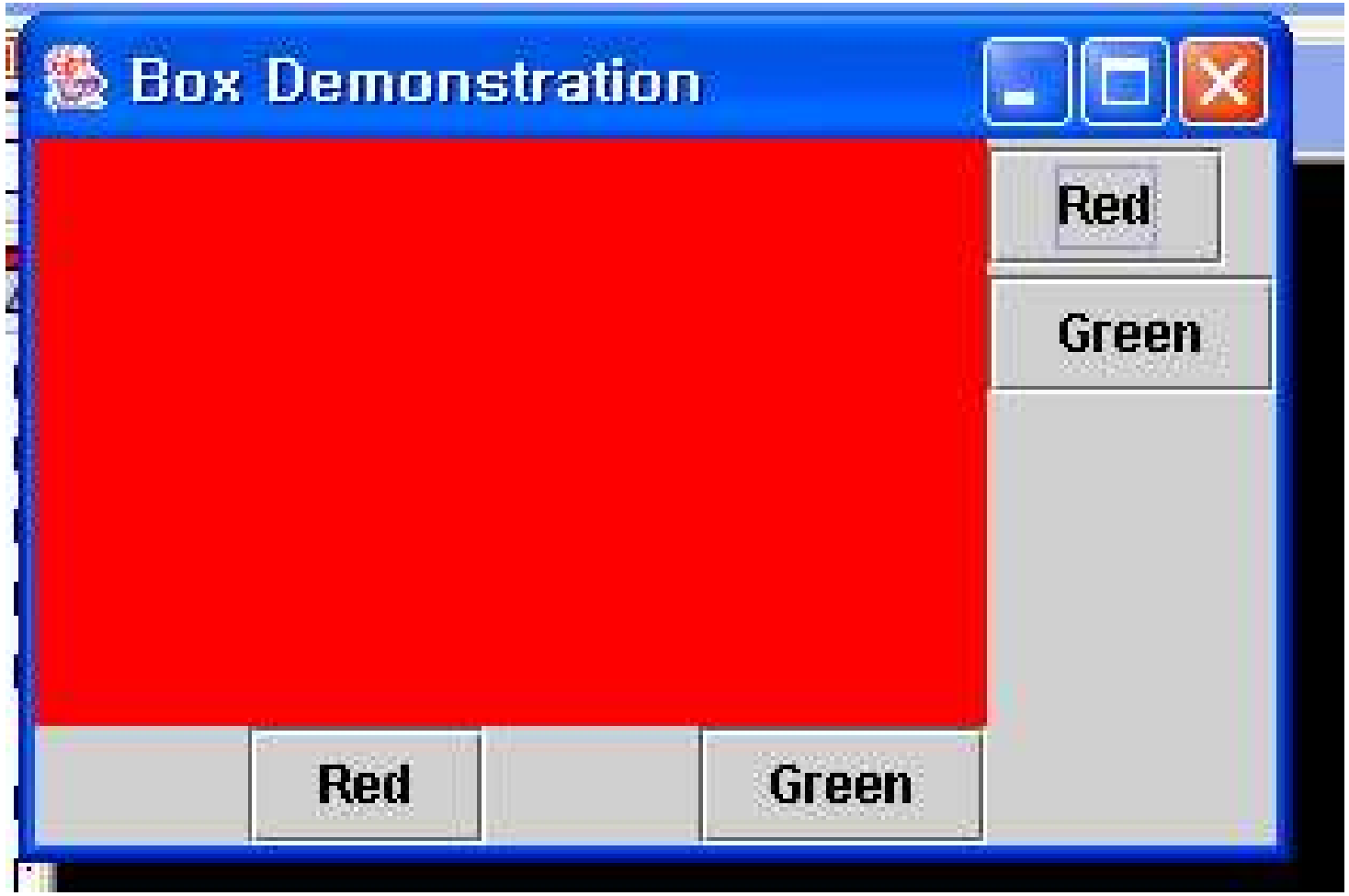
15.3 More Layout Manager

- Box Layout Manager
- Card Layout Manager

Box Layout Manager



Box Layout Manager

- Useful for a single column or single row of components
- Specify `X_AXIS` (horizontal) or `Y_AXIS` (vertical) layout as second parameter to constructor for layout manager
- Provides a means of separating components in a row or column
  - » —allocates a fixed amount of space between two components
  - » —allocates a variable amount of space between two components
- A container is a container that is automatically given a `BoxLayout` manager.

Box Layout Versus Other Layouts

- Horizontal box layout is similar to flow layout.
- Vertical box layout is similar to grid layout with only one column.

- Big advantage of box layout is control over spacing using struts and glue.
- Note that it is possible to use struts and glue with other layout managers but they will probably not work as intended.

