
BUTTONS AND TOOLBARS

Swing provides separate implementations of both the `JRadioButton` and the `JCheckBox`. A checkbox has two states and within a group of checkboxes, any number can be selected or deselected. Radio buttons should be grouped into a `ButtonGroup` object so that only one radio button of a group can be selected at a time.

Radio Buttons

Both radio buttons and check boxes can be instantiated with an image as well as a title and both can have rollover icons. The `JCheckBox` component is derived from the simpler `JToggleButton` object. `JToggleButton` is a button that can be switched between two states by clicking, but which stays in that new state (up or down) like a 2-state check box does. Further the `JToggleButton` can take on the exclusive aspects of a radio button by adding it to a `ButtonGroup`.

```
//create radio buttons in right panel
JRadioButton Rep, Dem, Flat;

right.add(Rep = new JRadioButton("Republicrat"));
right.add(Dem = new JRadioButton("Demmican"));
right.add(Flat = new JRadioButton("Flat Earth"));
ButtonGroup bgroup = new ButtonGroup();
bgroup.add(Rep);           //add to button group
bgroup.add(Dem);
bgroup.add(Flat);
```

If you neglect to add the radio buttons to a `ButtonGroup`, you can have several of them turned on at once. It is the `ButtonGroup` that assures that only one at a time can be turned on. The `ButtonGroup` object thus keeps track of the state of all the radio buttons in the group to enforce this only-one-on protocol. This is a clear example of the Mediator pattern we'll be discussing in the chapters ahead.

The JToolBar

`JToolBar` is a container bar for tool buttons of the type you see in many programs. Normally, the JDK documentation recommends that you add the `JToolBar` as the only component on one side of a `BorderLayout` typically the North side), and that you not add components to the other 3 sides. The

buttons you add to the toolbar are just small JButtons with picture icons and without text. The JToolBar class has two important methods: *add* and *addSeparator*.

```
JToolBar toolbar = new JToolBar();
JButton Open = new JButton("open.gif");
toolbar.add(Open);
toolbar.addSeparator();
```

By default, JButtons have a rectangular shape, and to make the usual square-looking buttons, you need to use square icons and set the Insets of the button to zero. On most toolbars, the icons are 25 x 25 pixels. We thus develop the simple ToolButton class below, which handles both the insets and the size:

```
public class ToolButton extends JButton
{
    public ToolButton(Icon img)
    {
        super(img);
        setMargin(new Insets(0,0,0,0));
        setSize(25,25);
    }
}
```

The JToolBar also has the characteristic that you can detach it from its anchored position along the top side of the program and attach it to another side, or leave it floating. This allows some user customization of the running program, but is otherwise not terribly useful. It also is not particularly well implemented and can be confusing to the user. Thus, we recommend that you use the *setFloatable(false)* method to turn this feature off.

Toggle Buttons

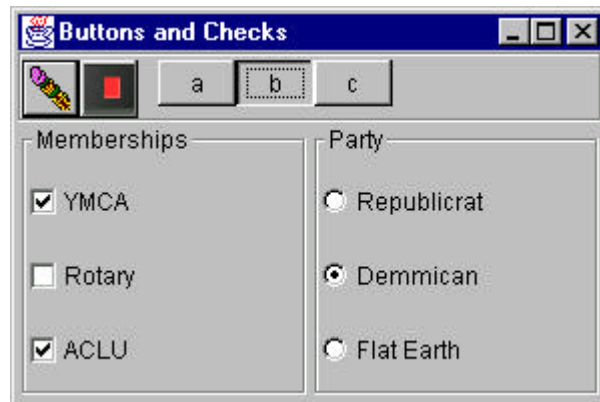
The JToggleButton class is actually the parent class for check boxes and radio buttons. It is a two-state button that will stay in an up or down position when clicked, and you can use it just like a check box. While toggle buttons look sort of strange on most screens, they look very reasonable as part of toolbars. You can use individual toggle buttons to indicate the state of actions the user might select. By themselves, toggle buttons behave like check boxes, so you can press as many as you want, and you can “uncheck” or raise toggle buttons by using the *setSelected(false)* method.

You can also add toggle buttons to a ButtonGroup so that they behave like radio buttons: only one at a time can be pressed down. However, once a ButtonGroup object is mediating them, you can’t raise the buttons

using the *setSelected* method. If you want to be able to raise them, but still only allow one at a time to be pressed, you need to write your own Medator class to replace the ButtonGroup object.

Sample Code

The simple program display below illustrates checkboxes, radio buttons, toolbar buttons, and toggle buttons:



Note the "b" JToggleButton is depressed permanently. While the user can select any number of organizations in which he holds memberships, he can only select one political party.