Copy the code that you have written for the `exitBtnActionPerformed` here.
-

Exercise 2

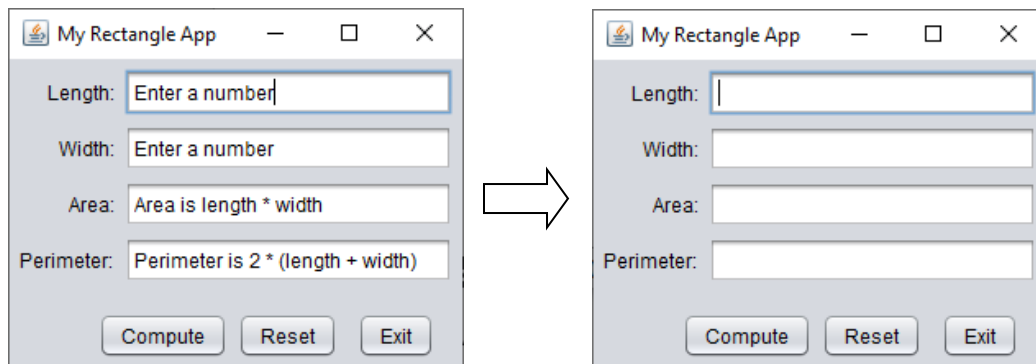
1. Set the instructions in all of the textfields so that when it runs, it shows:



2. How to set it?
 3. What happen when you click **Compute** without changing the texts into numbers?
 4. What should we do so that when the user clicks on the textfield, the text "**Enter a number**" will automatically disappear?
 5. Right click on `lengthTF` and select **Event > Mouse > mouseClicked**. What can `mouseClicked` do?
-

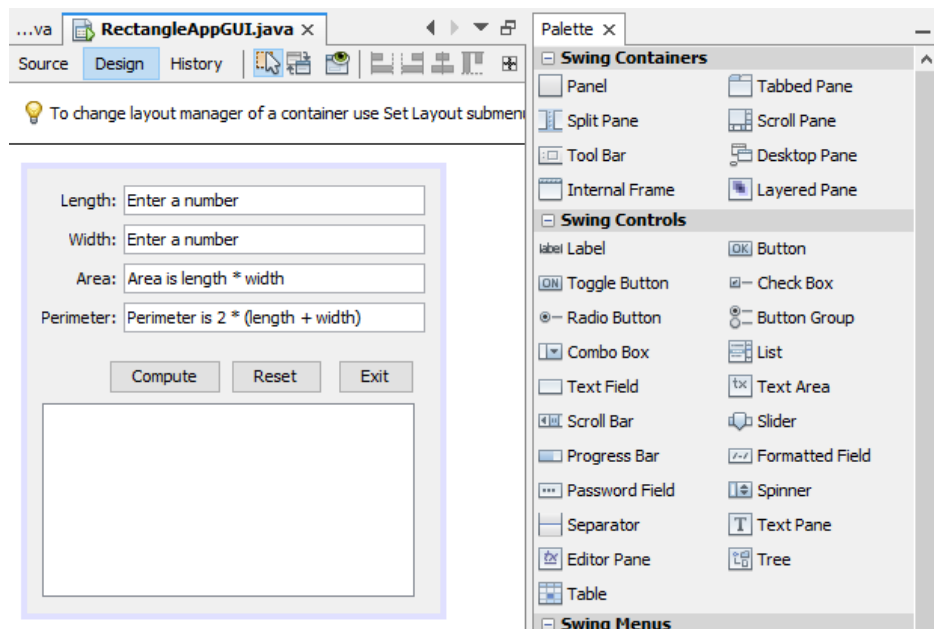
6. Modify the code so that when **lengthTF** textfield is clicked, all texts in textfields will be erased. Show your code.

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

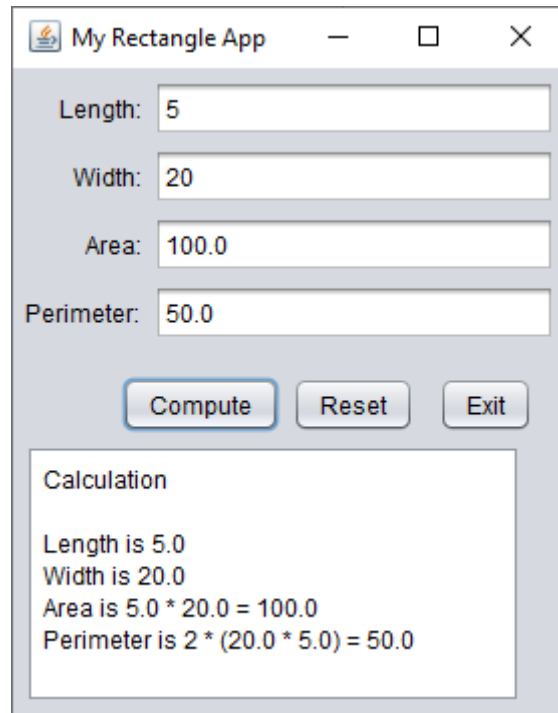


7. Try other **Events** such as **Events > Key** on any of the textfields. What does it do? (Hints: you can put `System.out.println("Testing");` to see what happens when your cursor is in the textfield, typing, leaving the textfield on the output window).

8. Modify the **Design** by adding a textArea and **Change Variable Name** to displayA.



9. Modify the `computeBtnActionPerformed` so that when the **Compute** button is clicked, it will display as shown in the textArea:



10. Copy the code that you wrote for the textArea, modified within the `computeBtnActionPerformed` to show as in the example.
-

11. Then, modify the `resetBtnActionPerformed` so that when the **Reset** button is clicked, the textArea will also be erased. What is the code that needs to be written?
-

12. Explore other components to get a better understanding on what they do. This will be useful to get you thinking on what you should be using for your **Congkak project**.

13. Save your Lab 7 as a PDF file and upload it on OL.