Box Layout Demo Program

```
JPanel horizontalPanel = new JPanel();
horizontalPanel.setLayout(
    new BoxLayout(horizontalPanel, BoxLayout.X_AXIS));
Component horizontalStrut =
    Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE);
horizontalPanel.add(horizontalStrut);
JButton hStopButton = new JButton("Red");
hStopButton.addActionListener(this);
horizontalPanel.add(hStopButton);
```

Specifies a horizontal layout

Static method in `Box` class used to create a strut of a particular size for spacing

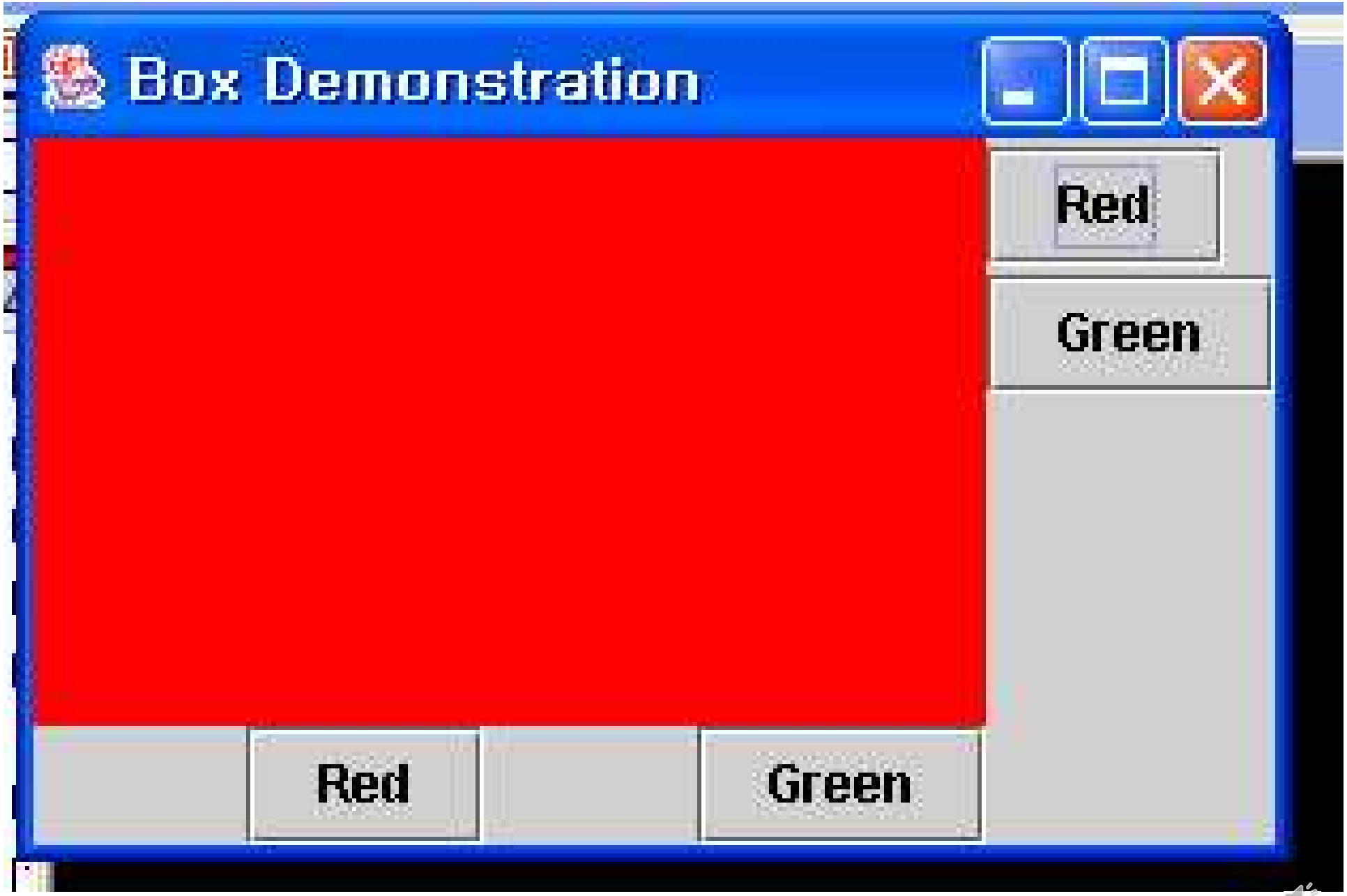
Listing 15.5 The BoxLayout Manager - BoxLayoutDemo.java

```
/**
 Simple demonstration of BoxLayout manager and the use of struts
 to separate components (in this case buttons). For an alternative
 implementation see BoxClassDemo in Display 14.11
*/
public class BoxLayoutDemo extends JFrame implements ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;
    //
    public static final int HORIZONTAL_STRUT_SIZE = 50;
    public static final int VERTICAL_STRUT_SIZE = 2;

    private JPanel colorPanel;
```

```
public BoxLayoutDemo( )
  {
      setSize(WIDTH, HEIGHT);
      addWindowListener(new WindowDestroyer( ));
      setTitle("Box Demonstration");
      Container content = getContentPane( );
      content.setLayout(new BorderLayout( ));

      colorPanel = new JPanel( );
      colorPanel.setBackground(Color.BLUE);
      content.add(colorPanel, BorderLayout.CENTER);

      //Horizontal buttons at bottom of frame:
      JPanel horizontalPanel = new JPanel( );
      horizontalPanel.setLayout(new BoxLayout(horizontalPanel,
BoxLayout.X_AXIS));
      // BoxLayout.X_AXIS : the layout is a horizontal panel.

      Component horizontalStrut =
          Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE); //50
      horizontalPanel.add(horizontalStrut);

      JButton hStopButton = new JButton("Red");
      hStopButton.addActionListener(this);
      horizontalPanel.add(hStopButton);
```

```
Component horizontalStrut2 =
    Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE); //50
horizontalPanel.add(horizontalStrut2);
JButton hGoButton = new JButton("Green");
hGoButton.addActionListener(this);
horizontalPanel.add(hGoButton);

content.add(horizontalPanel, BorderLayout.SOUTH);

//Vertical buttons on right side of frame:
 JPanel verticalPanel = new JPanel( );
        verticalPanel.setLayout(new
BoxLayout(verticalPanel,BoxLayout.Y_AXIS));

Component verticalStrut =
Box.createVerticalStrut(VERTICAL_STRUT_SIZE);//2
verticalPanel.add(verticalStrut);

JButton vStopButton = new JButton("Red");
vStopButton.addActionListener(this);
verticalPanel.add(vStopButton);

Component verticalStrut2 =
Box.createVerticalStrut(VERTICAL_STRUT_SIZE); //2
verticalPanel.add(verticalStrut2);
```

```
JButton vGoButton = new JButton("Green");
        vGoButton.addActionListener(this);
        verticalPanel.add(vGoButton);

        content.add(verticalPanel, BorderLayout.EAST);
    }

 public void actionPerformed(ActionEvent e)
    {
       if (e.getActionCommand( ).equals("Red"))
           colorPanel.setBackground(Color.RED);
        else if (e.getActionCommand( ).equals("Green"))
           colorPanel.setBackground(Color.GREEN);
        else
           System.out.println("Error in button interface.");
    }

    public static void main(String[] args)
    {
        BoxClassDemo gui = new BoxClassDemo( );
        gui.setVisible(true);
    }
}
```


