Adding icon on JButton
==========
1. To change or set icon on JButton Java provides us a method called setIcon() belonging to JButton object.

2. The prototype of this method is :

public void setIcon(Icon);

3. The argument passed to the method is object of Icon but since Icon is an interface so we pass object of its derived class called ImageIcon.

4. Both Icon and ImageIcon come from the package java.awt.

Adding Tooltip to JButton
==========
1. Tooltip text is the short text which gets displayed when the user hovers mouse pointer on a JButton.

2. Java provides us a method called setToolTipText() belonging to JButton object for setting Tooltip text on a JButton.

3. The prototype of the method is:

**public void setToolTipText(String)**

**Code**
**=====**

```
public class MyFrame extends javax.swing.JFrame {
    public MyFrame() {
        super("Sachin's Frame");
        initComponents();
        setLocationRelativeTo(null);
        ImageIcon icon=new ImageIcon("D:/images/ico files/colorfill.png");
        btnChangeColor.setIcon(icon);
        btnChangeColor.setToolTipText("Click me for changing Color");

    }
```

Handling events on swing builder
==========
Normally when we perform event handling in Java we need to do 3 steps:

1. Implement the required **Event** interface. In our example it was **ActionListener**

2. Override the abstract method coming from the Event interface. In our case it was **actionPerformed()**.

3. Register event source and listener by calling registration method. In our case it was **addActionListener()**.

But when we use swing builder of Netbeans, as a programmer we just have to perform 2nd activity. The other 2 activities are automatically done by Netbeans. In simple words we simply have to override **actionPerformed()** and inheritance of **ActionListener** as well as registration of source and listener is automatically handled by Netbeans.

**Code**
**=====**

```
private void btnChangeColorActionPerformed(java.awt.event.ActionEvent evt) {
        myPanel.setBackground(Color.cyan);
    }

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

Adding JLabel of Panel
==========

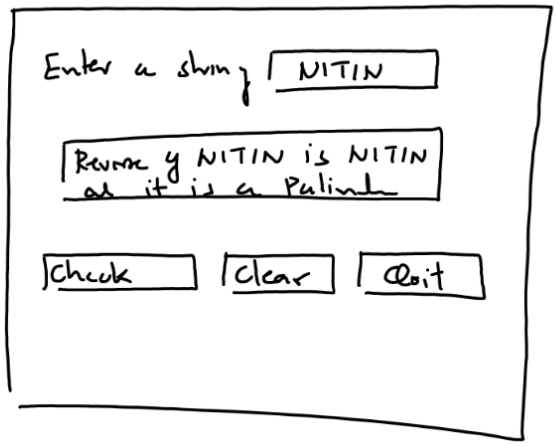
1. A JLabel component in GUI is used for displaying text or images on the user interface.

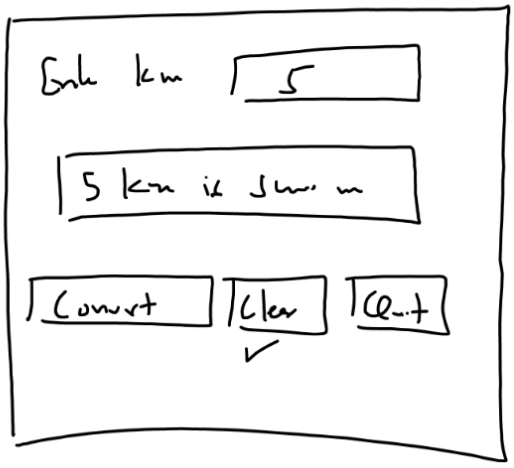
2. To add JLabel we can use the same drag and drop approach and set the properties of JLabel just like we use to set in JButton

CODE
=====

```
private void btnShowTimeStampActionPerformed(java.awt.event.ActionEvent evt) {
      Date today=new Date();
      lblShowTimeStamp.setText(today.toString());
   }
```

This is our JLabel which we added during design time






```
private void btnGreetActionPerformed(java.awt.event.ActionEvent evt) {
        String name=txtUserName.getText();
        lblGreetings.setText("Good Evening "+name);
    }
```