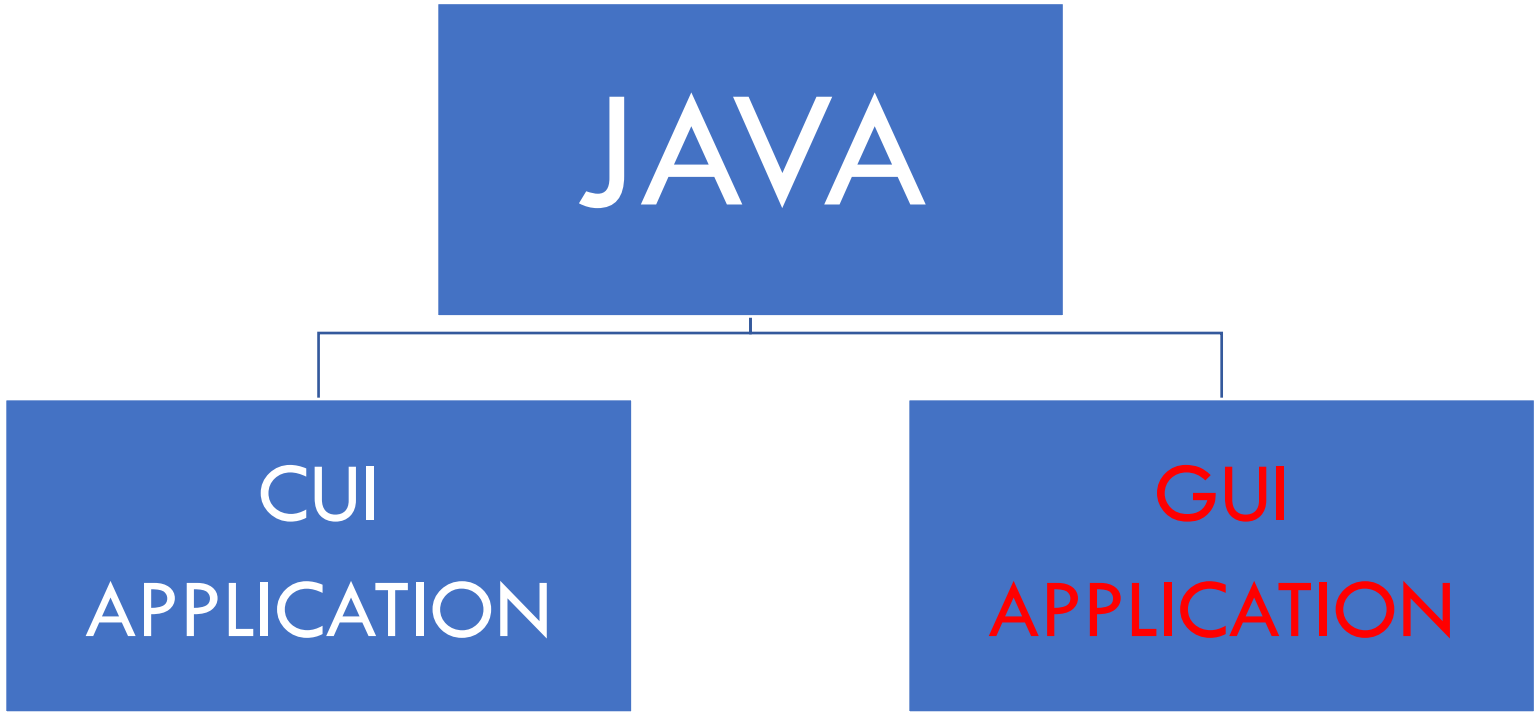
OBJECT ORIENTED PROGRAMMING USING JAVA



User Interface : User Interface comprises of everything the user can use to interact with the computer. It is basically the means by which the user and computer system can interact using input and output devices.

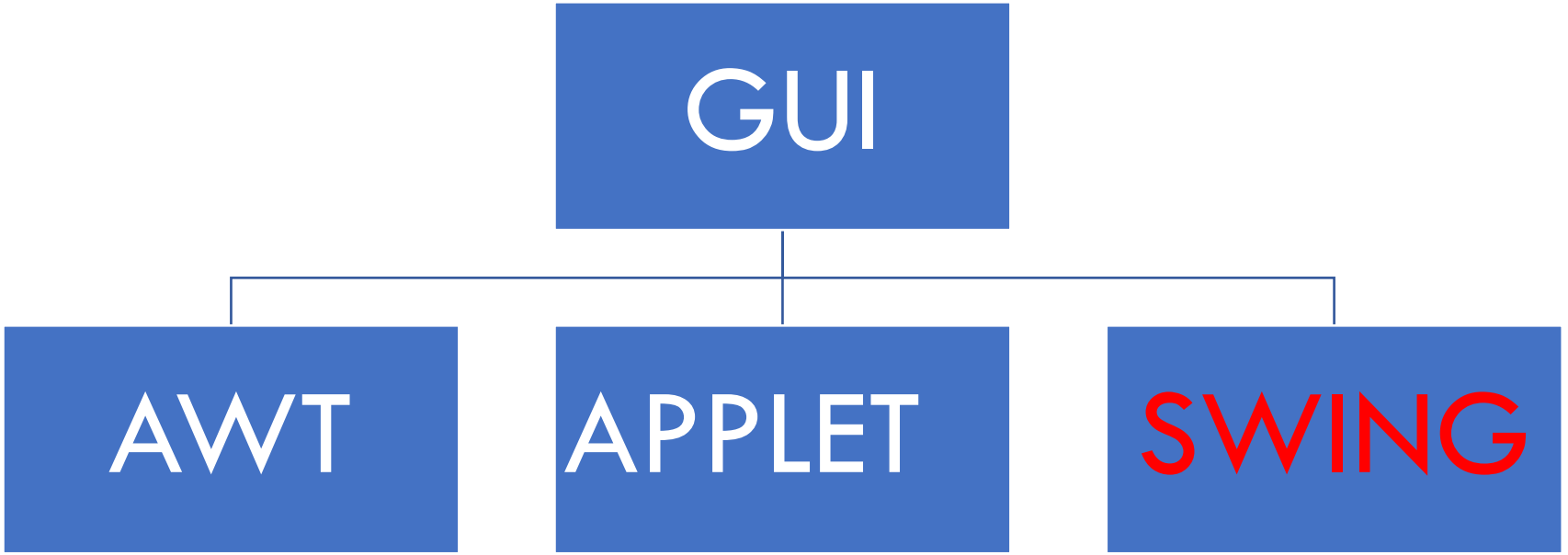
Character user interface application is the java application, where command interface is providing user interface.

Graphical user interface application where user interacts with the computer using graphics. Graphics include icons, navigation bars, images etc. Mouse can be used while using this interface to interact with the graphics. It is a very user-friendly interface and requires no expertise. Eg: Windows has GUI.

GUI VS CUI

PROPERTY	GUI	CUI
Interaction	Using graphics(images, icons)	Using commands(only text)
Navigation	Easier	Difficult
Peripherals used	Keyboard and mouse(or any pointing device)	Keyboard only
Precision	LOW	HIGH
Speed	LOW	HIGH
Ease of Operation	Easier	Difficult, requires expertise
Memory Required	HIGH	LOW
Flexibility	MORE Flexible	LESS Flexible
Customising Appearance	Highly customisable	Appearance cannot be changed

GUI in java



Java has provided three types of packages in order to perform java gui
Java.awt; // abstract windowing tool kit
Java.swing   // more functionality than awt
Java.applet // Applets are run by html technology

Swing

- Java Swing is a part of Java Foundation Classes (JFC) that is used to create window-based applications. It is built on the top of AWT (Abstract Windowing Toolkit) API and entirely written in java.

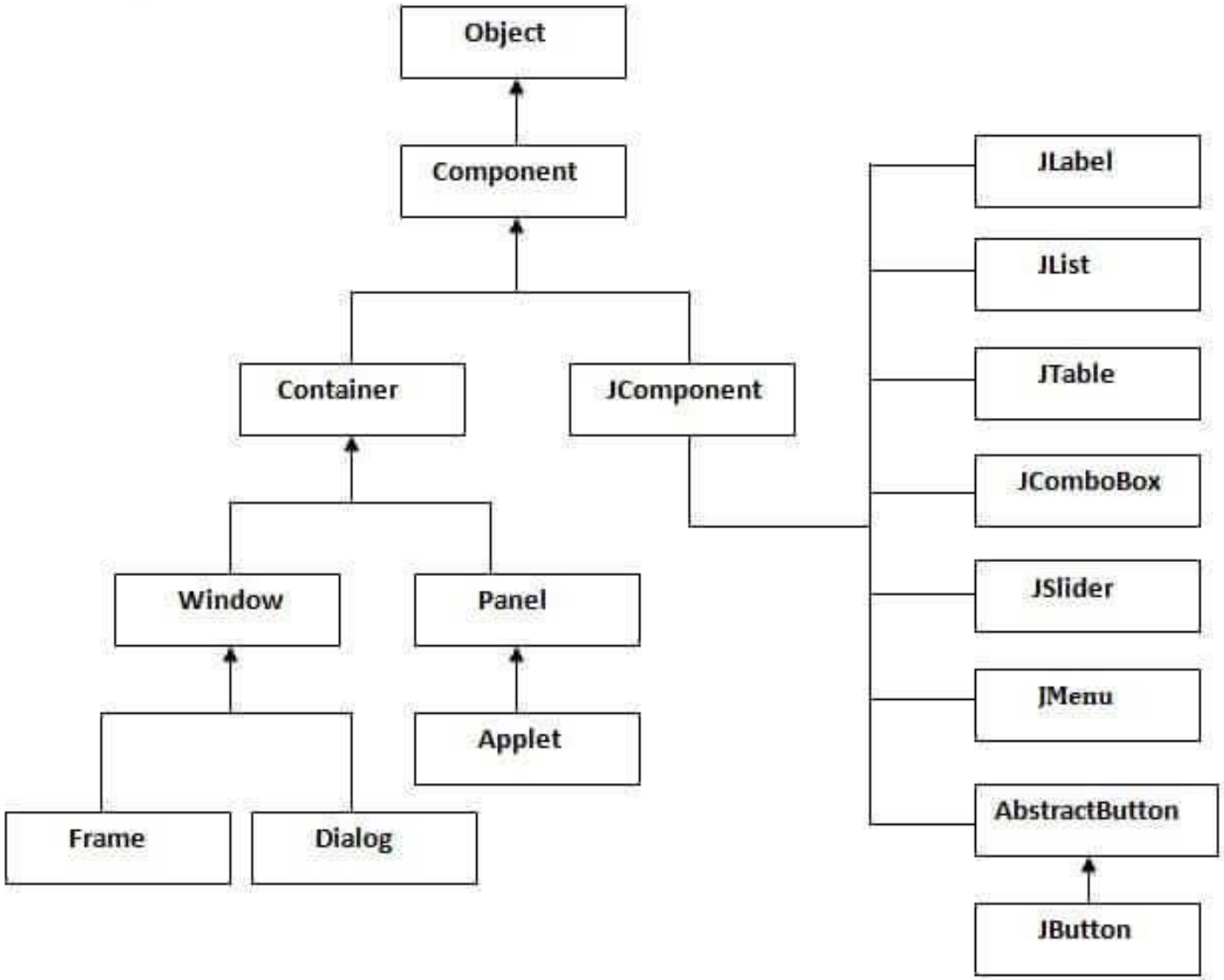
- Unlike AWT, Java Swing provides platform-independent and lightweight components.

- The javax.swing package provides classes for java swing API such as JButton, JTextField, JTextArea, JRadioButton, JCheckbox, JMenu, JColorChooser etc.

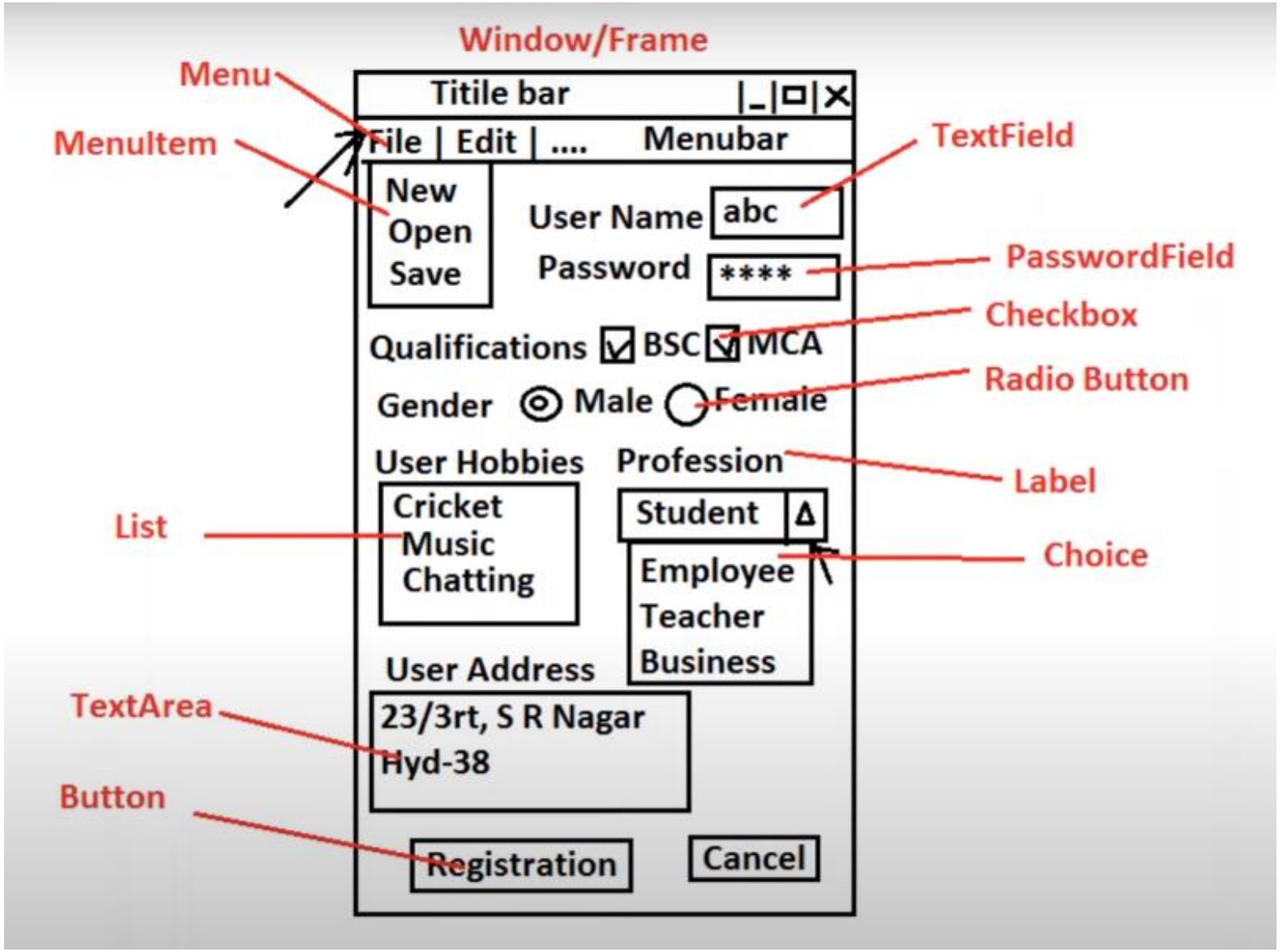
Swing Features

- Light Weight – Swing components are independent of native Operating System's API as Swing API controls are rendered mostly using pure JAVA code instead of underlying operating system calls.
- Rich Controls – Swing provides a rich set of advanced controls like Tree, TabbedPane, slider, colorpicker, and table controls.
- Highly Customizable – Swing controls can be customized in a very easy way as visual apperance is independent of internal representation.
- Pluggable look-and-feel – SWING based GUI Application look and feel can be changed at run-time, based on available values.