Box Demonstration
Red
Green
Red
Green

Struts and Glue

- invisible components used to add space between visible components
- ***horizontal strut***:
  - » programmer specifies width, which layout manager does not change
- ***vertical strut***:
  - » programmer specifies height, which layout manager does not change
- ***glue***:
  - » no [illegible] size
  - » can be added to layout to specify where extra space should go when container grows

Demo
GlueDemo.java

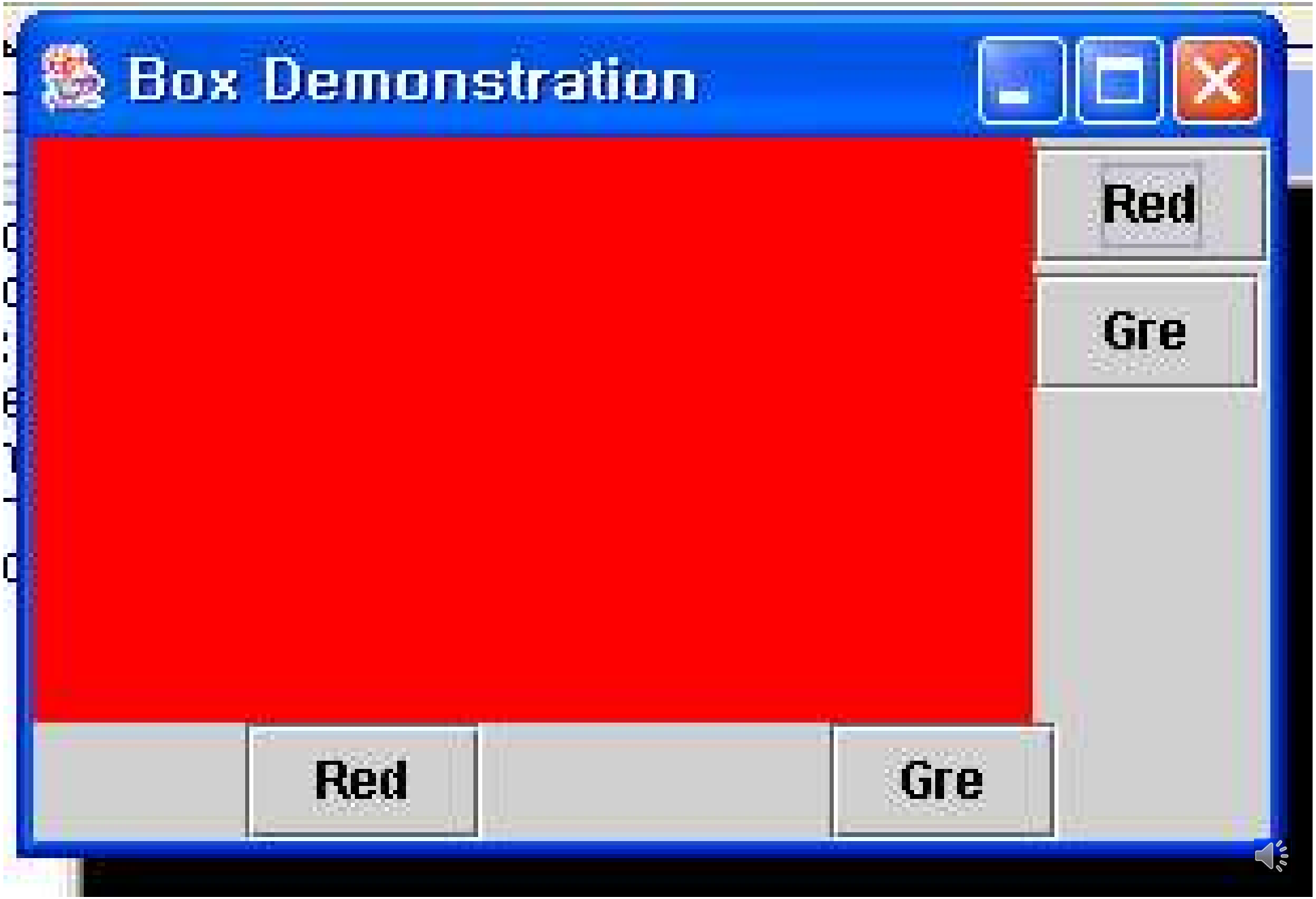
```
//Demo

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

/**
 Simple demonstration of BoxLayout manager class and the use of
 glue to separate components (in this case buttons).
*/
public class GlueDemo extends JFrame
                        implements ActionListener
{
   public static final int WIDTH = 300;
   public static final int HEIGHT = 200;
   public static final int HORIZONTAL_STRUT_SIZE = 50;
   public static final int VERTICAL_STRUT_SIZE = 2;

   private JPanel colorPanel;
```

```
public GlueDemo()
  {
      setSize(WIDTH, HEIGHT);
      addWindowListener(new WindowDestroyer());
      setTitle("Box Demonstration");
      Container content = getContentPane();
      content.setLayout(new BorderLayout());

      colorPanel = new JPanel();
      colorPanel.setBackground(Color.BLUE);
      content.add(colorPanel, BorderLayout.CENTER);

      //Horizontal buttons at bottom of frame:
      JPanel horizontalPanel = new JPanel();
      horizontalPanel.setLayout(
          new BoxLayout(horizontalPanel, BoxLayout.X_AXIS));

      Component horizontalStrut =
Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE);
      horizontalPanel.add(horizontalStrut);

      JButton hStopButton = new JButton("Red");
      hStopButton.addActionListener(this);
      horizontalPanel.add(hStopButton);
```

```
////
        Component horizontalGlue = Box.createHorizontalGlue();
        horizontalPanel.add(horizontalGlue);

 JButton hGoButton = new JButton("Gre");
        hGoButton.addActionListener(this);
        horizontalPanel.add(hGoButton);

        Component horizontalStrut2 =
Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE);
