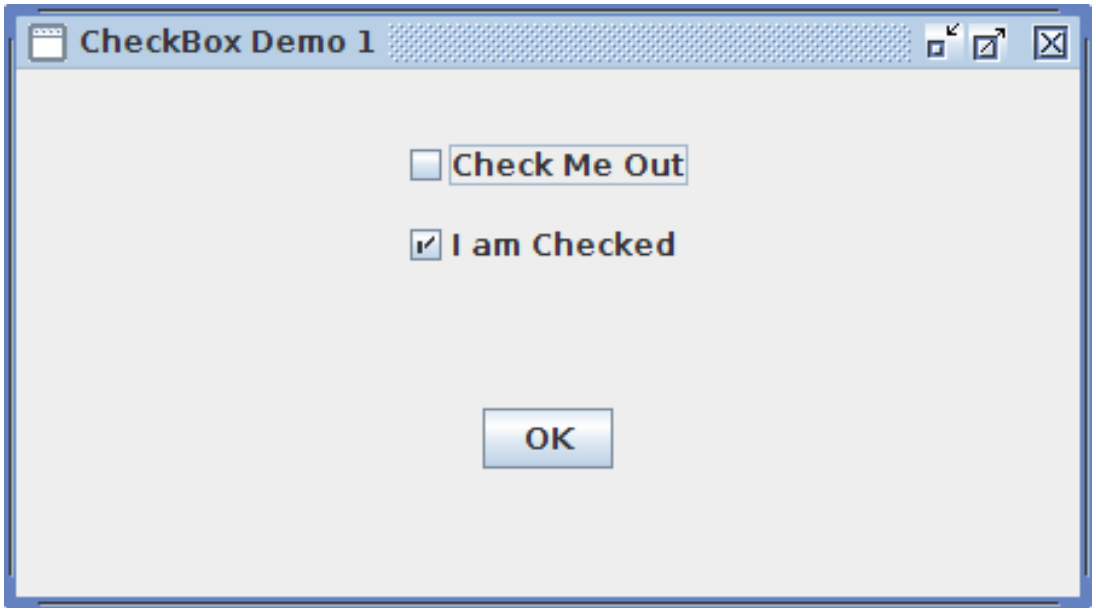
16. GUI Programming :: The *JCheckBox* Class

A GUI **check box** is a component with two states: **checked** or **not checked** (or **selected** or **unselected**). To create a check box we use the *javax.swing.JCheckBox* class.

```
JCheckBox cb1 = new JCheckBox("Check Me Out"); // Default is not checked.
JCheckBox cb2 = new JCheckBox("I am Checked", true);
```



To determine if a *JCheckBox* is checked or unchecked, we call the *isSelected*() method:

```
boolean b1 = cb1.isSelected();  // Would be false
boolean b2 = cb2.isSelected();  // Would be true
```

16. GUI Programming :: The *JCheckBox* Class :: *CheckBoxDemo1*

```
///*************************************************************************
// CLASS: CheckBoxDemo1 (CheckBoxDemo1.java)
//*************************************************************************
import java.awt.GridLayout;
import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import javax.swing.BoxLayout;
import javax.swing.JButton;
import javax.swing.JCheckBox;
import javax.swing.JFrame;
import javax.swing.JOptionPane;
import javax.swing.JPanel;

/**
 * This application demonstrates how to create GUI check boxes using the javax.
 * swing.JCheckBox class.
 */
public class CheckBoxDemo1 {
  public static void main(String[] args) { new CheckBoxDemo1().run(); }
  public void run() {
    JFrame.setDefaultLookAndFeelDecorated(true);

    final JCheckBox cb1 = new JCheckBox("Check Me Out", false);
    final JCheckBox cb2 = new JCheckBox("I am Checked", true);
```

16. GUI Programming :: The *JCheckBox* Class :: *CheckBoxDemo1*

```
// Add the two check boxes to a JPanel using the GridLayout with 2 rows
// and 1 column. Create a 10 pixel space between each row.
JPanel panelCheckGrid = new JPanel();
panelCheckGrid.setLayout(new GridLayout(2, 1, 0, 10));
panelCheckGrid.add(cb1);
panelCheckGrid.add(cb2);

// Create a JPanel which uses the default FlowLayout and add panelCheckGrid
// to the JPanel. This is done so the contents of panelCheckGrid will be
// centered in the frame.
JPanel panelCheck = new JPanel();
panelCheck.add(panelCheckGrid);

// Create a JButton and an event handler for the button using an anonymous
// class. When the button is clicked, we determine if each check box is
// selected or unselected calling the isSelected() method. Display the
// results in a JOptionPane dialog.
JButton butOk = new JButton("OK");
butOk.addActionListener(
  new ActionListener() {
    @Override public void actionPerformed(ActionEvent pEvent) {
      String msg = "cb1 is ";
      if (cb1.isSelected()) msg += "checked\n";   else msg += "not checked.\n";
      msg += "cb2 is ";
      if (cb2.isSelected()) msg += "checked\n";   else msg += "not checked.\n";
      JOptionPane.showMessageDialog(null, msg);
    }
  });
```

16. GUI Programming :: The *JCheckBox* Class :: *CheckBoxDemo1*

```
        // Creat a JPanel for the button. Use the default FlowLayout so the button
        // will be centered in the frame.
        JPanel panelButton = new JPanel();
        panelButton.add(butOk);

        // Create a main panel to hold the panelCheck and panelButton panels. Use
        // a vertical BoxLayout. Put vertical glue above, below, and in between each
        // panel.
        JPanel mainPanel = new JPanel();
        mainPanel.setLayout(new BoxLayout(mainPanel, BoxLayout.Y_AXIS));
        mainPanel.add(javax.swing.Box.createVerticalGlue());
        mainPanel.add(panelCheck);
        mainPanel.add(javax.swing.Box.createVerticalGlue());
        mainPanel.add(panelButton);
        mainPanel.add(javax.swing.Box.createVerticalGlue());

        // Create the JFrame.
        JFrame frame = new JFrame("CheckBox Demo 1");
        frame.setSize(450, 250);
        frame.add(mainPanel);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.setVisible(true);
    }
}
```

16. GUI Programming :: The *JCheckBox* Class :: *CheckBoxDemo1*