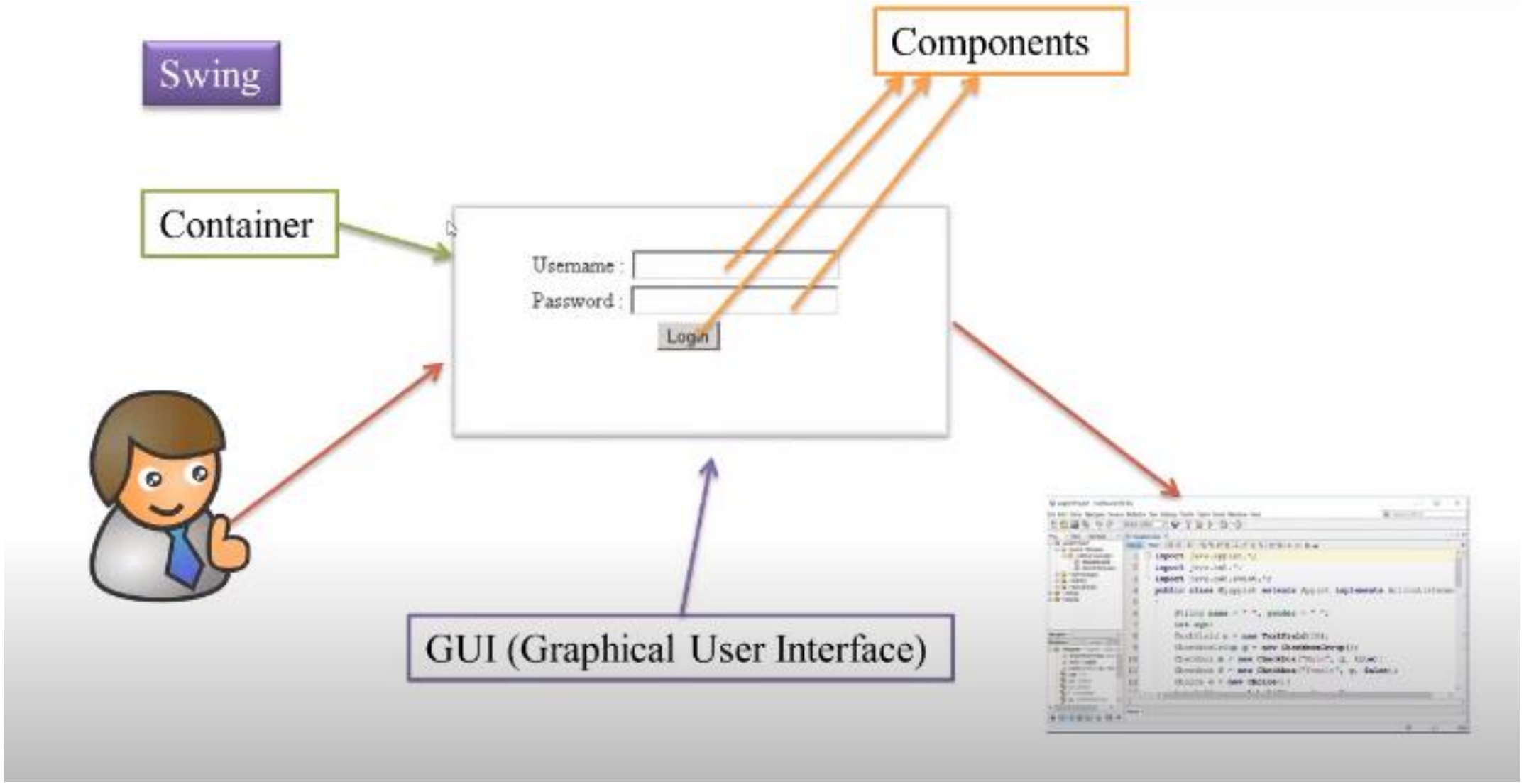
Hierarchy of Java Swing classes



GUI with terminology



Swing Diagram



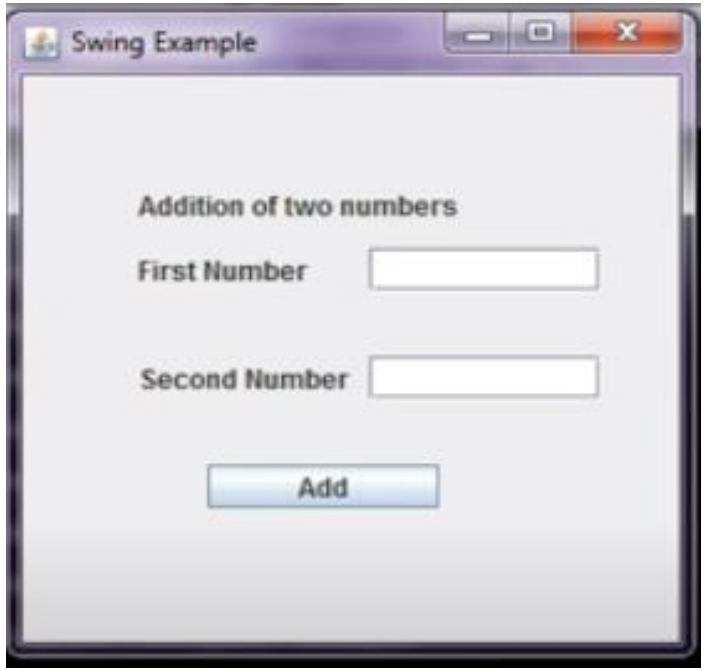
Swing UI elements

1. **JLabel**
2. **ImageIcon**
3. **JTextField**
4. **JButton**
5. **JTabded pan**
6. **JScrolpan**
7. **Action listener**
8. **Jframe**
9. JcheckBox
10. Jradiobutton
11. Jlist
12. Jcombobox
13. Jtextfield
14. Jpasswordfield
15. Joptionplane etc.

Swing Example 1

```
import javax.swing.*;
public class Main
{
public static void main (String args[])
{
// first step is to create a container
JFrame jf=new JFrame("Swing Example");
jf.setSize(300,300);
jf.setVisible(true);
jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
}
}
```

Swing Example 2



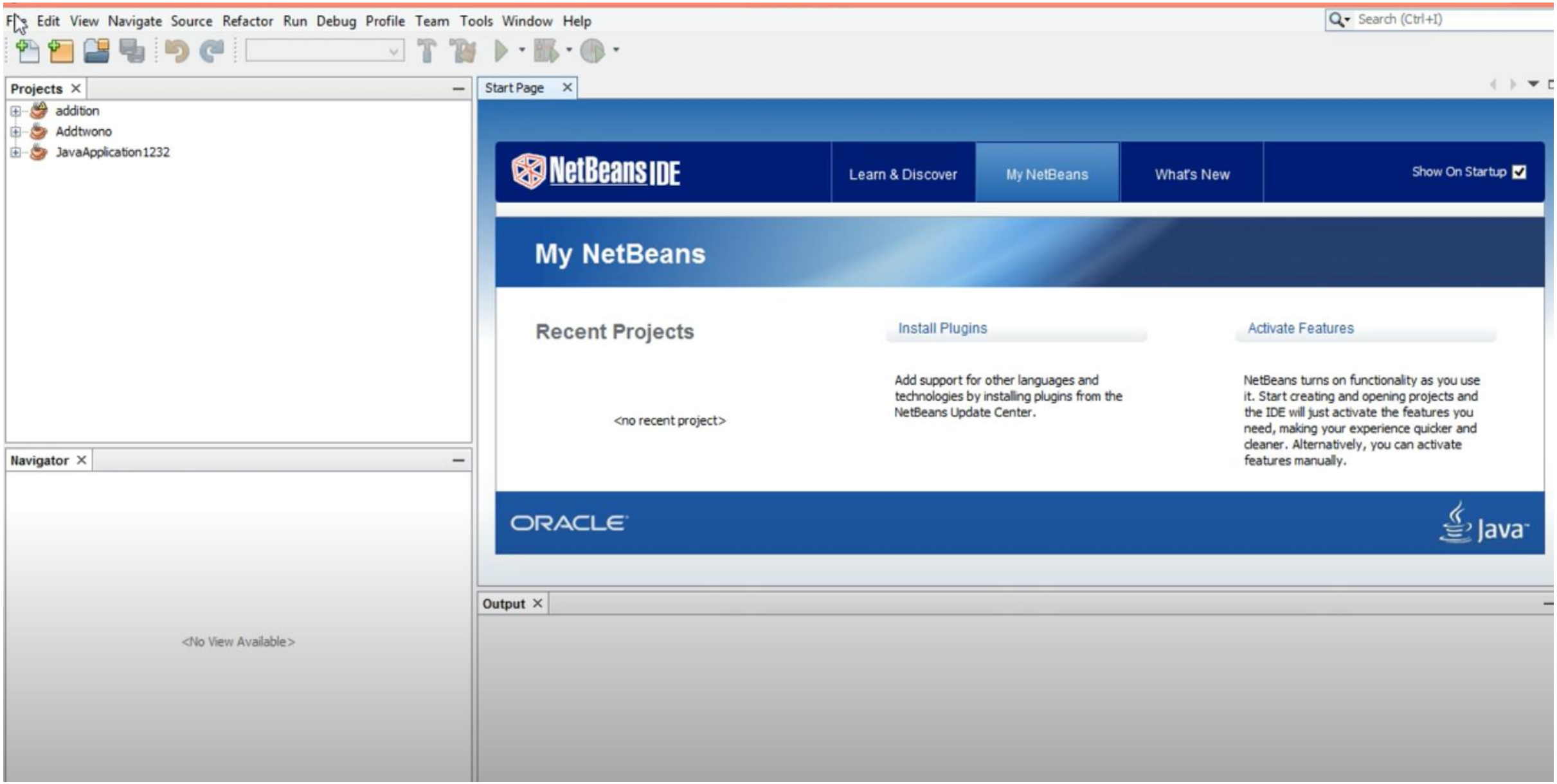

```
import javax.swing.*;
public class Main extends Jframe
{
JLabel l1,l2,l3,l4;
JTextField  t1,t2;
JButton b1;
public Main(string s)
{
super(s);
}
public static void setcomponents()
{
l1= new JLabel("addition of two numbers");
l2= new JLabel("first numbers");
l3= new JLabel("second numbers");
l4= new Jlabel();
t1=new JTestField();
t2=new JTestField();
b1=new JButton("ADD");
setLayout(null);
l1.setbounds(50,50,200,20);
l2.setbounds(50,80,100,20);
t1.setbounds(150,80,100,20);
l3.setbounds(50,130,100,20);
t2.setbounds(150,130,100,20);
b1.setBounds(80,180,100,20);
l4.setBounds(50,240,200,20);
add(l1);
add(l2);
add(l3);
add(l4);
add(t1);
add(t2);
add(b1);
}
public static void main (String args[])
{
// first step is to create a container
Main jf=new Main("Swing Example");
jf.setComponents();
jf.setSize(300,300);
jf.setVisible(true);
Jf.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE); }}
```

Swing Example 3

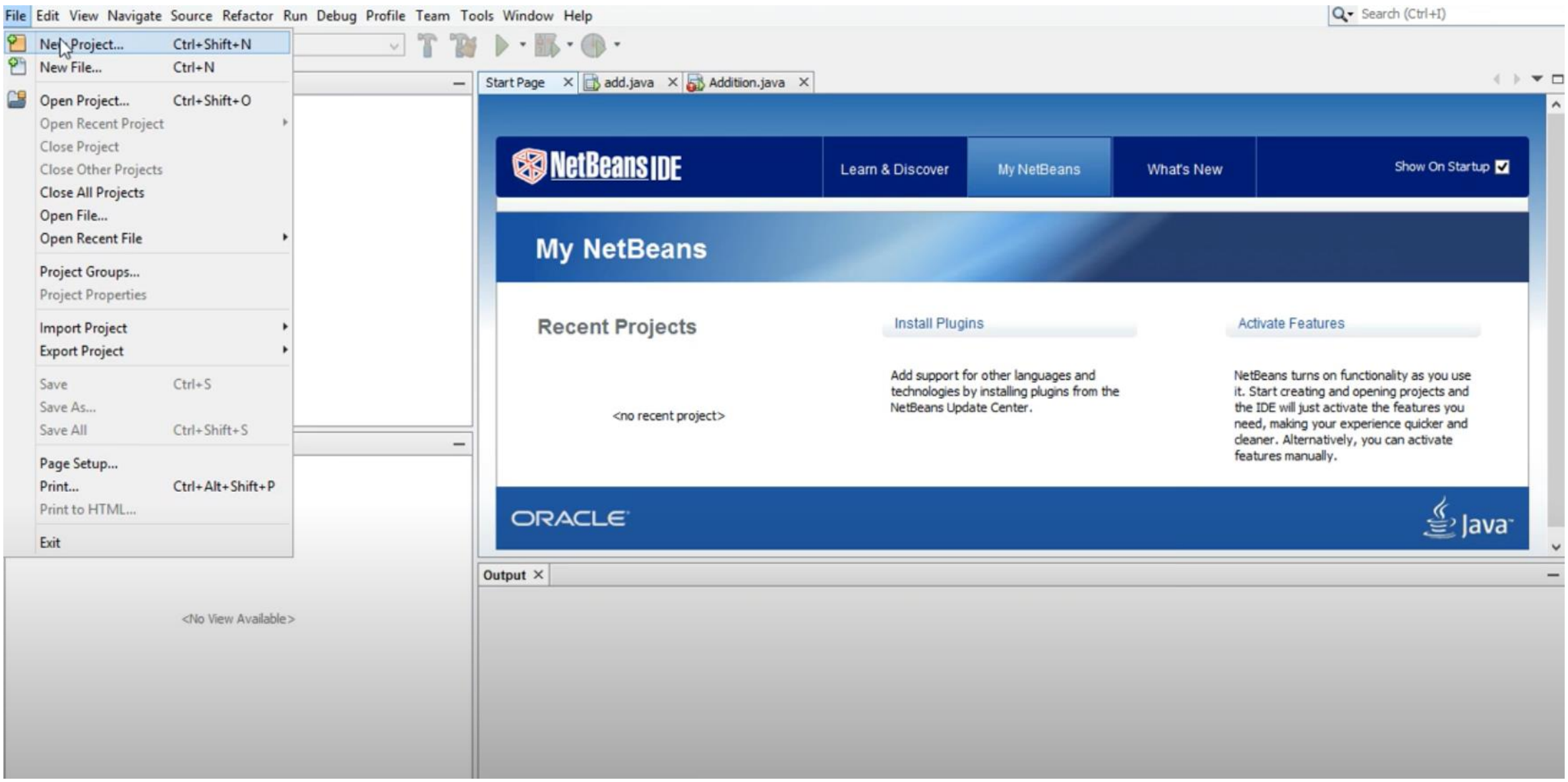
- Let's see a simple swing example where we are creating one button and adding it on the JFrame object inside the main() method.

```
import javax.swing.*;
public class FirstSwingExample {
public static void main(String[] args) {
JFrame f=new JFrame();//creating instance of JFrame

JButton b=new JButton("click");//creating instance of JButton
b.setBounds(130,100,100, 40);//x axis, y axis, width, height

f.add(b);//adding button in JFrame

f.setSize(400,500);//400 width and 500 height
f.setLayout(null);//using no layout managers
f.setVisible(true);//making the frame visible
}
}
```

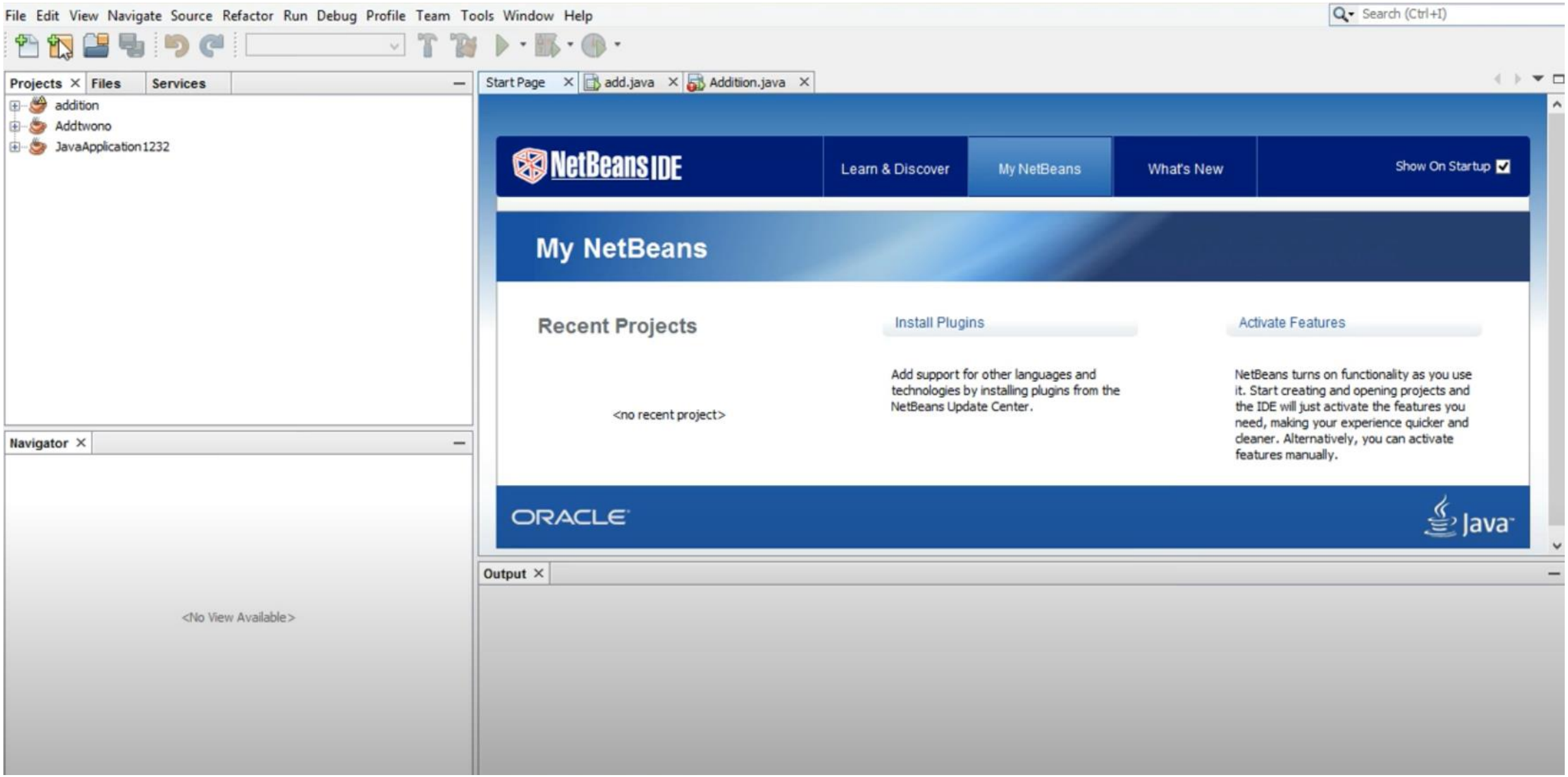
Demo on NetBeans: Addition of two number