        horizontalPanel.add(horizontalStrut2);

        content.add(horizontalPanel, BorderLayout.SOUTH);
```

```
    //Vertical buttons on right side of frame:
    JPanel verticalPanel = new JPanel();
    verticalPanel.setLayout(
        new BoxLayout(verticalPanel, BoxLayout.Y_AXIS));

    Component verticalStrut =
        Box.createVerticalStrut(VERTICAL_STRUT_SIZE);//2
    verticalPanel.add(verticalStrut);

    JButton vStopButton = new JButton("Red");
    vStopButton.addActionListener(this);
    verticalPanel.add(vStopButton);

    Component verticalStrut2 =
        Box.createVerticalStrut(VERTICAL_STRUT_SIZE);//2
    verticalPanel.add(verticalStrut2);

   JButton vGoButton = new JButton("Gre");
    vGoButton.addActionListener(this);
    verticalPanel.add(vGoButton);

    content.add(verticalPanel, BorderLayout.EAST);
}
```

```
    public void actionPerformed(ActionEvent e)
    {
      if (e.getActionCommand().equals("Red"))
          colorPanel.setBackground(Color.RED);
       else if (e.getActionCommand().equals("Gre"))
         colorPanel.setBackground(Color.GREEN);
       else
          System.out.println("Error in button interface.");
    }

    public static void main(String[] args)
    {
       GlueDemo gui = new GlueDemo();
       gui.setVisible(true);
    }
}
```


Box Demonstration
Red
Gre
Red
Gre

Setting the Spacing Between Components

- To control spacing with layouts other than the `Box` layout, use `setHgap` and `setVgap`:

```
public void setHgap(int hgap)
```

  - » sets the horizontal gap between components
  - » argument is size of gap in pixels

```
public void setVgap(int vgap)
```

  - » sets the vertical gap between components
  - » argument is size of gap in pixels

- Can also use `EmptyBorder` in any layout manager which will add space as part of a component

The **Box** Container Class

- A `Box` object works like a panel with a `BoxLayout` manager.
- Created using static methods:

```
Box horizontalBox = Box                    );
Box verticalBox = Box
```

- Automatically given a `BoxLayout` manager when created
  - » You should not use `setLayout` method with a box object.

