Supplement IV.G: OverlayLayout

For Introduction to Java Programming
Y. Daniel Liang

OverlayLayout is a Swing layout manager that arranges components on top of each other. To create an OverlayLayout, use the following constructor:

```
public OverlayLayout(Container target)
```

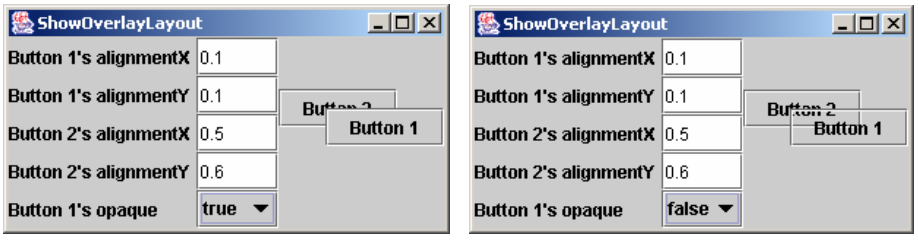
The constructor creates a layout manager that is dedicated to the given target container. For example, the following code creates an OverlayLayout for panel p1:

```
JPanel p1 = new JPanel();
OverlayLayout overlayLayout = new OverlayLayout(p1);
p1.setLayout(overlayLayout);
```

You still need to invoke the setLayout method on p1 to set the layout manager.

A component is on top of another component if it is added to the container before the other one. Suppose components p1, p2, and p3 are added to a container of the OverlayLayout in this order, then p1 is on top of p2, and p2 is on top of p3.

Listing 31.5 gives an example that overlays two buttons in a panel of OverlayLayout, as shown in Figure 31.10. The first button is on top of the second button. The program enables the user to set the alignmentX and alignmentY properties of the two buttons dynamically. You can also set the opaque (blocked) property of the first button. When the opaque property is set to true, the first button blocks the scene of the second button, as shown in Figure 31.10(a). When the opaque property is set to false, the first button becomes transparent to allow the second button to be seen through the first button, as shown in Figure 31.10(b).



(a) (b)

Figure 31.10

The components are overlaid in the container of OverlayLayout.

Listing 31.5 ShowOverLayLayout.java

******PD: Please add line numbers in the following code******
******Layout: Please layout exactly. Don't skip the space. This is true for all source code in the book. Thanks, AU.***
<Side Remark line 26: overlay layout>
<Side Remark line 90: main omitted>

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

public class ShowOverlayLayout extends JApplet {
  private JButton jbt1 = new JButton("Button 1");
  private JButton jbt2 = new JButton("Button 2");

  private JTextField jtfButton1AlignmentX = new JTextField(4);
  private JTextField jtfButton1AlignmentY = new JTextField(4);
  private JTextField jtfButton2AlignmentX = new JTextField(4);
  private JTextField jtfButton2AlignmentY = new JTextField(4);
  private JComboBox jcboButton1Opaque = new JComboBox(
    new Object[]{new Boolean(true), new Boolean(false)});

  // Panel p1 to hold two buttons
  private JPanel p1 = new JPanel();

  public ShowOverlayLayout() {
    // Add two buttons to p1 of OverlayLayout
    p1.setLayout(new OverlayLayout(p1));
    p1.add(jbt1);
    p1.add(jbt2);

    JPanel p2 = new JPanel();
    p2.setLayout(new GridLayout(5, 1));
    p2.add(new JLabel("Button 1's alignmentX"));
    p2.add(new JLabel("Button 1's alignmentY"));
    p2.add(new JLabel("Button 2's alignmentX"));
    p2.add(new JLabel("Button 2's alignmentY"));
    p2.add(new JLabel("Button 1's opaque"));

    JPanel p3 = new JPanel();
    p3.setLayout(new GridLayout(5, 1));
    p3.add(jtfButton1AlignmentX);
    p3.add(jtfButton1AlignmentY);
    p3.add(jtfButton2AlignmentX);
    p3.add(jtfButton2AlignmentY);
    p3.add(jcboButton1Opaque);

    JPanel p4 = new JPanel();
    p4.setLayout(new BorderLayout(4, 4));
    p4.add(p2, BorderLayout.WEST);
    p4.add(p3, BorderLayout.CENTER);

    add(p1, BorderLayout.CENTER);
    add(p4, BorderLayout.WEST);

    jtfButton1AlignmentX.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
```

```
        jbt1.setAlignmentX(
          Float.parseFloat(jtfButton1AlignmentX.getText()));
        p1.revalidate(); // Cause the components to be rearranged
        p1.repaint(); // Cause the viewing area to be repainted
      }
    });
    jtfButton1AlignmentY.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jbt1.setAlignmentY(
          Float.parseFloat(jtfButton1AlignmentY.getText()));
        p1.revalidate(); // Cause the components to be rearranged
        p1.repaint(); // Cause the viewing area to be repainted
      }
    });
    jtfButton2AlignmentX.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jbt2.setAlignmentX(
          Float.parseFloat(jtfButton2AlignmentX.getText()));
        p1.revalidate(); // Cause the components to be rearranged
        p1.repaint(); // Cause the viewing area to be repainted
      }
    });
    jtfButton2AlignmentY.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jbt2.setAlignmentY(
          Float.parseFloat(jtfButton2AlignmentY.getText()));
        p1.revalidate(); // Cause the components to be rearranged
        p1.repaint(); // Cause the viewing area to be repainted
      }
    });
    jcboButton1Opaque.addActionListener(new ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jbt1.setOpaque(((Boolean)(jcboButton1Opaque.
          getSelectedItem())).booleanValue());
        p1.revalidate(); // Cause the components to be rearranged
        p1.repaint(); // Cause the viewing area to be repainted
      }
    });
  }
       }
```

A panel p1 of OverlayLayout is created (line 21) to hold two buttons (lines 22-23). Since Button 1 is added before Button 2, Button 1 is on top of Button 2.

The alignmentX and alignmentY properties specify how the two buttons are aligned relative to each other along the x-axis and y-axis (lines 51, 59). These two properties are used in BoxLayout and OverlayLayout, but are ignored by other layout managers. Note that the alignment is a float type number between 0 and 1.

The opaque property is defined in JComponent for all Swing lightweight components. By default, it is true for JButton, which means that the button is nontransparent. So if Button 1's opaque is true, you cannot see any other components behind JButton 1. To enable the components behind Button 1 to be seen, set Button 1's opaque property to false (lines

83-86).

.