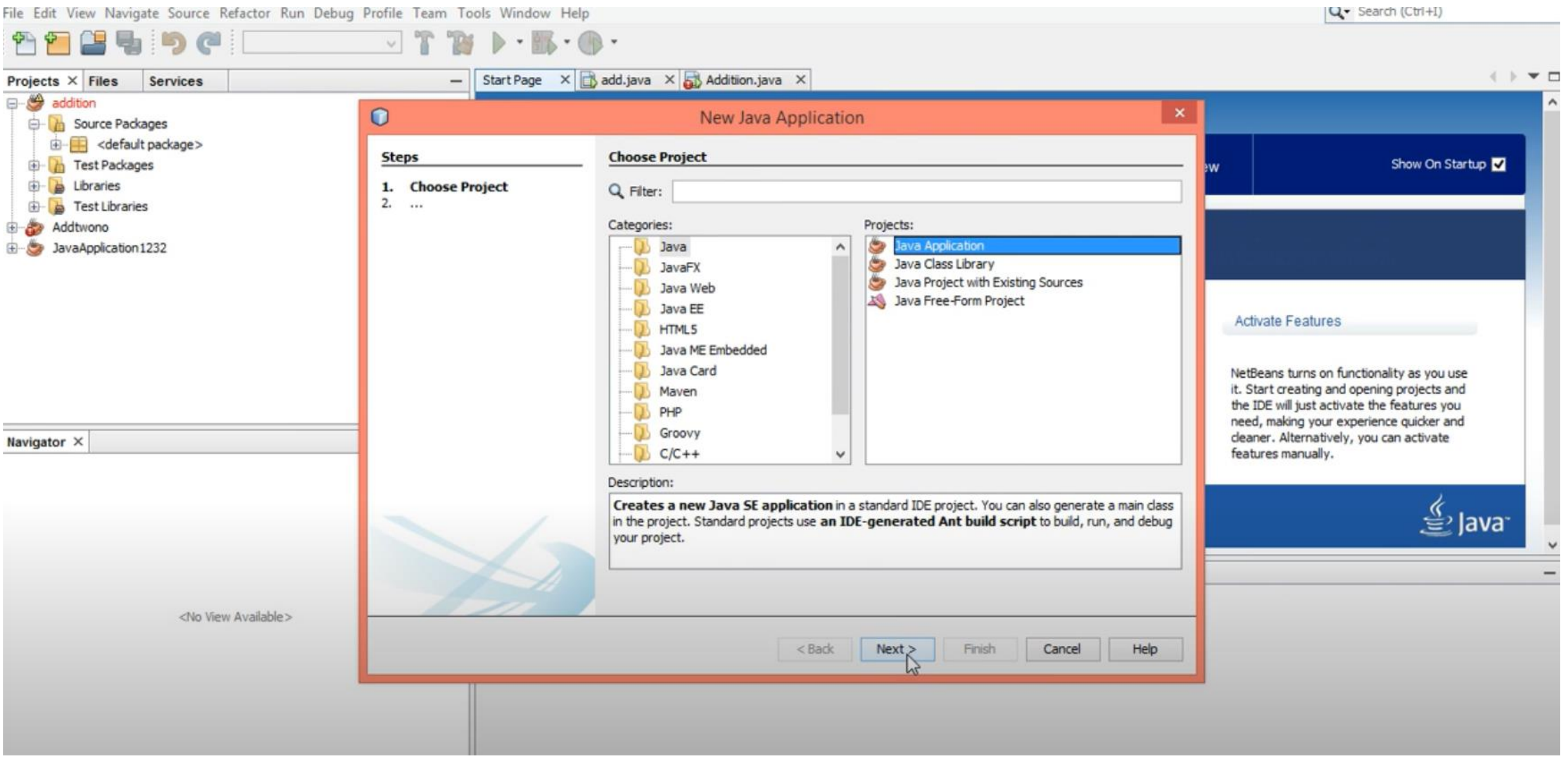
Demo on NetBeans: Addition of two number



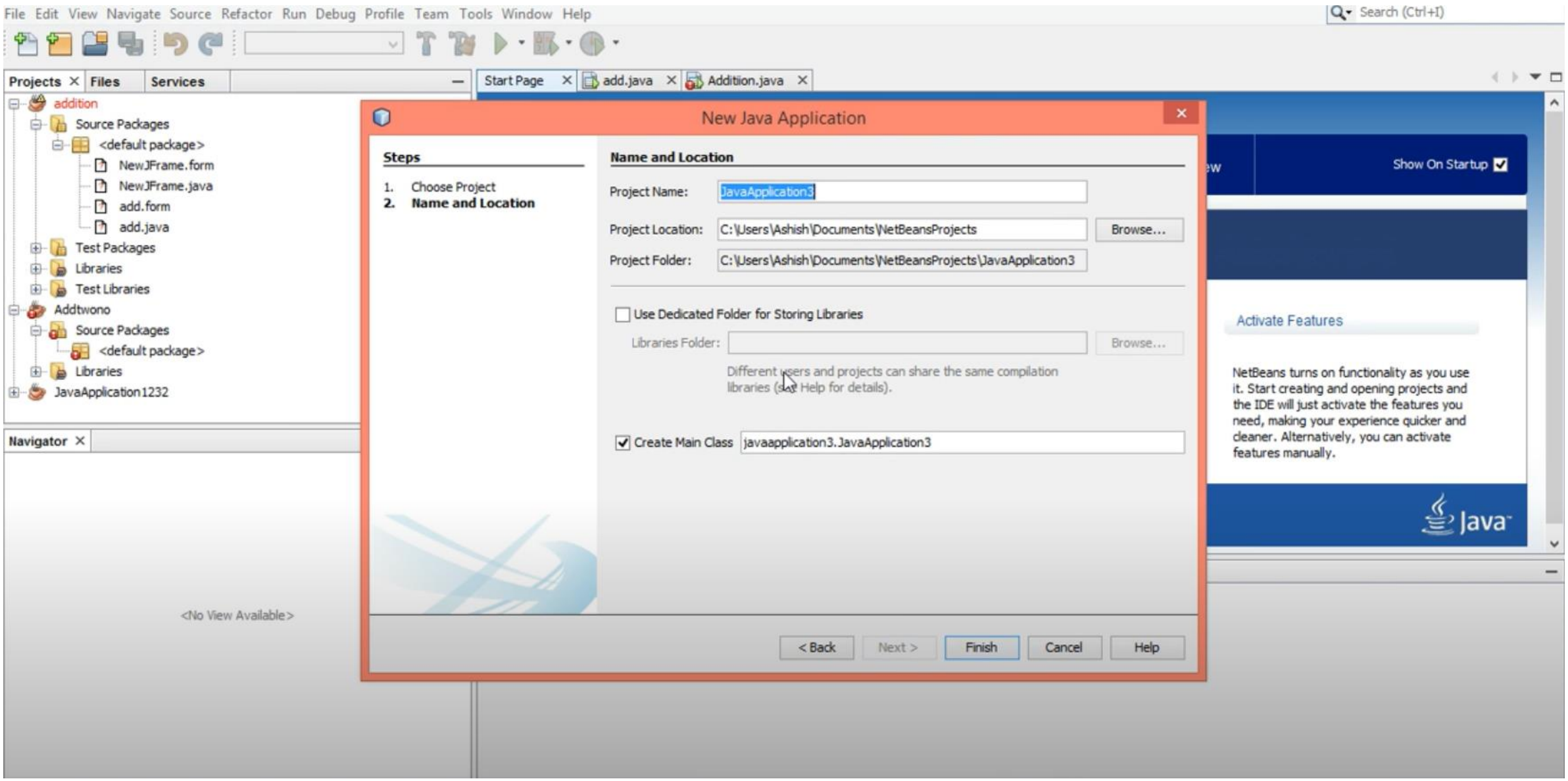
Demo on NetBeans: Addition of two number



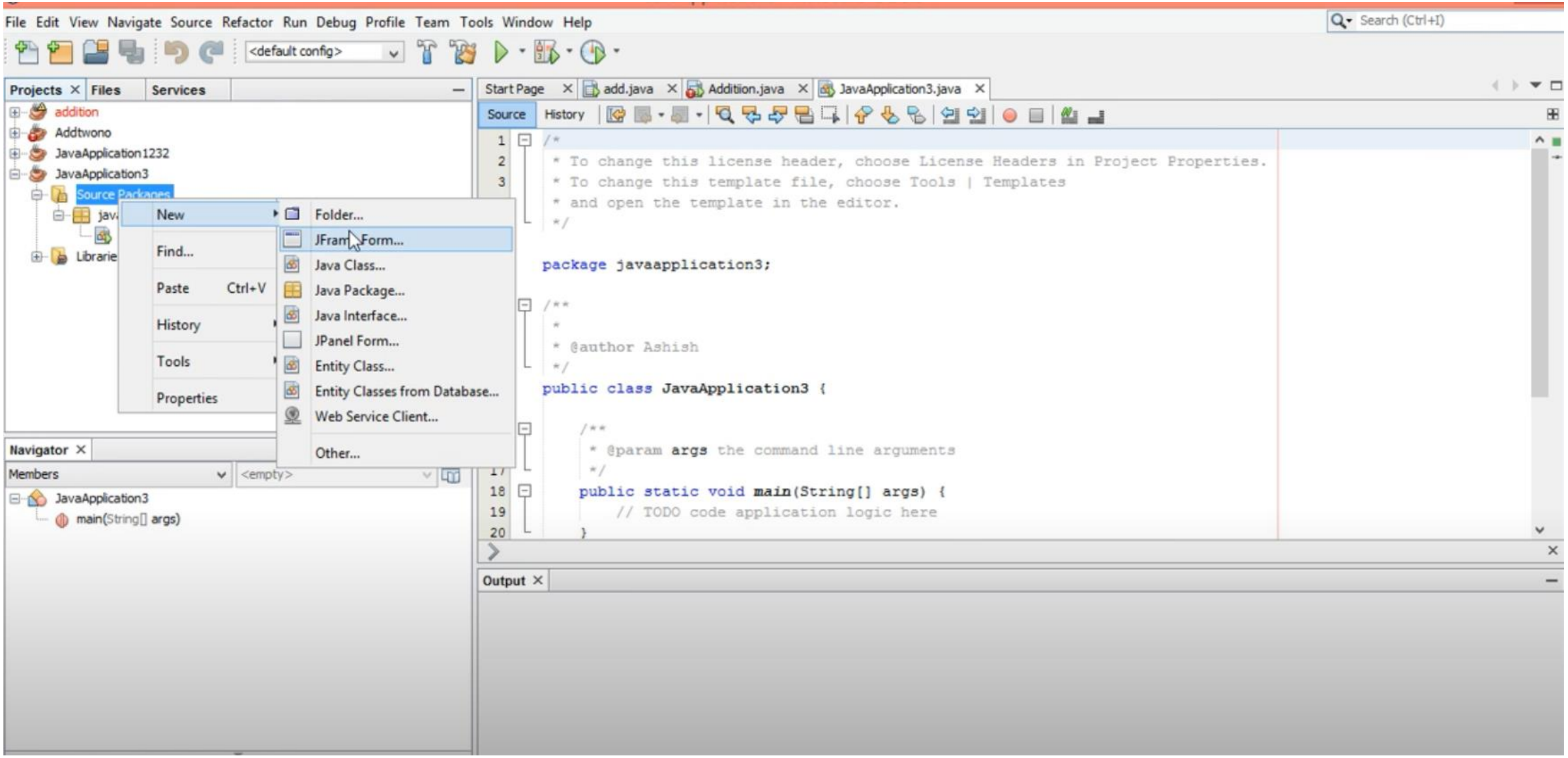
Demo on NetBeans: Addition of two number



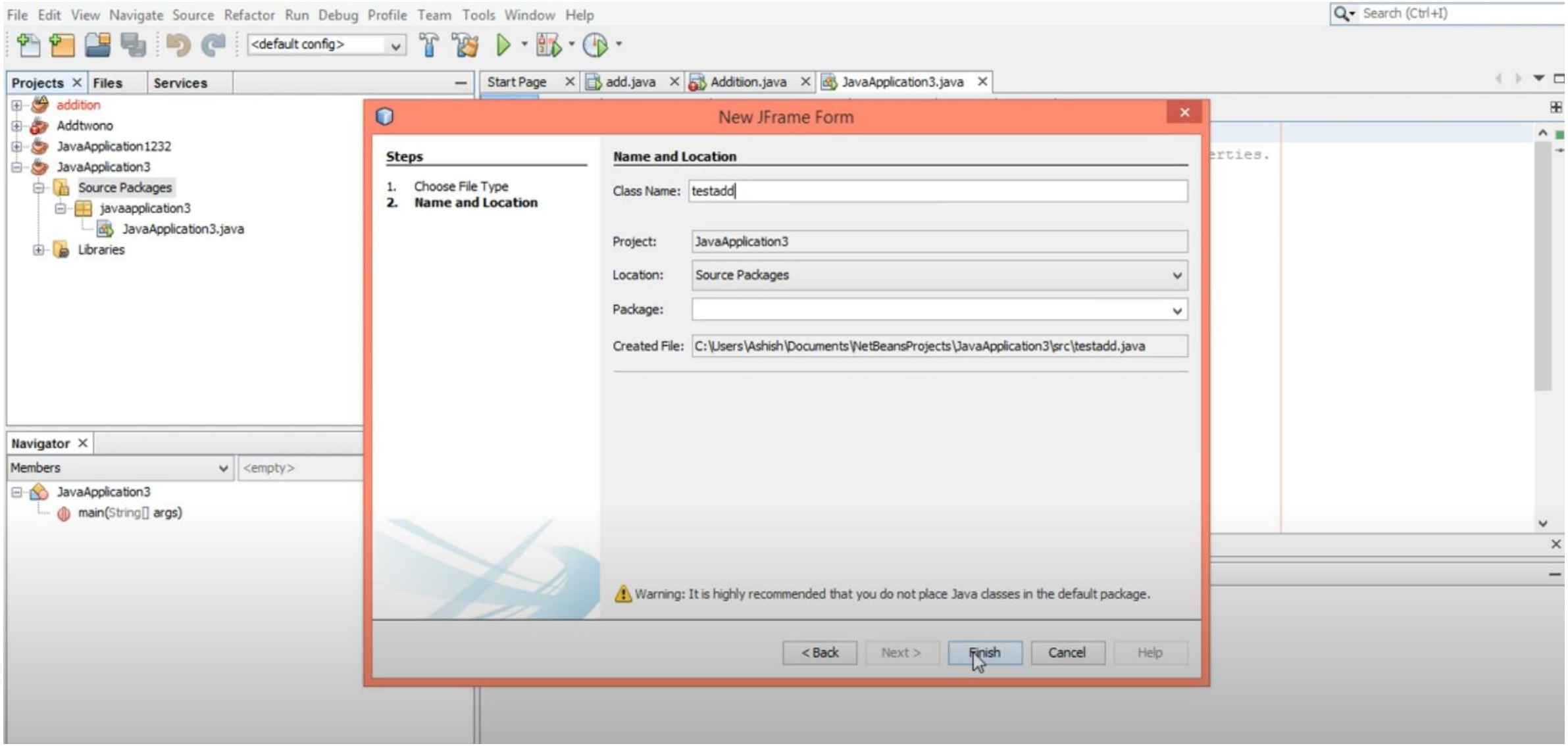
Demo on NetBeans: Addition of two number