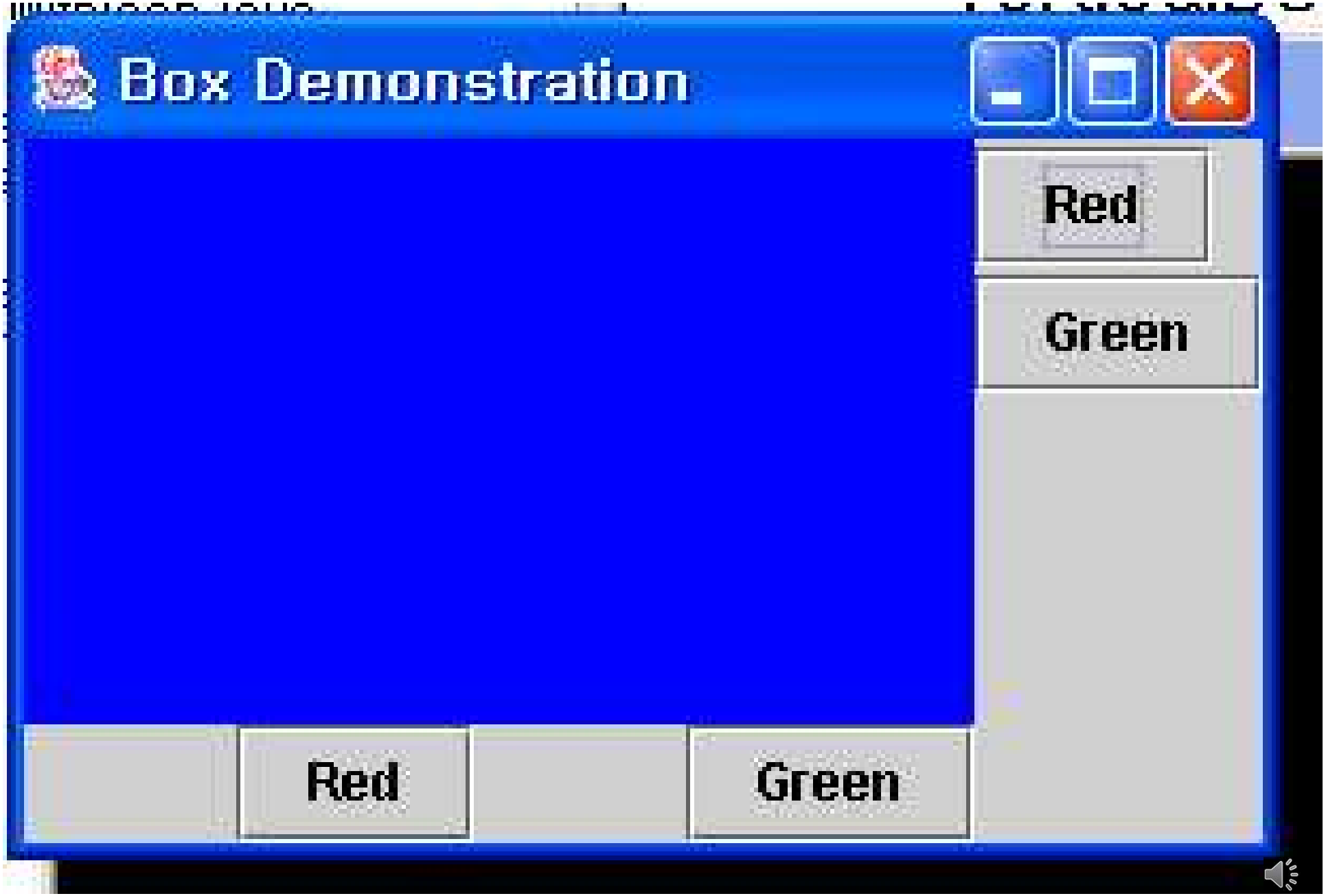
Listing 15.6 The Box Container Class - BoxClassDemo.java

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

/**
 Simple demonstration of Box container class and the use of struts
 to separate components (in this case buttons). For an alternative
 implementation see BoxLayoutDemo in Display 13.10.
*/
public class BoxClassDemo extends JFrame implements ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;
    //
    public static final int HORIZONTAL_STRUT_SIZE = 50;
    public static final int VERTICAL_STRUT_SIZE = 2;

    private JPanel colorPanel;
```

```
public BoxClassDemo( )
  {
      setSize(WIDTH, HEIGHT);
      addWindowListener(new WindowDestroyer( ));
      setTitle("Box Demonstration");
      Container content = getContentPane( );
      content.setLayout(new BorderLayout( ));

      colorPanel = new JPanel( );
      colorPanel.setBackground(Color.BLUE);
      content.add(colorPanel, BorderLayout.CENTER);