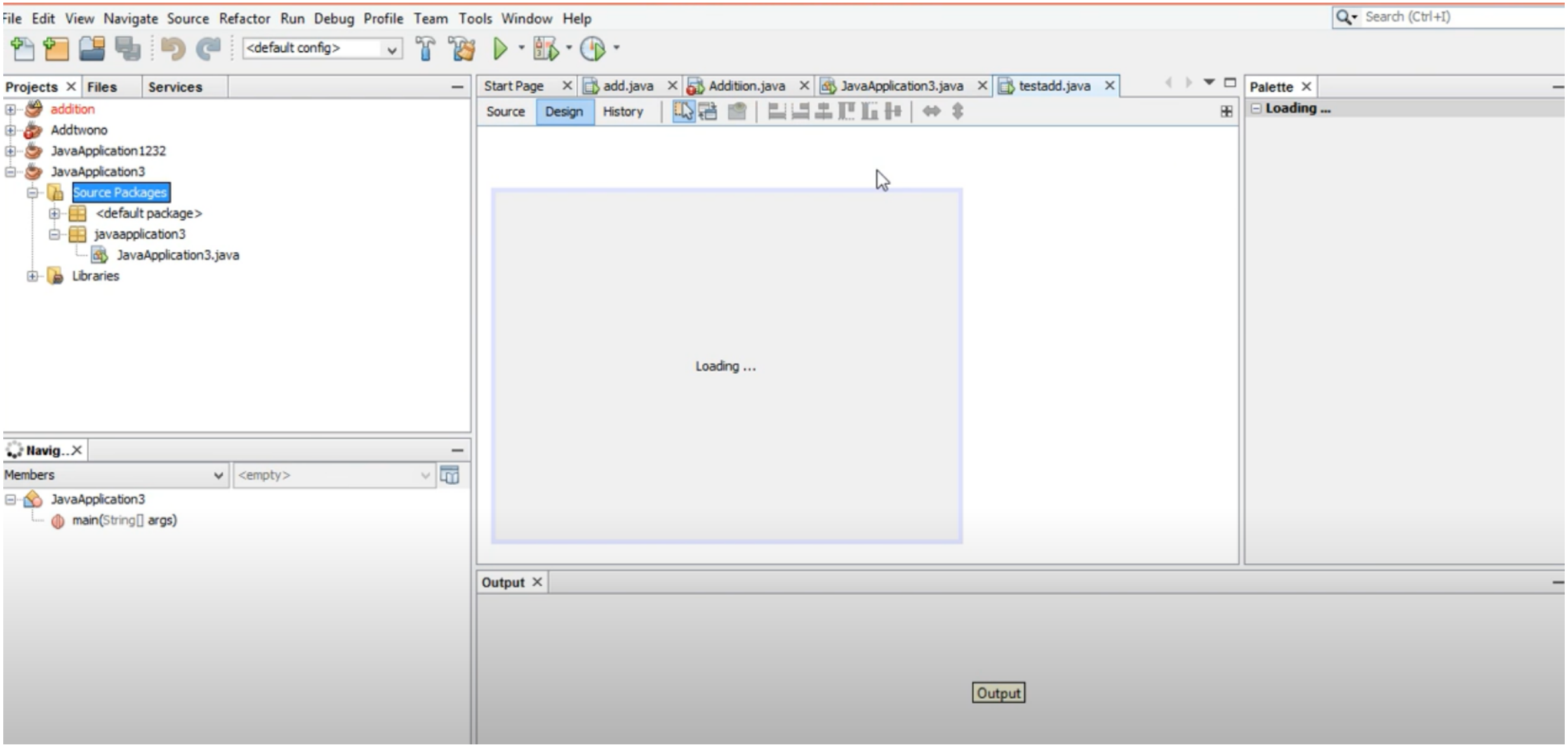
Demo on NetBeans: Addition of two number



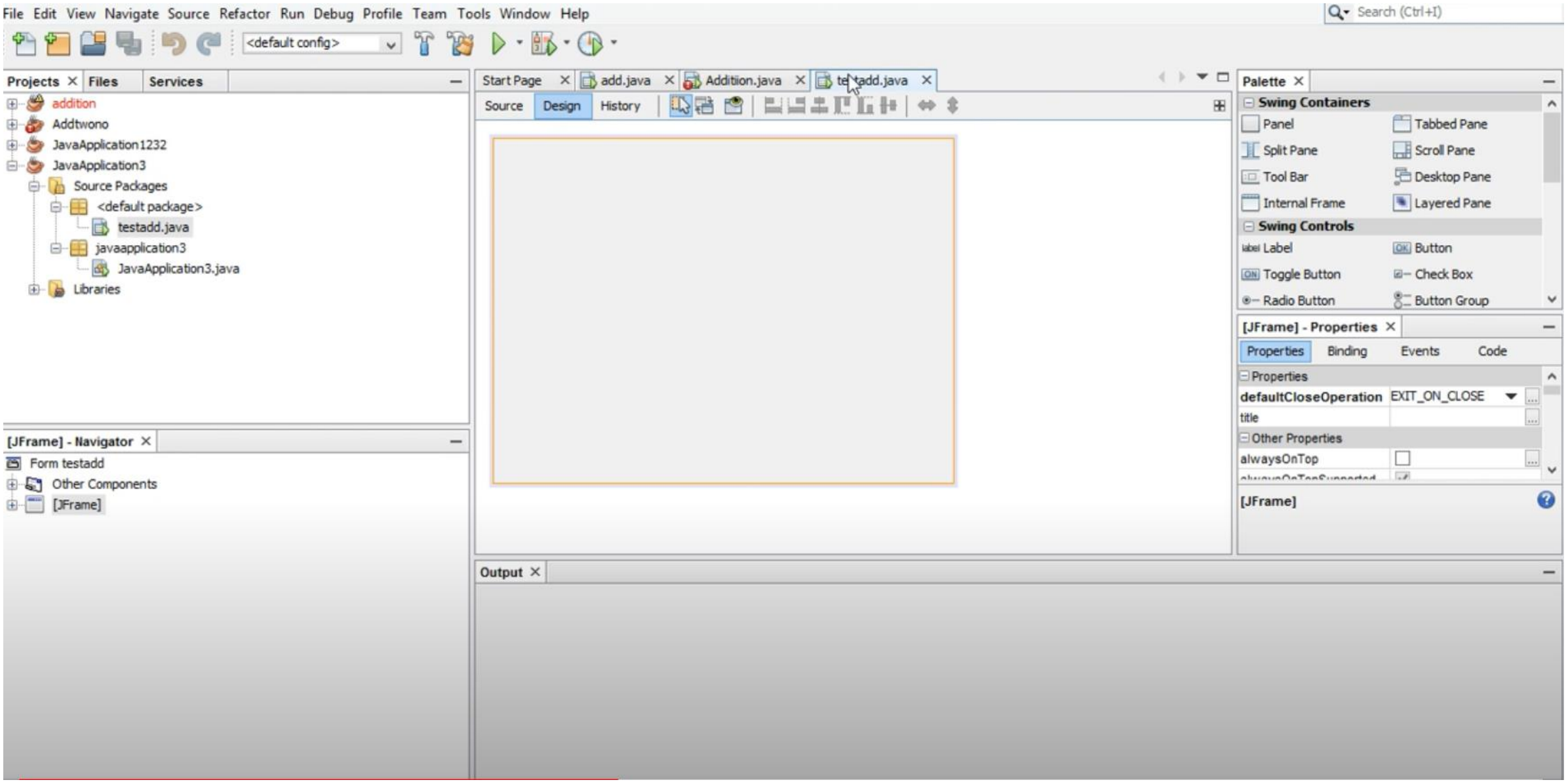
Demo on NetBeans: Addition of two number



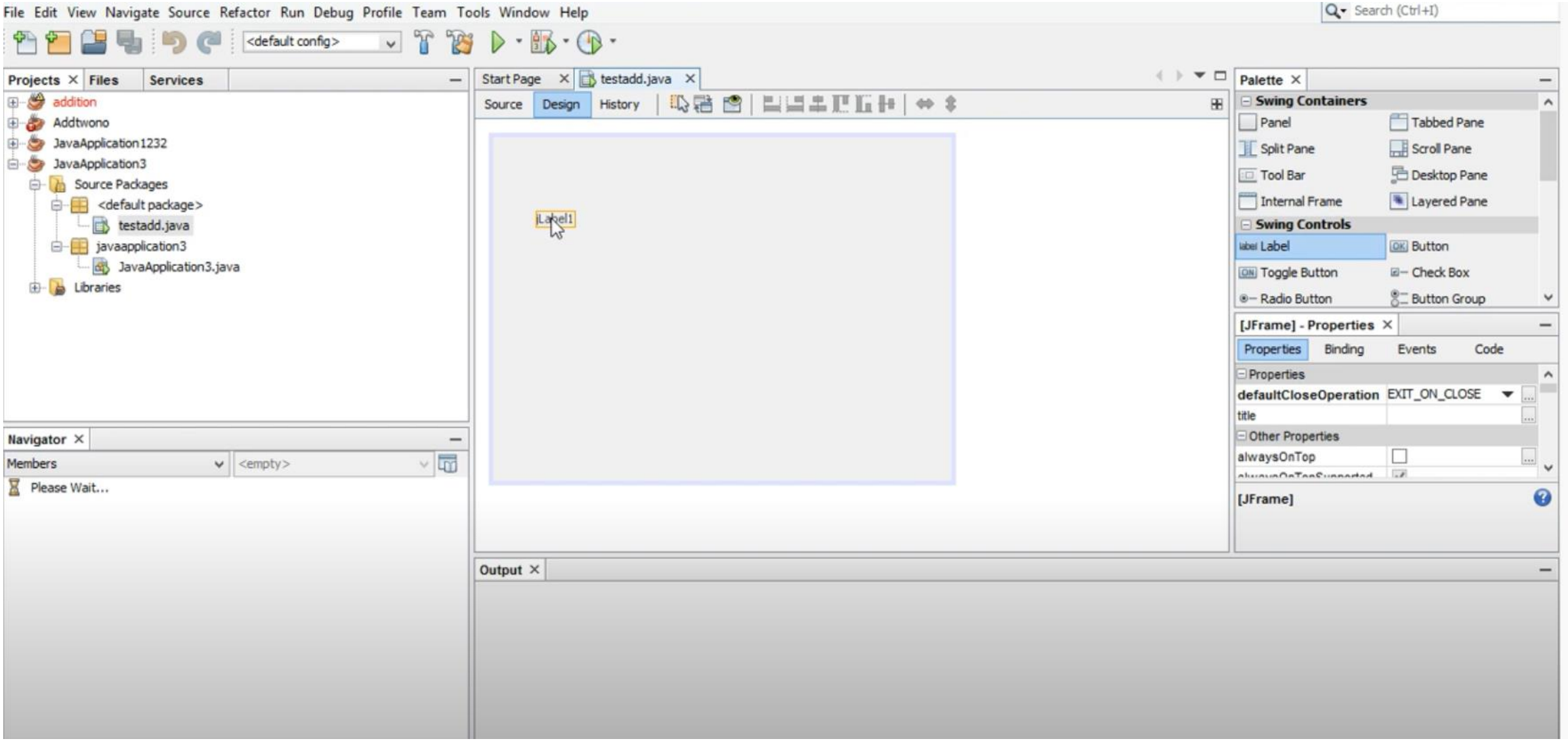
Demo on NetBeans: Addition of two number



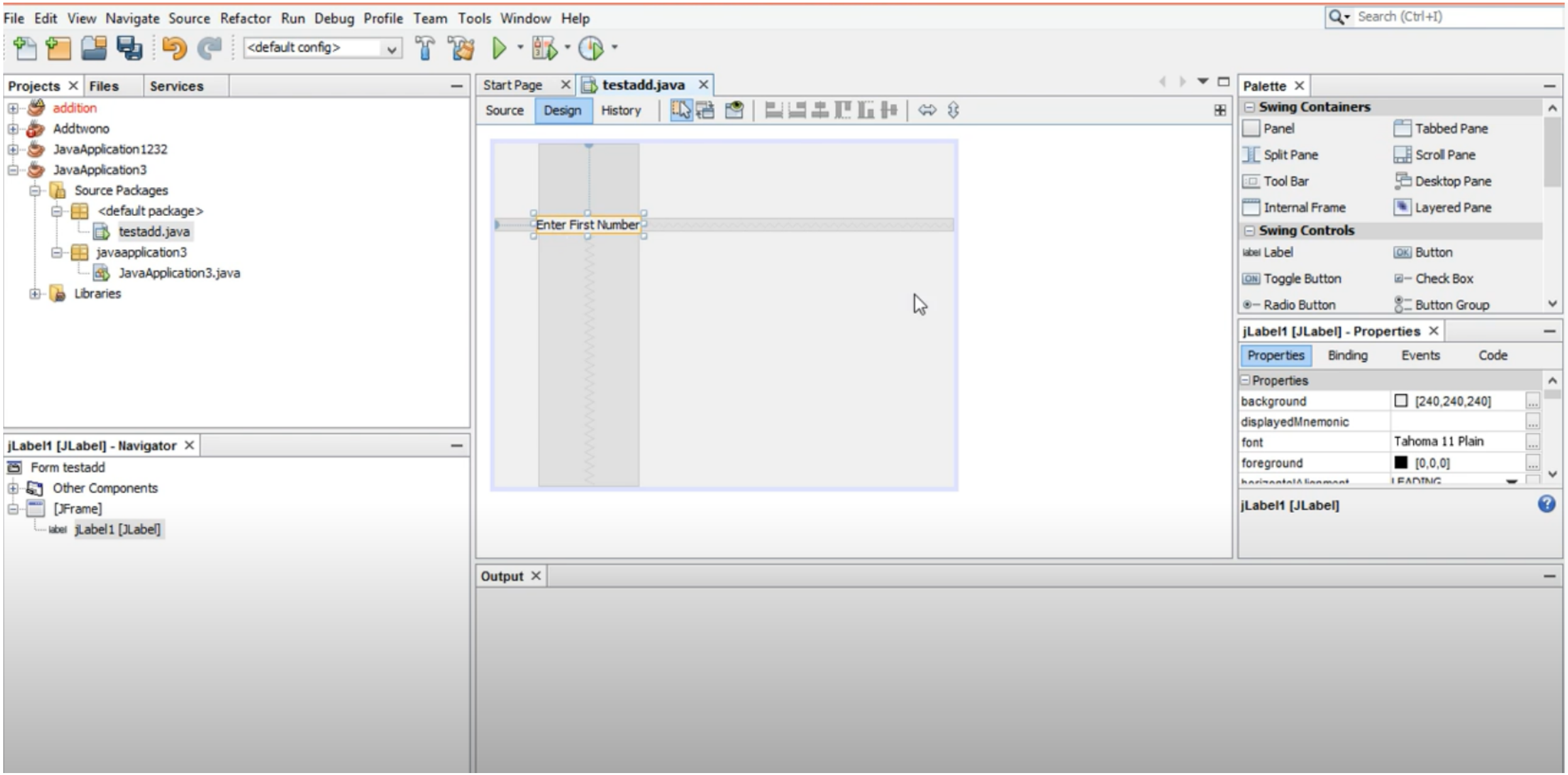
Demo on NetBeans: Addition of two number



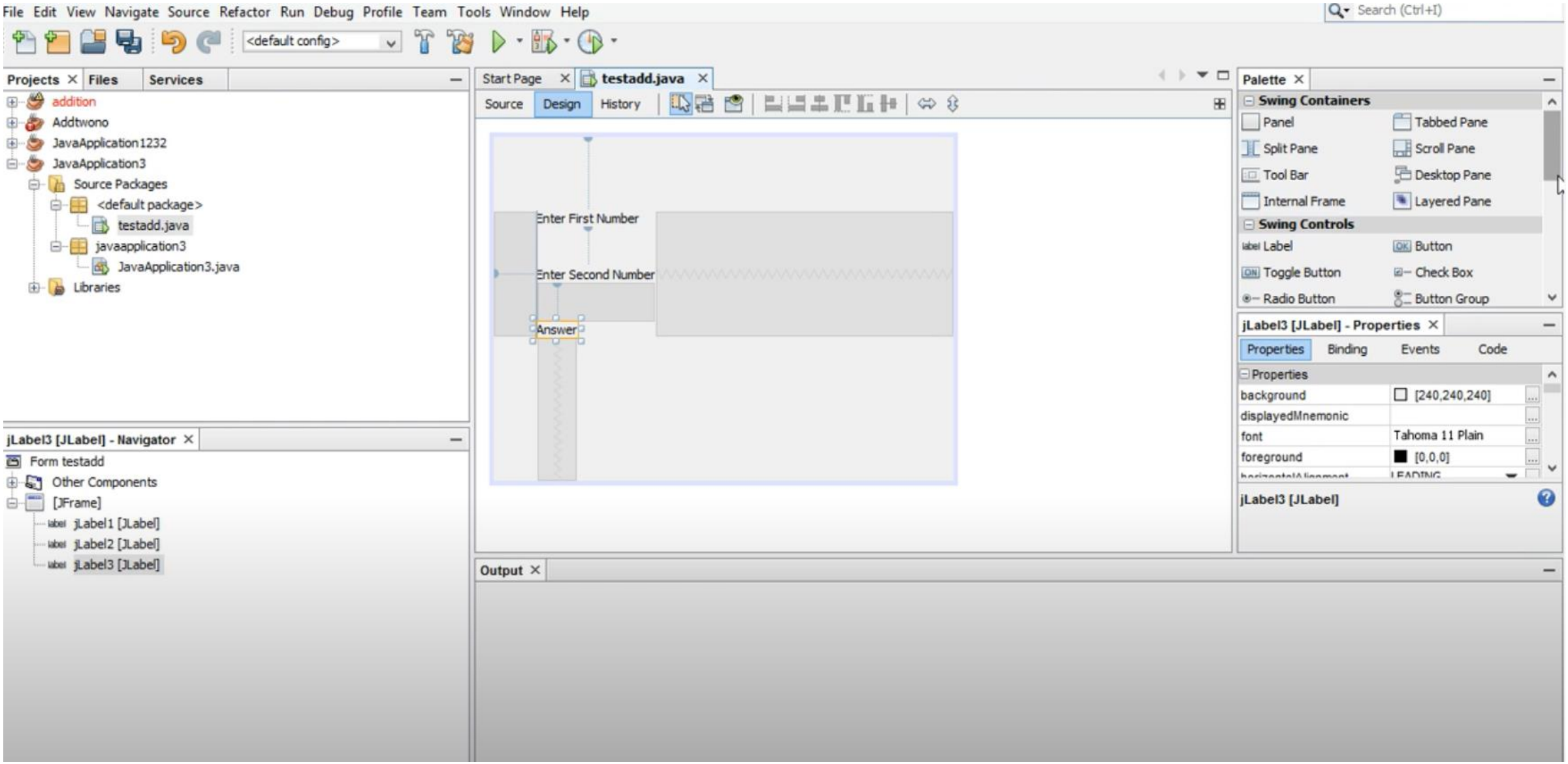
Demo on NetBeans: Addition of two number