      //Horizontal buttons at bottom of frame:
      Box horizontalBox = Box.createHorizontalBox( );

      Component horizontalStrut =
          Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE);
      horizontalBox.add(horizontalStrut);

      JButton hStopButton = new JButton("Red");
      hStopButton.addActionListener(this);
      horizontalBox.add(hStopButton);

Component horizontalStrut2 =
          Box.createHorizontalStrut(HORIZONTAL_STRUT_SIZE);
      horizontalBox.add(horizontalStrut2);
```

```
JButton hGoButton = new JButton("Green");
        hGoButton.addActionListener(this);
        horizontalBox.add(hGoButton);

        content.add(horizontalBox, BorderLayout.SOUTH);

        //Vertical buttons on right side of frame:
        Box verticalBox = Box.createVerticalBox( );

        Component verticalStrut =
Box.createVerticalStrut(VERTICAL_STRUT_SIZE);
        verticalBox.add(verticalStrut);

        JButton vStopButton = new JButton("Red");
        vStopButton.addActionListener(this);
        verticalBox.add(vStopButton);

        Component verticalStrut2 =
Box.createVerticalStrut(VERTICAL_STRUT_SIZE);
        verticalBox.add(verticalStrut2);

        JButton vGoButton = new JButton("Green");
        vGoButton.addActionListener(this);
        verticalBox.add(vGoButton);

        content.add(verticalBox, BorderLayout.EAST);
    }
```

```
    public void actionPerformed(ActionEvent e)
    {
      if (e.getActionCommand( ).equals("Red"))
          colorPanel.setBackground(Color.RED);
       else if (e.getActionCommand( ).equals("Green"))
         colorPanel.setBackground(Color.GREEN);
       else
          System.out.println("Error in button interface.");
    }

    public static void main(String[] args)
    {
       BoxClassDemo gui = new BoxClassDemo( );
       gui.setVisible(true);
    }
}
```


Box Demonstration
Red
Green
Red
Green

The CardLayout Manager

- Allows a set of views (components) to choose among
  - » Only one view is
  - » Can go through views in order or jump to any view.
  - » Often the components added to a `CardLayout` will be panels.
- Each component added to a `CardLayout` has a string associated with it that works like a name:

```
deckPanel.add("start", startCardPanel);
```

  - » The string can be used later to display that component (or view):

```
dealer.show(deckPanel, "start");
```

- Need reference to layout manager to change views, so do not use anonymous object:

```
deckPanel.setLayout(new CardLayout());
```

legal but useless

horizontalPanel.setLayout(
new BoxLayout(horizontalPanel, BoxLayout.X_AXIS));

CardLayoutDemo

```
deckPanel = new JPanel();
dealer = new CardLayout();
deckPanel.setLayout(dealer);
...
deckPanel.add("start", startCardPanel);
...
deckPanel.add("green", greenCardPanel);
...
deckPanel.add("red", redCardPanel);
...
dealer.show(deckPanel, "red");
...
dealer.next(deckPanel);
dealer.first(deckPanel);
```

Only one of these three panels will be visible at a time.

will show `redCardPanel`

if `redCardPanel` is currently displayed, will show `startCardPanel`

The `CardLayout` Manager Class

- The first argument of method add names the component provided as the second argument.
- example

```
deckPanel.add("start", startCardPanel);
…
deckPanel.add("green", greenCardPanel);
…
deckPanel.add("red", redCardPanel);
```

The `CardLayout` Manager Class

- Two other methods, `first` and `next`, permit you to select a view.
- examples

```
dealer.first(deckPanel);
dealer.next(deckPanel);
```

- The container always starts with the first component on view.
- After the last component, `next` goes back to the first component.

Some Methods in the CardLayout Manager Class

- to display the first “card” in the container

```
public void first
    (Container theContainer);
```

- to display the last “card” in the container

```
public void last
    (Container theContainer);
```

- to display the next card

```
public void next
    (Container theContainer);
```

Some Methods in the CardLayout Manager Class, cont.

- to display the previous “card”

```
public void previous(Container theContainer);
```

- to display the “card” that was added with `cardName` as its name

```
public void show
     (Container, theContainer,
      String cardName);
```

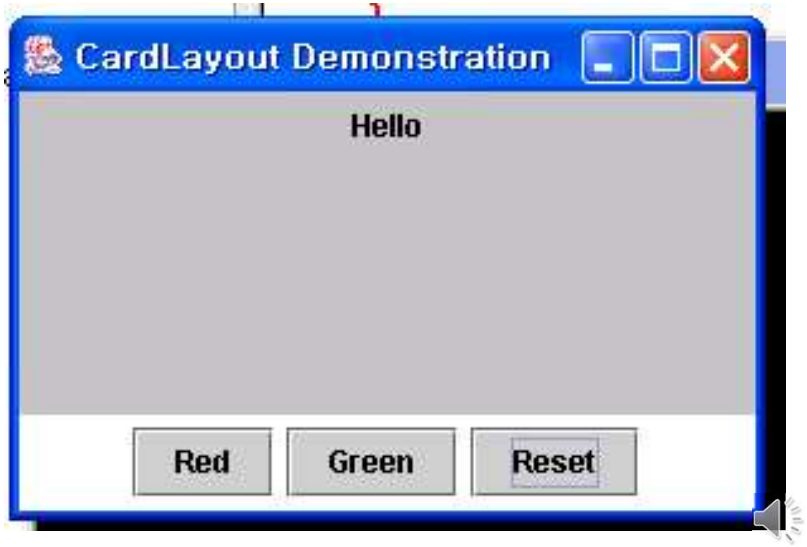
CardLayout Methods

- Figure 15.5 Some methods in the `CardLayout` manager class

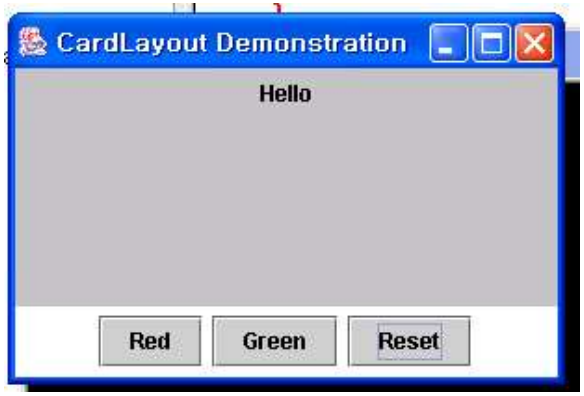
`public void first(Container theContainer)` Causes the first "card" in `theContainer` to be displayed.
`public void last(Container theContainer)` Causes the last "card" in `theContainer` to be displayed.
`public void next(Container theContainer)` Causes the next "card" in `theContainer` to be displayed. The next card is the one that was added after the currently displayed card was added. If the currently displayed card is the last card, the method displays the first card.
`public void previous(Container theContainer)` Causes the previous "card" in `theContainer` to be displayed. The previous card is the one that was added before the currently displayed card was added. If the currently displayed card is the first card, the method displays the last card.
`public void show(Container theContainer, String cardName)` Displays the "card" that was identified as `cardName` when it was added.

Listing 15.7 The CardLayout Manager - CardLayoutDemo.java

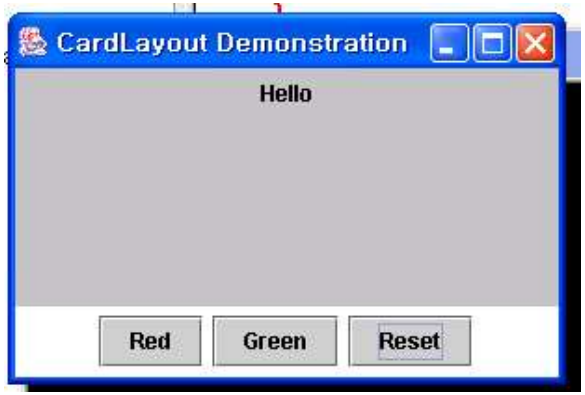
```
//Listing 15.7 The CardLayout Manager

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

public class CardLayoutDemo extends JFrame implements
ActionListener
{
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;