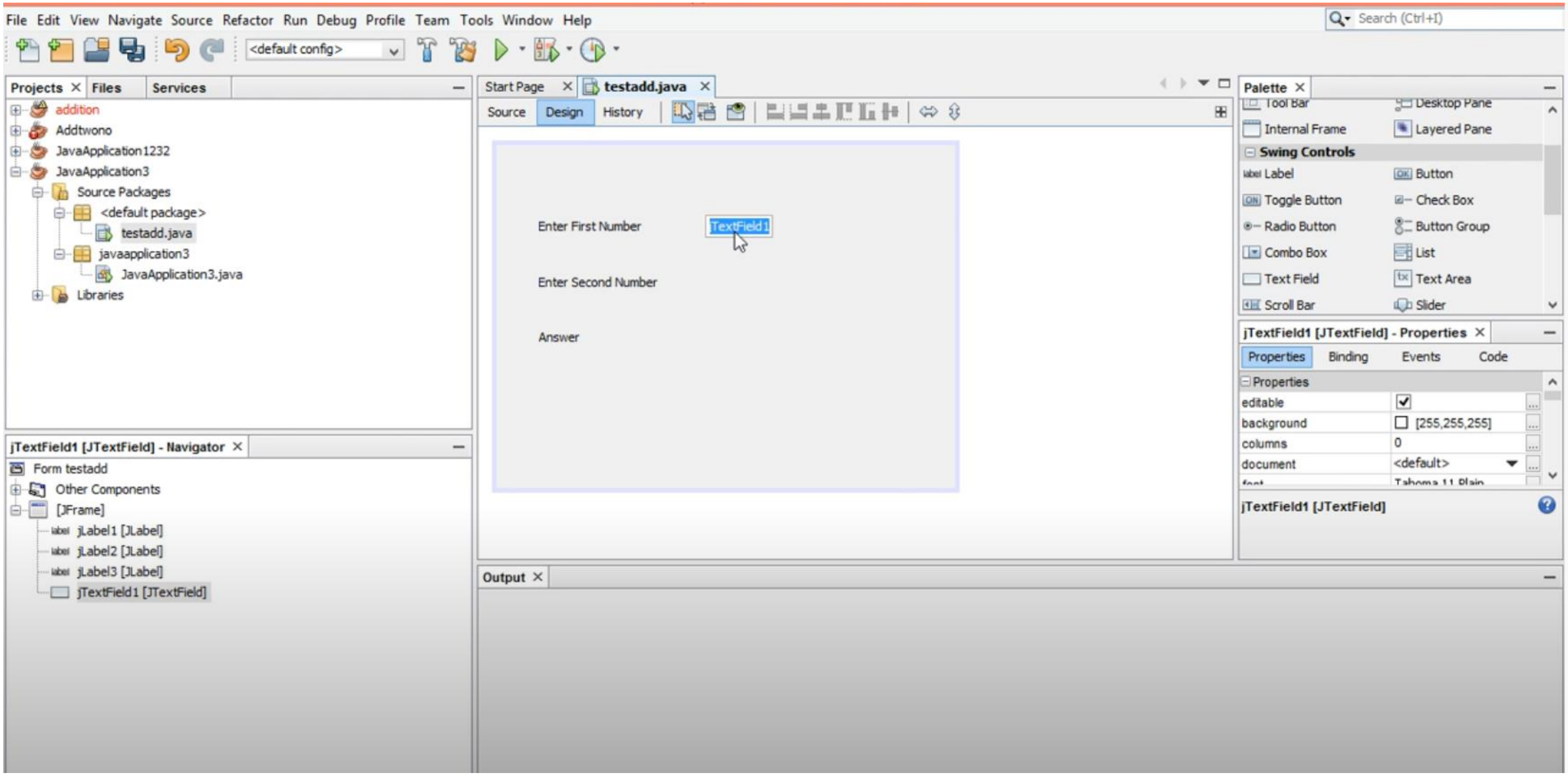
Demo on NetBeans: Addition of two number



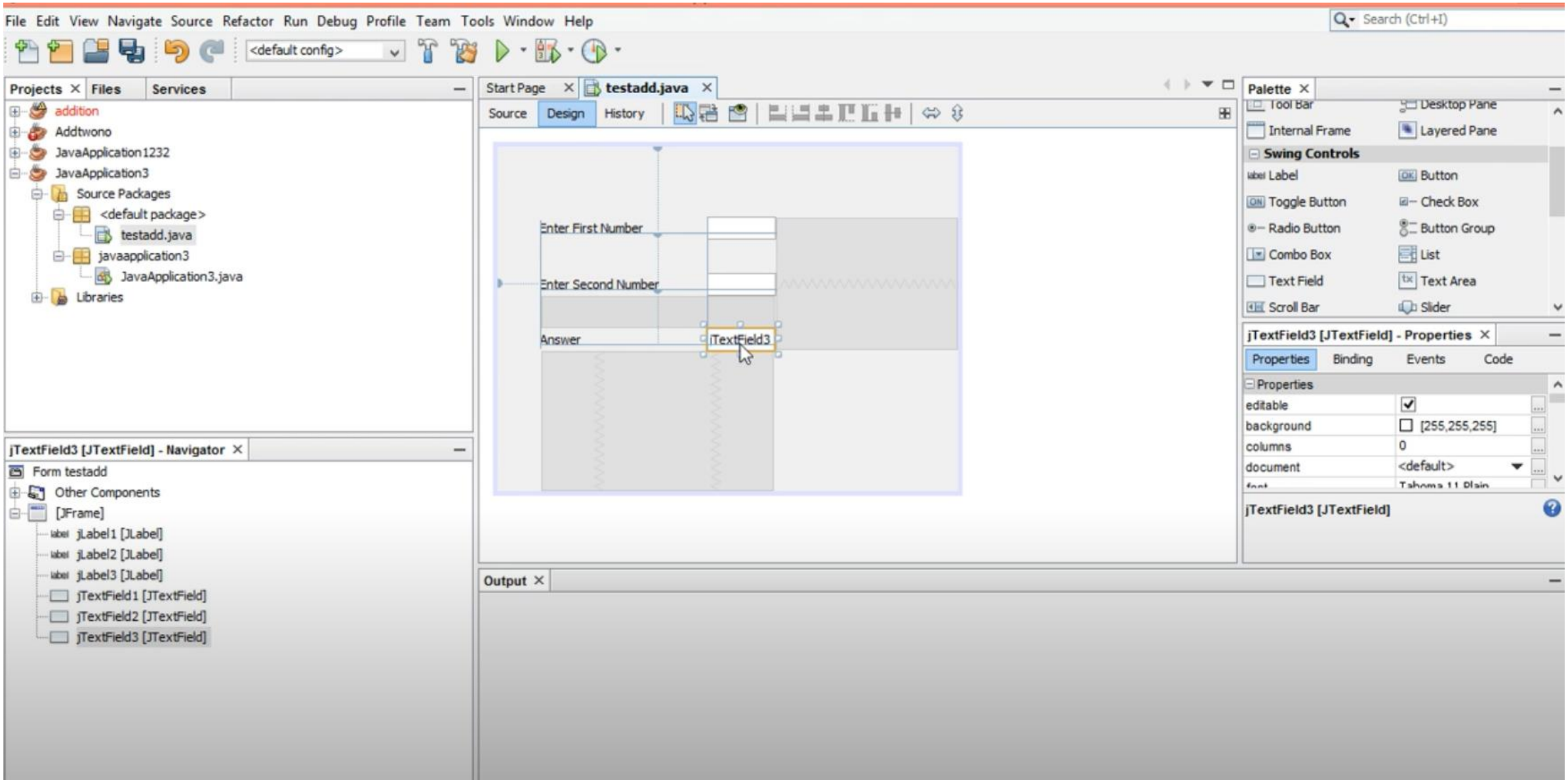
Demo on NetBeans: Addition of two number



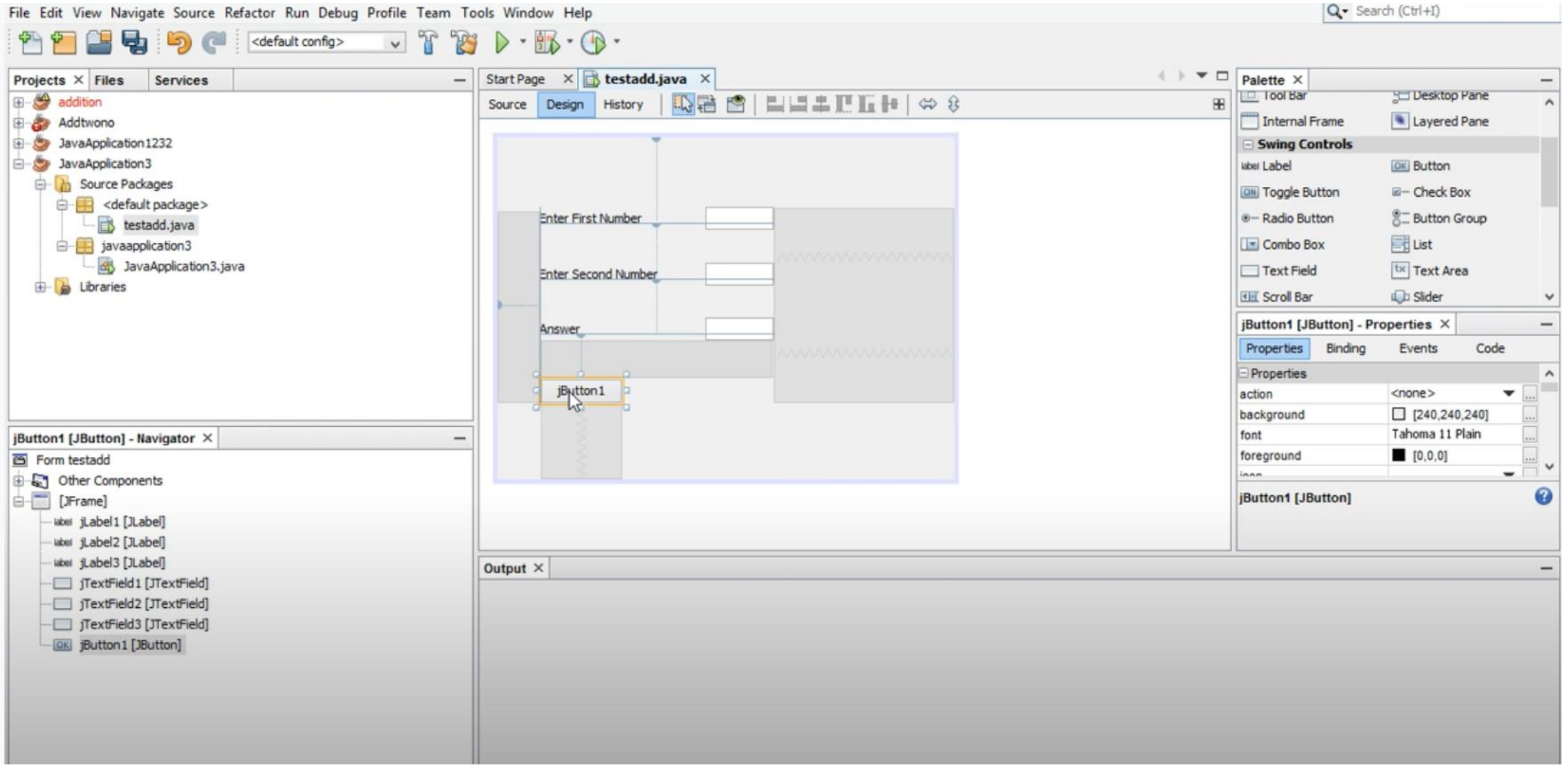
Demo on NetBeans: Addition of two number



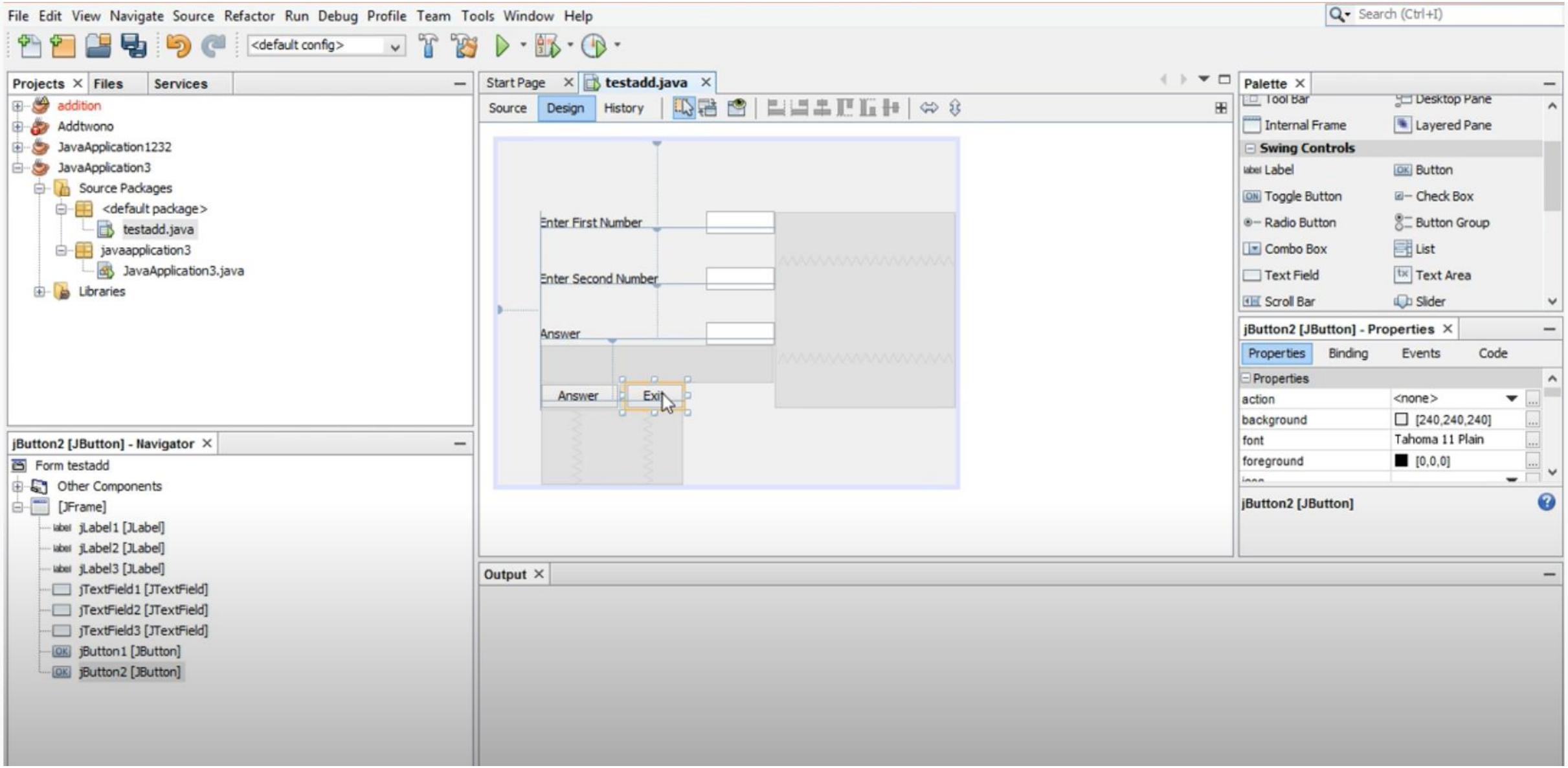
Demo on NetBeans: Addition of two number



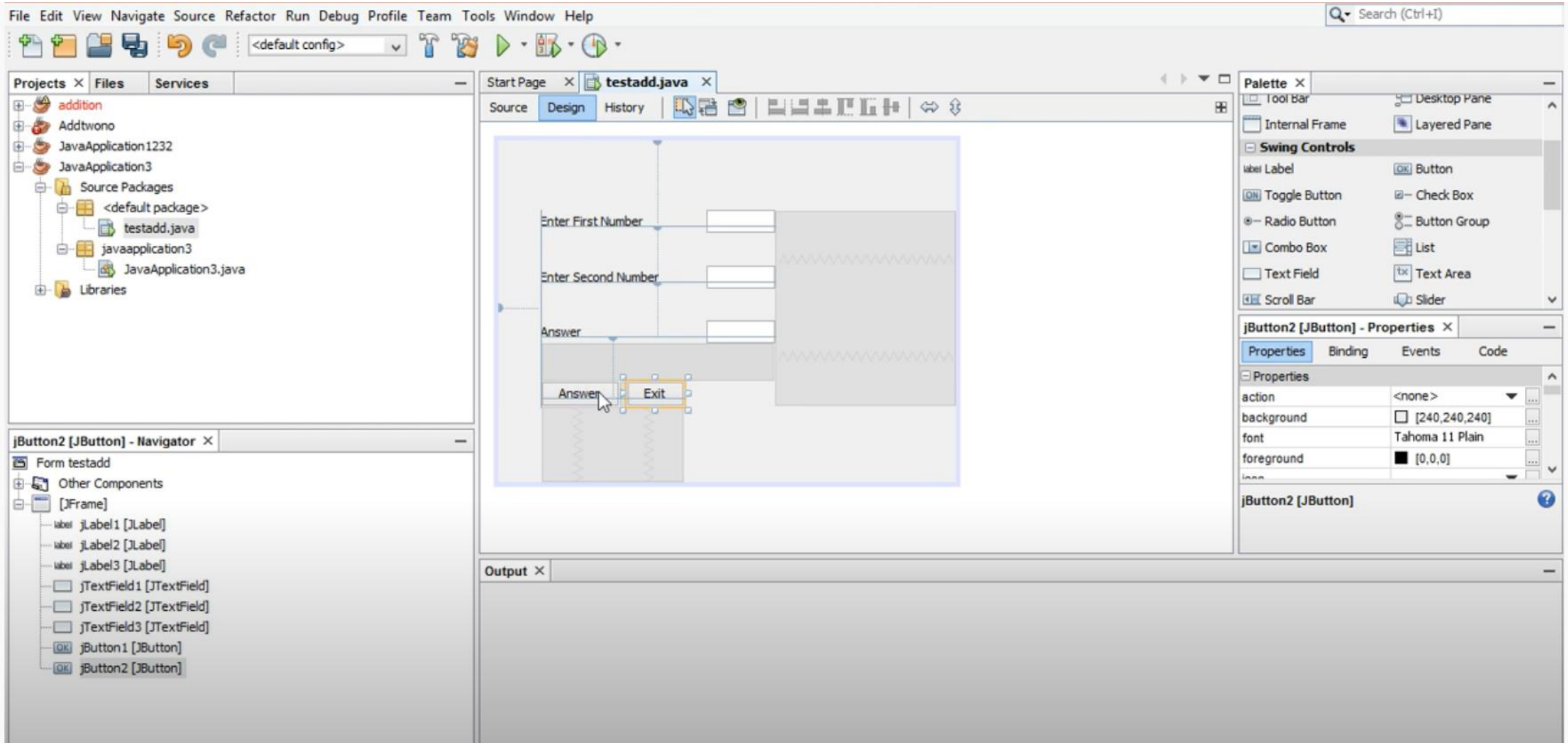
Demo on NetBeans: Addition of two number



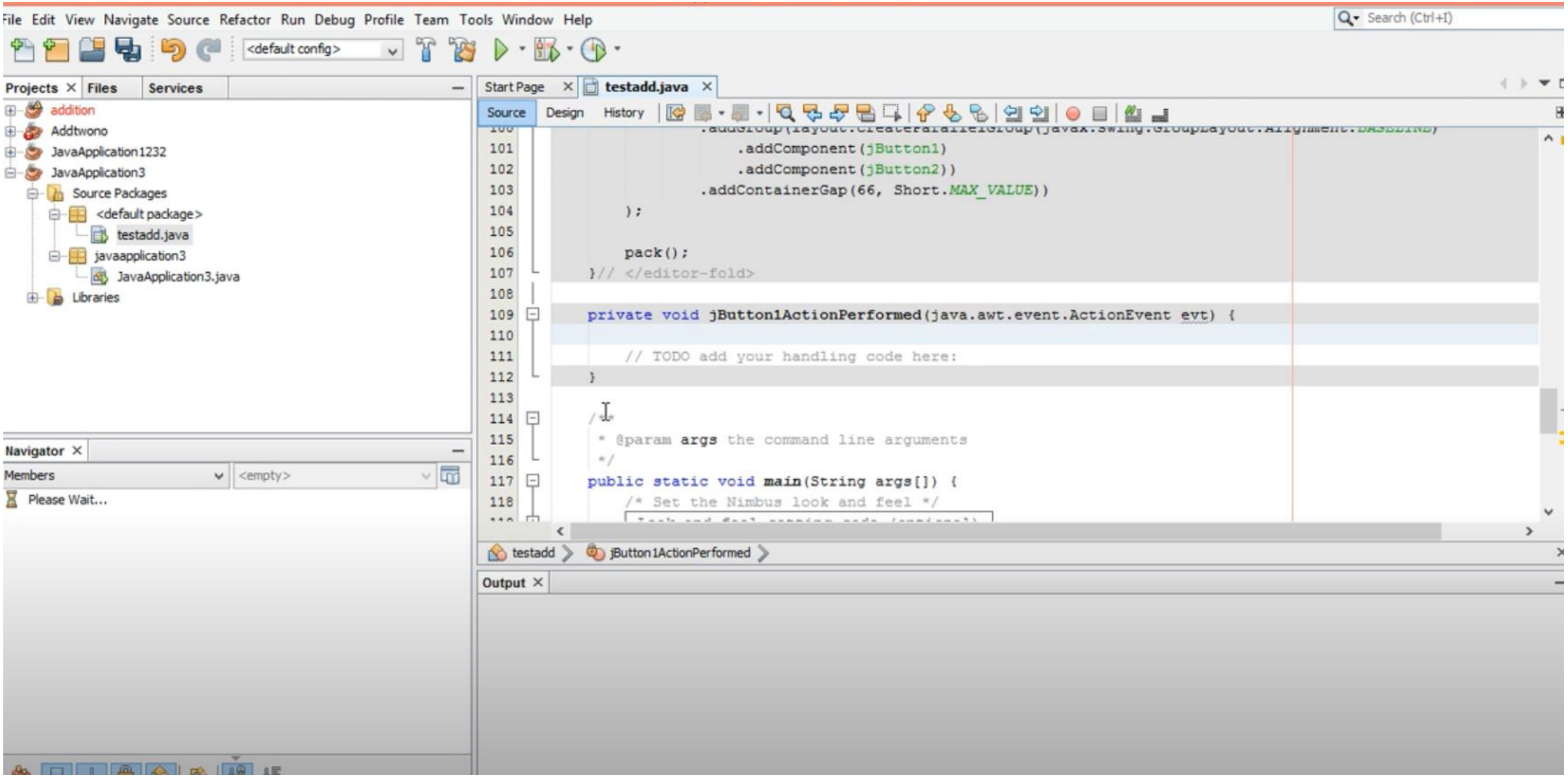
Demo on NetBeans: Addition of two number



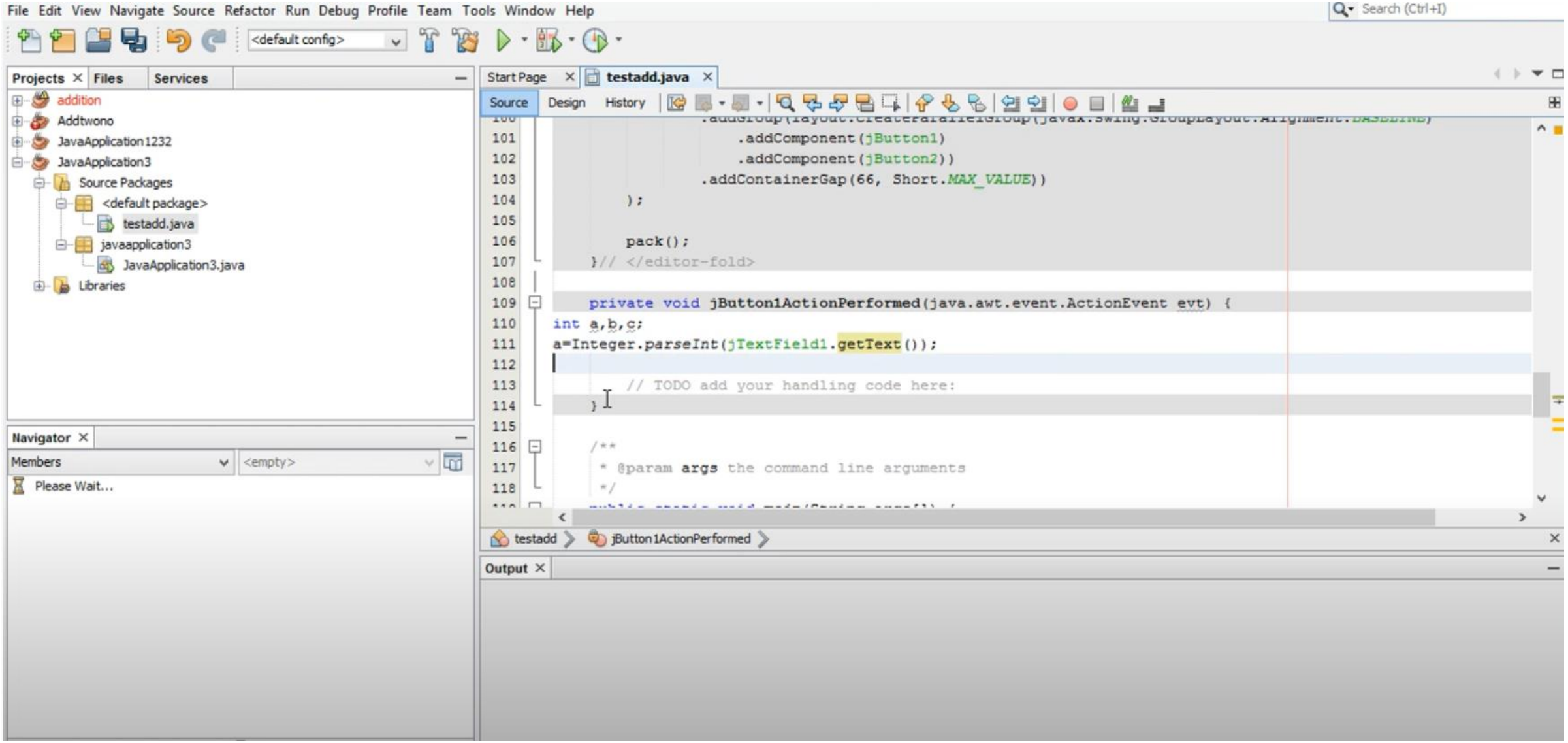
Demo on NetBeans: Addition of two number



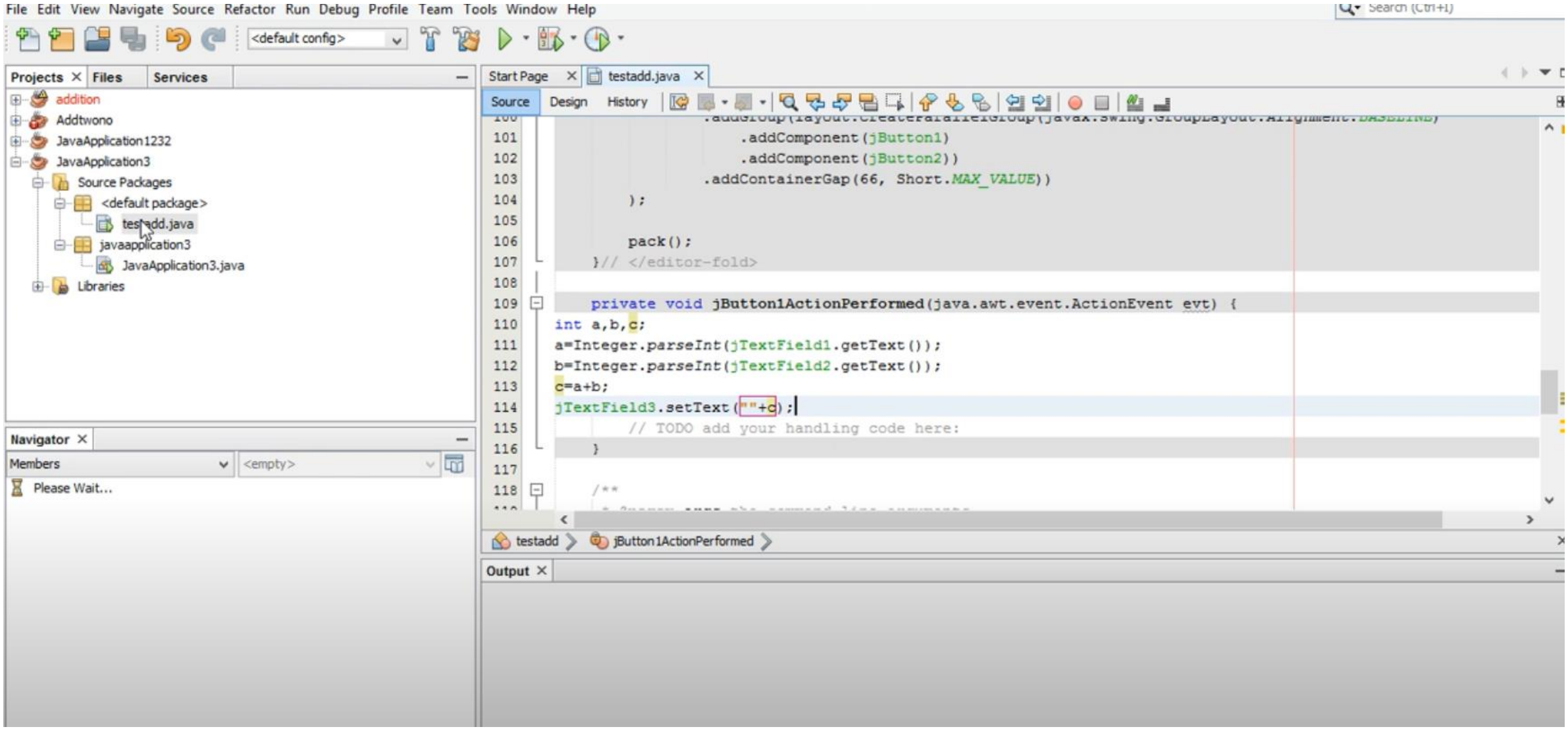
Demo on NetBeans: Addition of two number



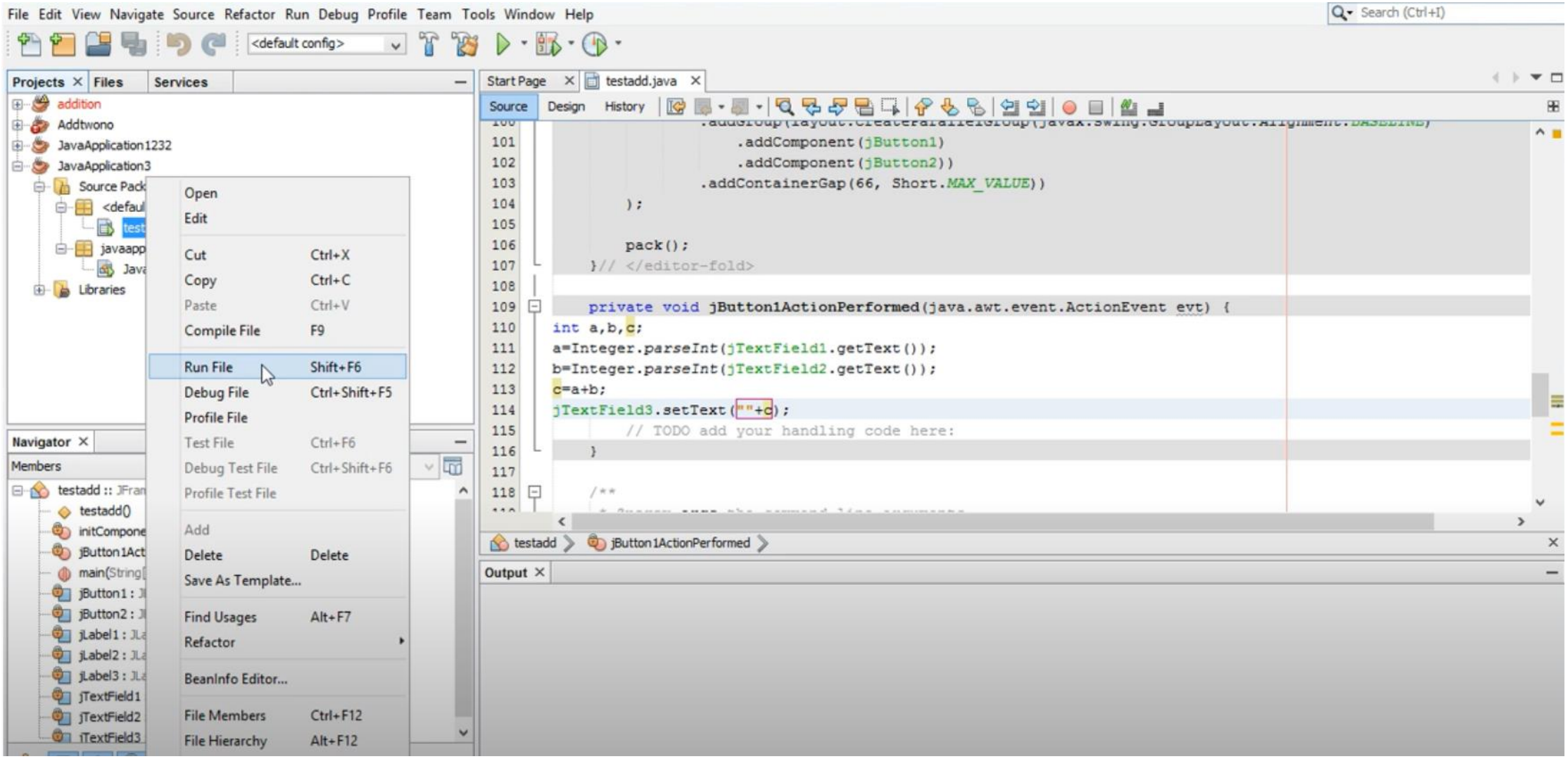
Demo on NetBeans: Addition of two number