    private CardLayout dealer;
    private JPanel deckPanel;
```



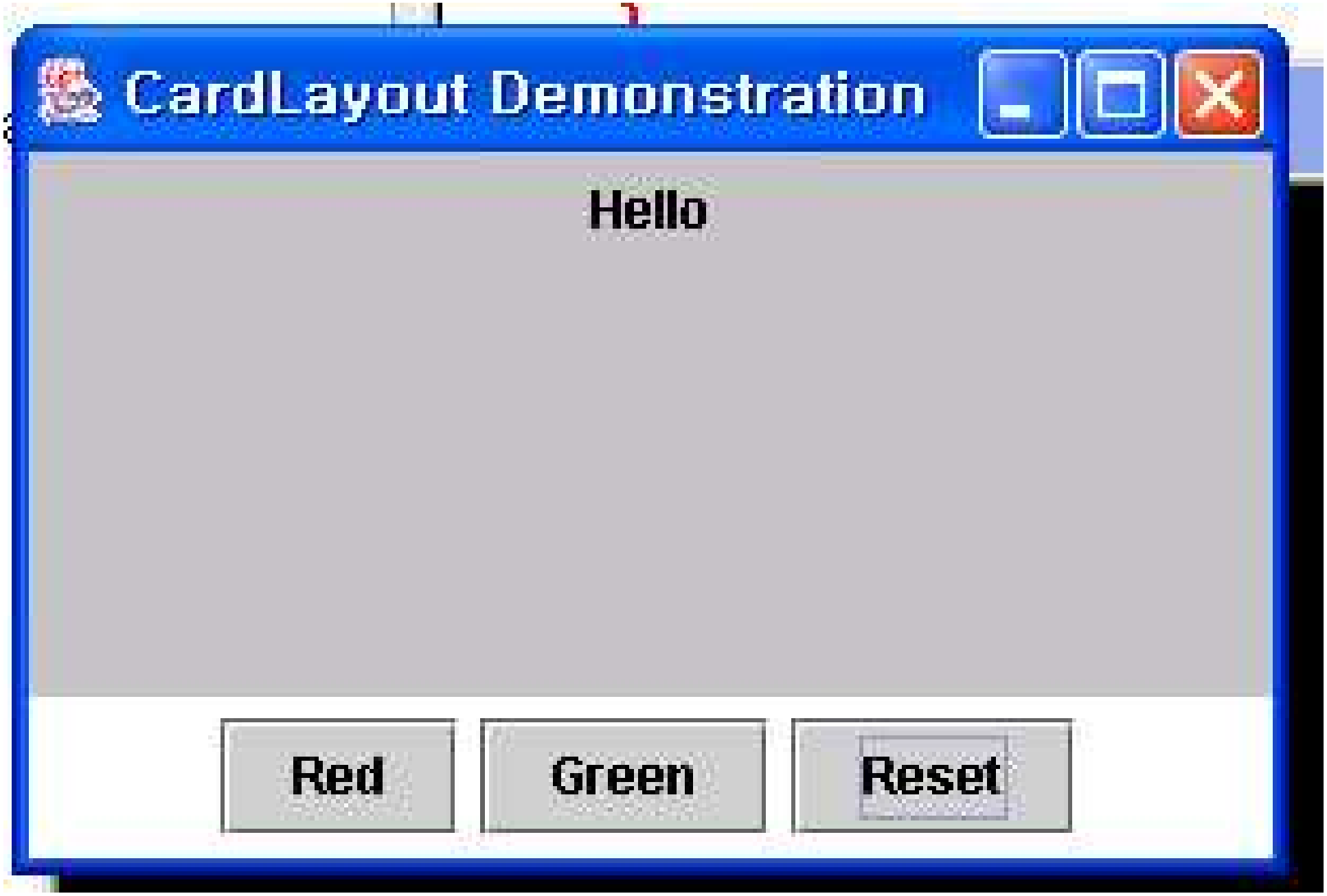
```
public CardLayoutDemo( )
  {
      setSize(WIDTH, HEIGHT);
      addWindowListener(new WindowDestroyer( ));
      setTitle("CardLayout Demonstration");
      Container contentPane = getContentPane( );
      contentPane.setLayout(new BorderLayout( ));

      deckPanel = new JPanel( );
      dealer = new CardLayout( );
      deckPanel.setLayout(dealer);

      JPanel startCardPanel = new JPanel( );
      startCardPanel.setLayout(new FlowLayout( ));
      startCardPanel.setBackground(Color.LIGHT_GRAY);
      JLabel startLabel = new JLabel("Hello");
      startCardPanel.add(startLabel);
      deckPanel.add("start", startCardPanel);

      JPanel greenCardPanel = new JPanel( );
      greenCardPanel.setLayout(new FlowLayout( ));
      greenCardPanel.setBackground(Color.GREEN);
      JLabel goLabel = new JLabel("Go");
      greenCardPanel.add(goLabel);
      deckPanel.add("green", greenCardPanel);
```



```
JPanel redCardPanel = new JPanel( );
    redCardPanel.setLayout(new FlowLayout( ));
    redCardPanel.setBackground(Color.RED);
    JLabel stopLabel = new JLabel("Stop");
    redCardPanel.add(stopLabel);
    deckPanel.add("red", redCardPanel);

    contentPane.add(deckPanel, BorderLayout.CENTER);

    JPanel buttonPanel = new JPanel( );
    buttonPanel.setBackground(Color.WHITE);
    buttonPanel.setLayout(new FlowLayout( ));
    JButton stopButton = new JButton("Red");
    stopButton.addActionListener(this);
    buttonPanel.add(stopButton);
    JButton goButton = new JButton("Green");
    goButton.addActionListener(this);
    buttonPanel.add(goButton);
    JButton resetButton = new JButton("Reset");
    resetButton.addActionListener(this);
    buttonPanel.add(resetButton);
    contentPane.add(buttonPanel, BorderLayout.SOUTH);
    dealer.first(deckPanel);//Optional
}
```



```
public void actionPerformed(ActionEvent e)
  {
     String actionCommand = e.getActionCommand( );

     if (actionCommand.equals("Red"))
        dealer.show(deckPanel, "red");
     else if (actionCommand.equals("Green"))
        dealer.show(deckPanel, "green");
     else if (actionCommand.equals("Reset"))
        dealer.show(deckPanel, "start");
     else
        System.out.println("Error in CardLayout Demo.");
   }


   public static void main(String[] args)
   {
      CardLayoutDemo demoGui = new CardLayoutDemo( );
      demoGui.setVisible(true);
   }
}
```


CardLayout Demonstration
Hello
Red
Green
Reset

끝