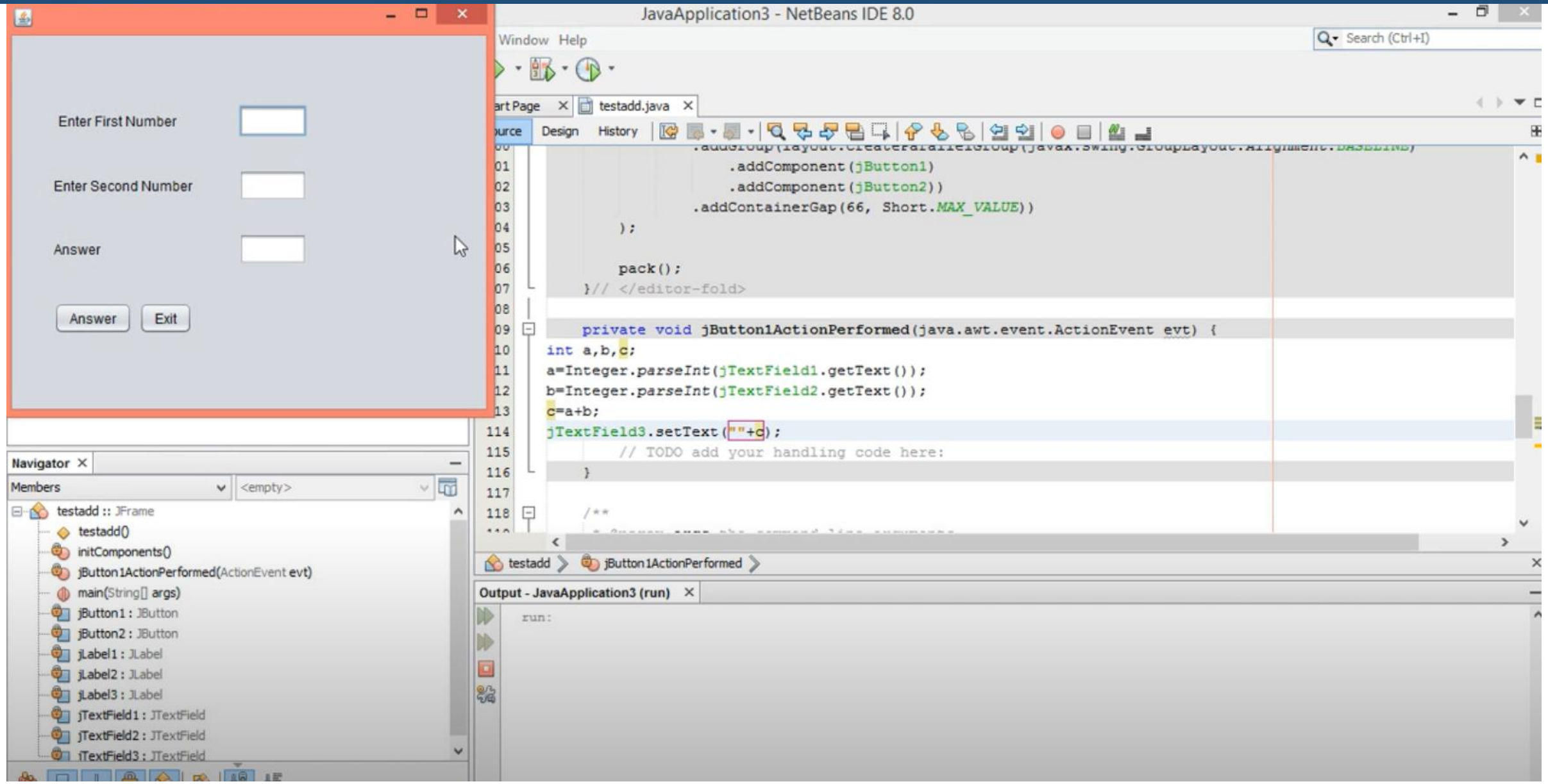
Demo on NetBeans: Addition of two number



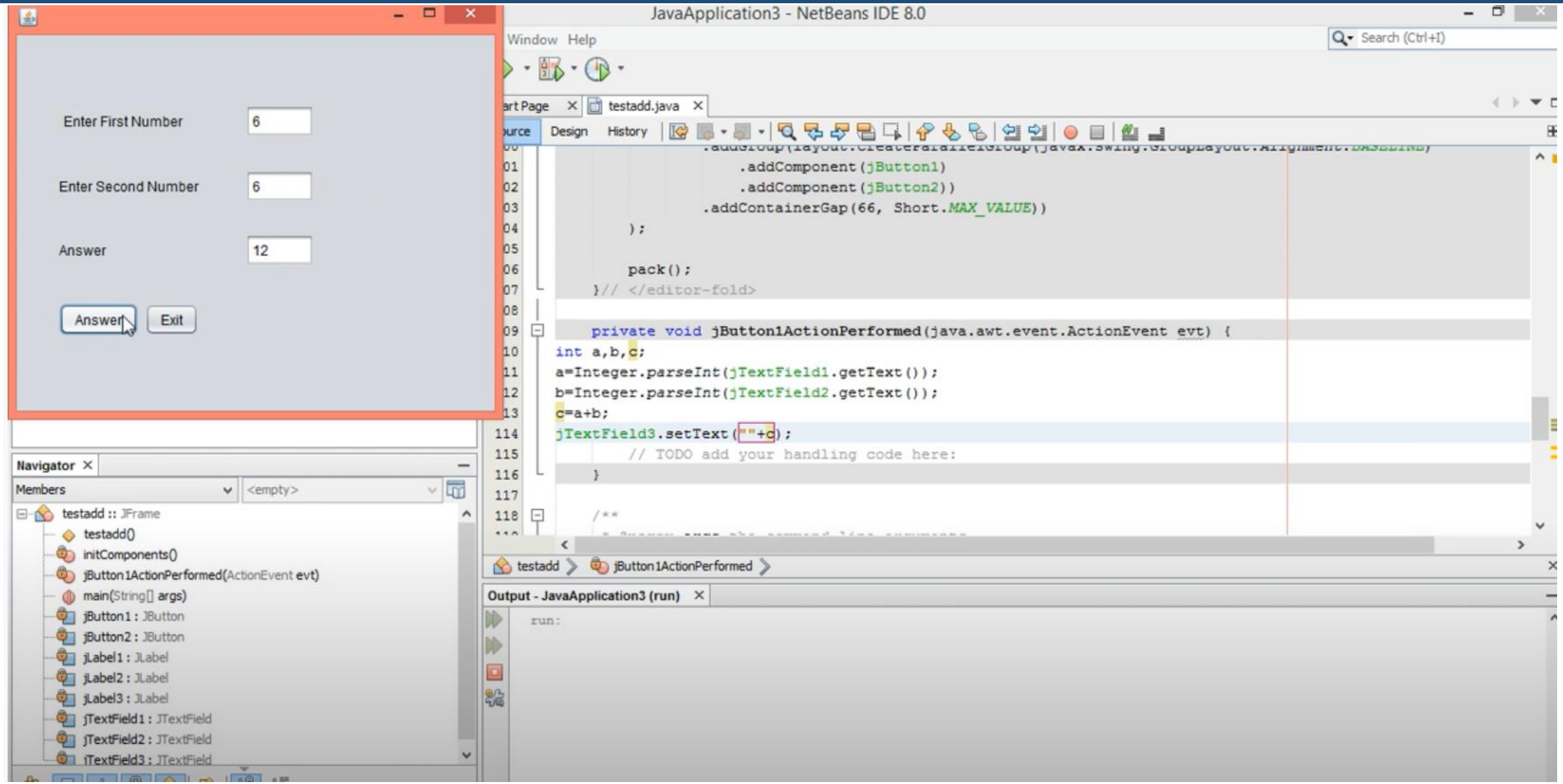
Demo on NetBeans: Addition of two number



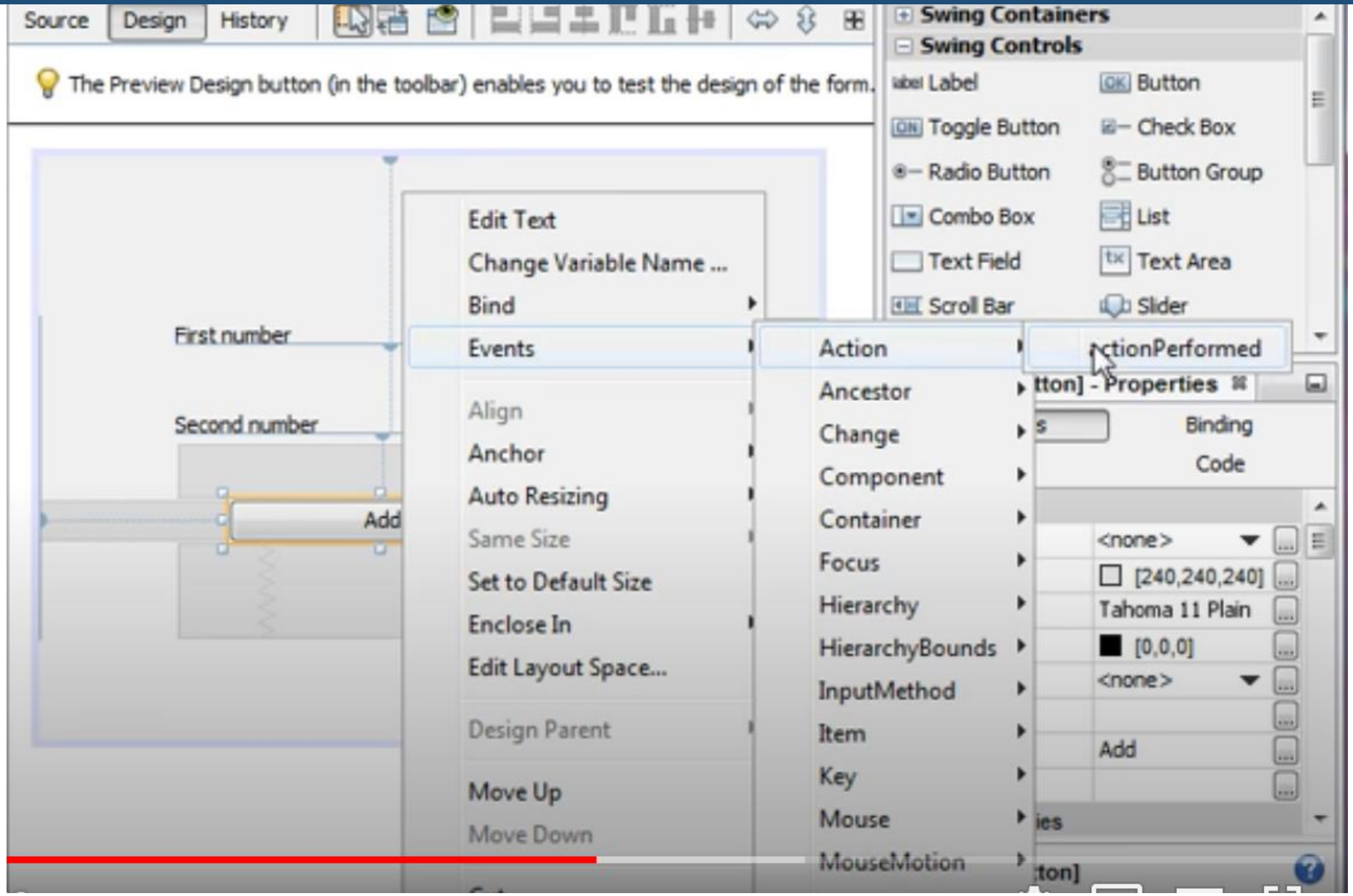
Demo on NetBeans: Addition of two number



Demo on NetBeans: Addition of two number



Demo on NetBeans: Addition of two number



Commonly used Methods of Component class

Method	Description
public void add(Component c)	add a component on another component.
public void setSize(int width,int height)	sets size of the component.
public void setLayout(LayoutManager m)	sets the layout manager for the component.
public void setVisible(boolean b)	sets the visibility of the component. It is by default false

Java JButton

- The JButton class is used to create a labeled button that has platform independent implementation. The application result in some action when the button is pushed. It inherits AbstractButton class.

JButton class declaration

- **Let's see the declaration for javax.swing.JButton class.**
- **public class JButton extends AbstractButton implements Accessible**

Commonly used Constructors:

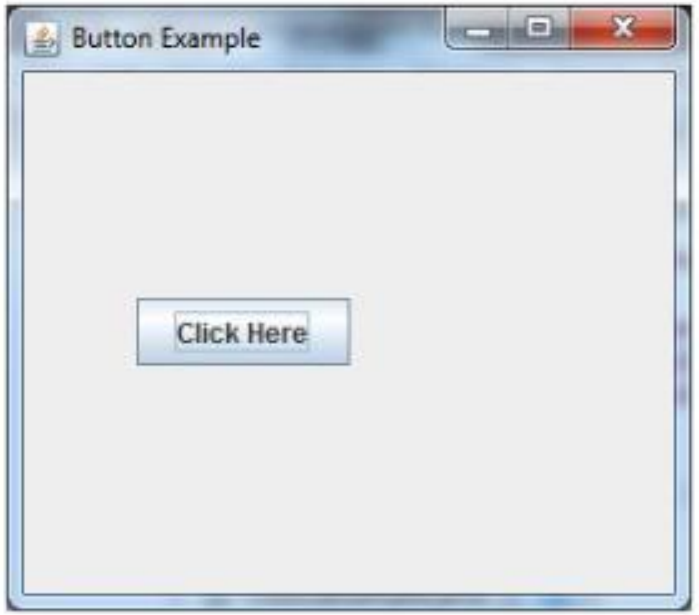
Constructor	Description
JButton()	It creates a button with no text and icon.
JButton(String s)	It creates a button with the specified text.
JButton(Icon i)	It creates a button with the specified icon object.

Commonly used Methods of AbstractButton class:

Methods	Description
void setText(String s)	It is used to set specified text on button
String getText()	It is used to return the text of the button.
void setEnabled(boolean b)	It is used to enable or disable the button.
void setIcon(Icon b)	It is used to set the specified Icon on the button.
Icon getIcon()	It is used to get the Icon of the button.
void setMnemonic(int a)	It is used to set the mnemonic on the button.
void addActionListener(ActionListener a)	It is used to add the action listener to this object.

Java JButton Example

```
import javax.swing.*;
public class ButtonExample {
public static void main(String[] args) {
    JFrame f=new JFrame("Button Example");
    JButton b=new JButton("Click Here");
    b.setBounds(50,100,95,30);
    f.add(b);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
}
```



Example of displaying image on the button:

```
import javax.swing.*;
public class ButtonExample{
ButtonExample(){
JFrame f=new JFrame("Button Example");
JButton b=new JButton(new ImageIcon("D:\\icon.png"));
b.setBounds(100,100,100, 40);
f.add(b);
f.setSize(300,400);
f.setLayout(null);
f.setVisible(true);
f.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
    }
public static void main(String[] args) {
    new ButtonExample();
}
}
```

Java JLabel

- The object of JLabel class is a component for placing text in a container. It is used to display a single line of read only text. The text can be changed by an application but a user cannot edit it directly. It inherits JComponent class.

JLabel class declaration

- **the declaration for javax.swing.JLabel class.**
- **public class JLabel extends JComponent implements SwingConstants , Accessible**

Commonly used Constructors:

Constructor	Description
JLabel()	Creates a JLabel instance with no image and with an empty string for the title.
JLabel(String s)	Creates a JLabel instance with the specified text.
JLabel(Icon i)	Creates a JLabel instance with the specified image.
JLabel(String s, Icon i, int horizontalAlignment)	Creates a JLabel instance with the specified text, image, and horizontal alignment

Commonly used Methods:

Methods	Description
String getText()	it returns the text string that a label displays.
void setText(String text)	It defines the single line of text this component will display.
void setHorizontalAlignment(int alignment)	It sets the alignment of the label's contents along the X axis.
Icon getIcon()	It returns the graphic image that the label displays.
int getHorizontalAlignment()	It returns the alignment of the label's contents along the X axis

Java JLabel Example:

```
import javax.swing.*;
class LabelExample
{
public static void main(String args[])
    {
    JFrame f= new JFrame("Label Example");
    JLabel l1,l2;
    l1=new JLabel("First Label.");
    l1.setBounds(50,50, 100,30);
    l2=new JLabel("Second Label.");
    l2.setBounds(50,100, 100,30);
    f.add(l1); f.add(l2);
    f.setSize(300,300);
    f.setLayout(null);
    f.setVisible(true);
    }
    }
```

Java JTextField

- The object of a JTextField class is a text component that allows the editing of a single line text. It inherits JTextComponent class.

JTextField class declaration

the declaration for javax.swing.JTextField class.

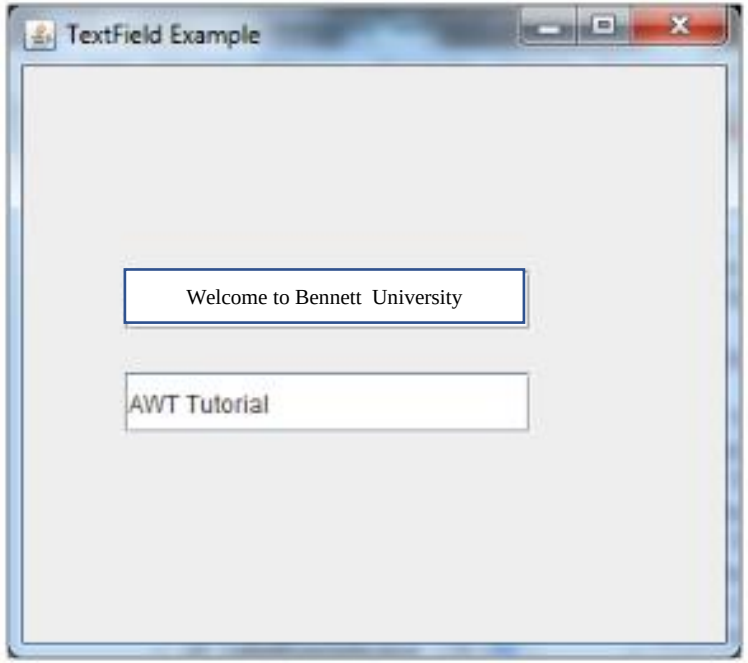
public class JTextField extends JTextComponent implements SwingConstants

Commonly used Constructors:

Constructor	Description
JTextField()	Creates a new TextField
JTextField(String text)	Creates a new TextField initialized with the specified text.
JTextField(String text, int columns)	Creates a new TextField initialized with the specified text and columns.
JTextField(int columns)	Creates a new empty TextField with the specified number of columns.

Java JTextField Example

```
import javax.swing.*;
class TextFieldExample
{
public static void main(String args[])
    {
    JFrame f= new JFrame("TextField Example");
    JTextField t1,t2;
    t1=new JTextField("  "Welcome to Bennett University"  ");
    t1.setBounds(50,100, 200,30);
    t2=new JTextField("AWT Tutorial");
    t2.setBounds(50,150, 200,30);
    f.add(t1); f.add(t2);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
    }
    }
```



Java JScrollPane

- A JscrollPane is used to make scrollable view of a component. When screen size is limited, we use a scroll pane to display a large component or a component whose size can change dynamically.

Constructors

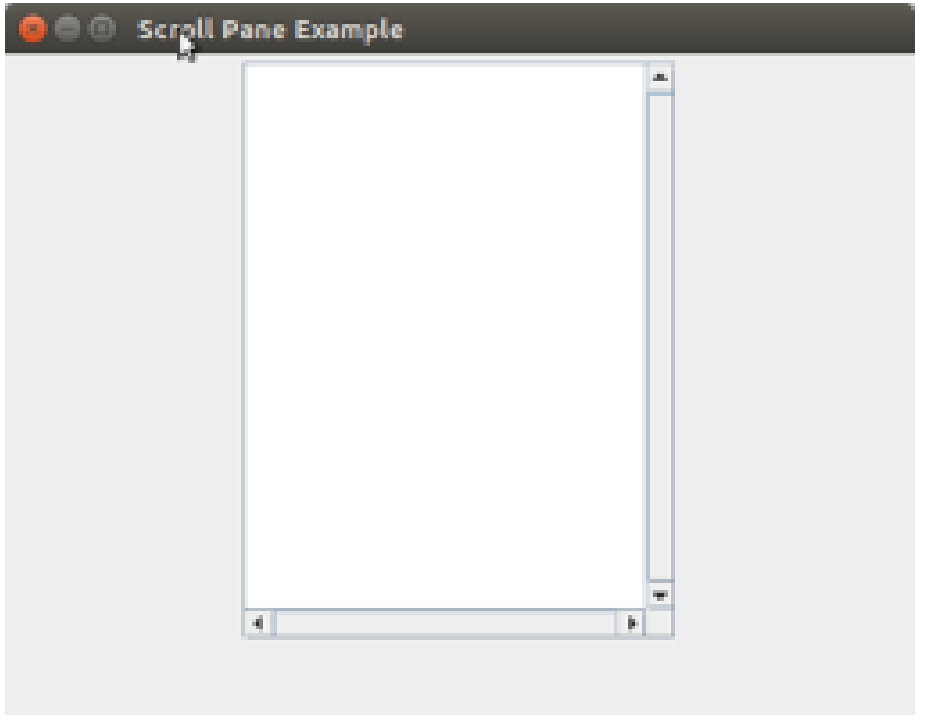
Constructor	Purpose
JScrollPane()	It creates a scroll pane. The Component parameter, when present, sets the scroll pane's client. The two int parameters, when present, set the vertical and horizontal scroll bar policies (respectively).
JScrollPane(Component)	
JScrollPane(int, int)	
JScrollPane(Component, int, int)	

Useful Methods

Modifier	Method	Description
void	setColumnHeaderView(Component)	It sets the column header for the scroll pane.
void	setRowHeaderView(Component)	It sets the row header for the scroll pane.
void	setCorner(String, Component)	It sets or gets the specified corner. The int parameter specifies which corner and must be one of the following constants defined in ScrollPaneConstants: UPPER_LEFT_CORNER, UPPER_RIGHT_CORNER, LOWER_LEFT_CORNER, LOWER_RIGHT_CORNER, LOWER_LEADING_CORNER, LOWER_TRAILING_CORNER, UPPER_LEADING_CORNER, UPPER_TRAILING_CORNER.
Component	getCorner(String)	
void	setViewportView(Component)	Set the scroll pane's client.

```
import java.awt.FlowLayout;
import javax.swing.JFrame;
import javax.swing.JScrollPane;
import javax.swing.JtextArea;

public class JScrollPaneExample {
   private static final long serialVersionUID = 1L;

   private static void createAndShowGUI() {

      // Create and set up the window.
      final JFrame frame = new JFrame("Scroll Pane Example");

      // Display the window.
      frame.setSize(500, 500);
      frame.setVisible(true);
      frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

      // set flow layout for the frame
      frame.getContentPane().setLayout(new FlowLayout());

      JTextArea textArea = new JTextArea(20, 20);
      JScrollPane scrollableTextArea = new JScrollPane(textArea);

      scrollableTextArea.setHorizontalScrollBarPolicy(JScrollPane.HORIZONTAL_SCROLLBAR_ALWAYS);
      scrollableTextArea.setVerticalScrollBarPolicy(JScrollPane.VERTICAL_SCROLLBAR_ALWAYS);

      frame.getContentPane().add(scrollableTextArea);
   }
   public static void main(String[] args) {


      javax.swing.SwingUtilities.invokeLater(new Runnable() {

         public void run() {
            createAndShowGUI();
         }
      });
   }
}
```

Output:

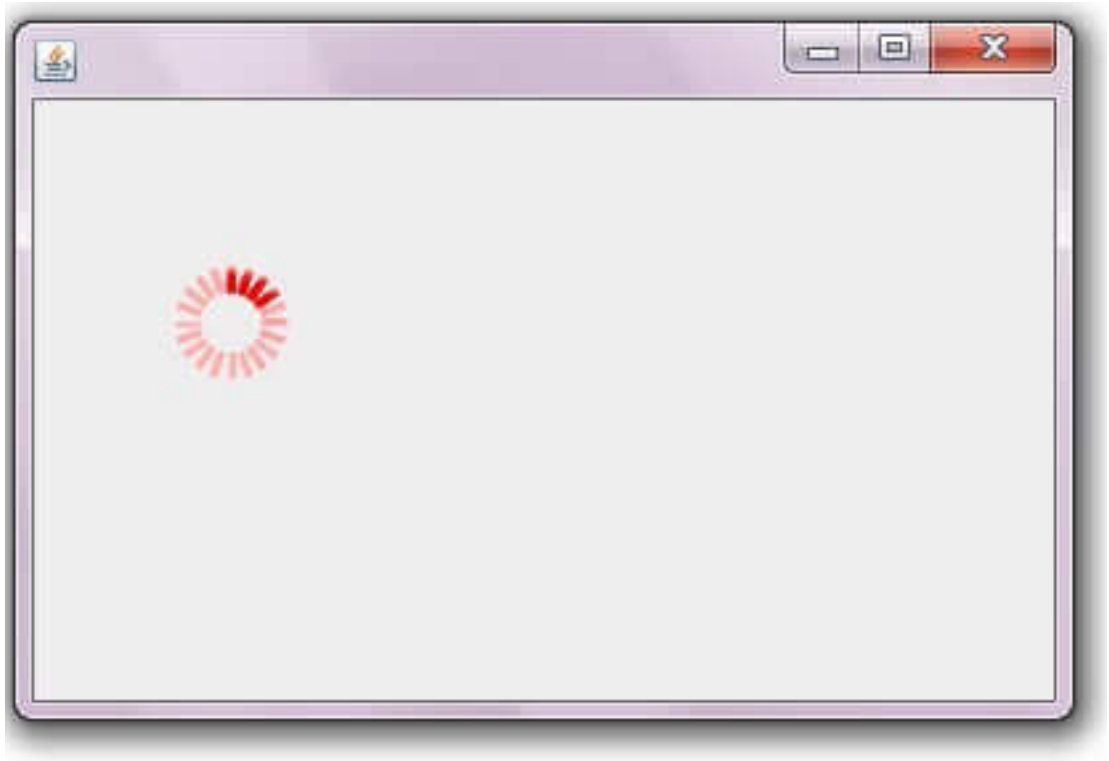


Displaying image in swing:

- For displaying image, we can use the method drawImage() of Graphics class.
- Syntax of drawImage() method:
- public abstract boolean drawImage(Image img, int x, int y, ImageObserver observer): is used draw the specified image.

Example:

```
import java.awt.*;
import javax.swing.JFrame;

public class MyCanvas extends Canvas{

    public void paint(Graphics g) {

        Toolkit t=Toolkit.getDefaultToolkit();
        Image i=t.getImage("p3.gif");
        g.drawImage(i, 120,100,this);

    }
        public static void main(String[] args) {
        MyCanvas m=new MyCanvas();
        JFrame f=new JFrame();
        f.add(m);
        f.setSize(400,400);
        f.setVisible(true);
    }

}
```



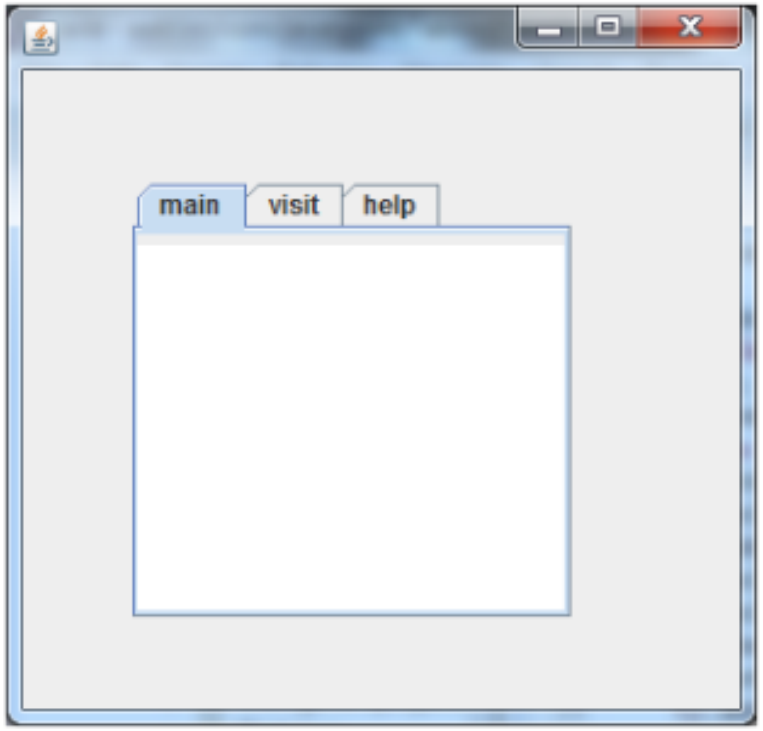
Java JTabbedPane

- The JTabbedPane class is used to switch between a group of components by clicking on a tab with a given title or icon. It inherits JComponent class.
- JTabbedPane class declaration
- public class JTabbedPane extends JComponent implements Serializable, Accessible, SwingConstants
- Commonly used Constructors:

Constructor	Description
JTabbedPane()	Creates an empty TabbedPane with a default tab placement of JTabbedPane.Top.
JTabbedPane(int tabPlacement)	Creates an empty TabbedPane with a specified tab placement.
JTabbedPane(int tabPlacement, int tabLayoutPolicy)	Creates an empty TabbedPane with a specified tab placement and tab layout policy

Example:

```
import javax.swing.*;
public class TabbedPaneExample {
JFrame f;
TabbedPaneExample(){
    f=new JFrame();
    JTextArea ta=new JTextArea(200,200);
    JPanel p1=new JPanel();
    p1.add(ta);
    JPanel p2=new JPanel();
    JPanel p3=new JPanel();
    JTabbedPane tp=new JTabbedPane();
    tp.setBounds(50,50,200,200);
    tp.add("main",p1);
    tp.add("visit",p2);
    tp.add("help",p3);
    f.add(tp);
    f.setSize(400,400);
    f.setLayout(null);
    f.setVisible(true);
}
public static void main(String[] args) {
    new TabbedPaneExample();
}}
```



Java ActionListener Interface

- The Java ActionListener is notified whenever you click on the button or menu item. It is notified against ActionEvent. The ActionListener interface is found in java.awt.event package. It has only one method: actionPerformed().
- **public abstract void** actionPerformed(ActionEvent e);

How to write ActionListener

1) Implement the ActionListener interface in the class:

```
public class ActionListenerExample Implements ActionListener
```

2) Register the component with the Listener:

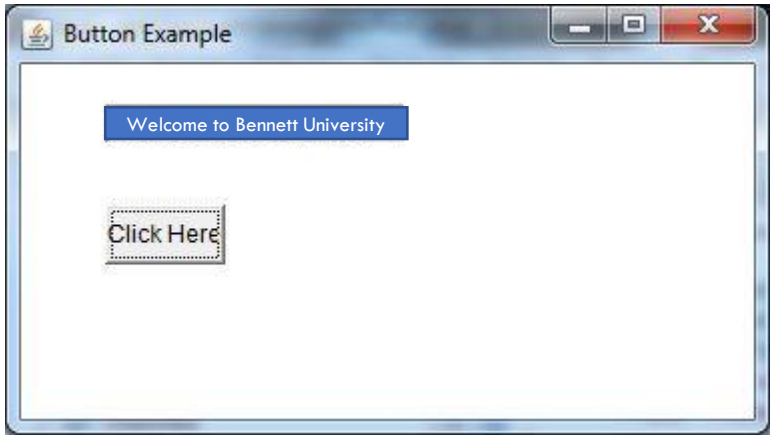
```
component.addActionListener(instanceOfListenerclass);
```

3) Override the actionPerformed() method:

```
public void actionPerformed(ActionEvent e){
        //Write the code here
}
```

Example:

```
1.import java.awt.*;
2.import java.awt.event.*;
3.//1st step
4.public class ActionListenerExample implements ActionListener{
5.public static void main(String[] args) {
6.    Frame f=new Frame("Button Example");
7.    final TextField tf=new TextField();
8.    tf.setBounds(50,50, 150,20);
9.    Button b=new Button("Click Here");
10.     b.setBounds(50,100,60,30);
11.     //2nd step
12.     b.addActionListener(this);
13.     f.add(b);f.add(tf);
14.     f.setSize(400,400);
15.     f.setLayout(null);
16.     f.setVisible(true);
17.}
18.//3rd step
19.public void actionPerformed(ActionEvent e){
20.             tf.setText("Welcome to Bennett University.");
21.}
22.}
```



Thankyou