GUI Programming

- Readings: Savitch, Chapter 12

Components, Containers, and Layout Managers in AWT

- A graphical user interface normally contains a window (or a series of windows) with different components (such as buttons, text fields, menus etc) in it.
- The window is normally built by inheriting from an existing container class.

- The class Container is defined in the API. All its descendent classes (such as Window, Frame, Panel) are called container classes. Each container class has a method called add(). We can use add() to add almost any AWT objects (such as menus, buttons and text fields) to the container class.

Frame, Button and LayoutManager

- The following example demonstrates how to build up a window, add items into the window, and organise the items.

Example

```
// ButtonDemo.java
import java.awt.*;
import java.awt.event.*;

/* Display the colour chosen by the user
   (Adapted from Savitch) */
public class ButtonDemo extends Frame implements
    ActionListener {
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;
    public static final int X = 20;
    public static final int Y = 20;

    private String theText = “Press a button …";
```

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

public ButtonDemo() {
        setSize(WIDTH, HEIGHT);
        setLocation(X, Y);
        addWindowListener(new WindowDestroyer());
        setTitle("Button Demonstration");
        setBackground(Color.blue);

        setLayout(new FlowLayout());
```

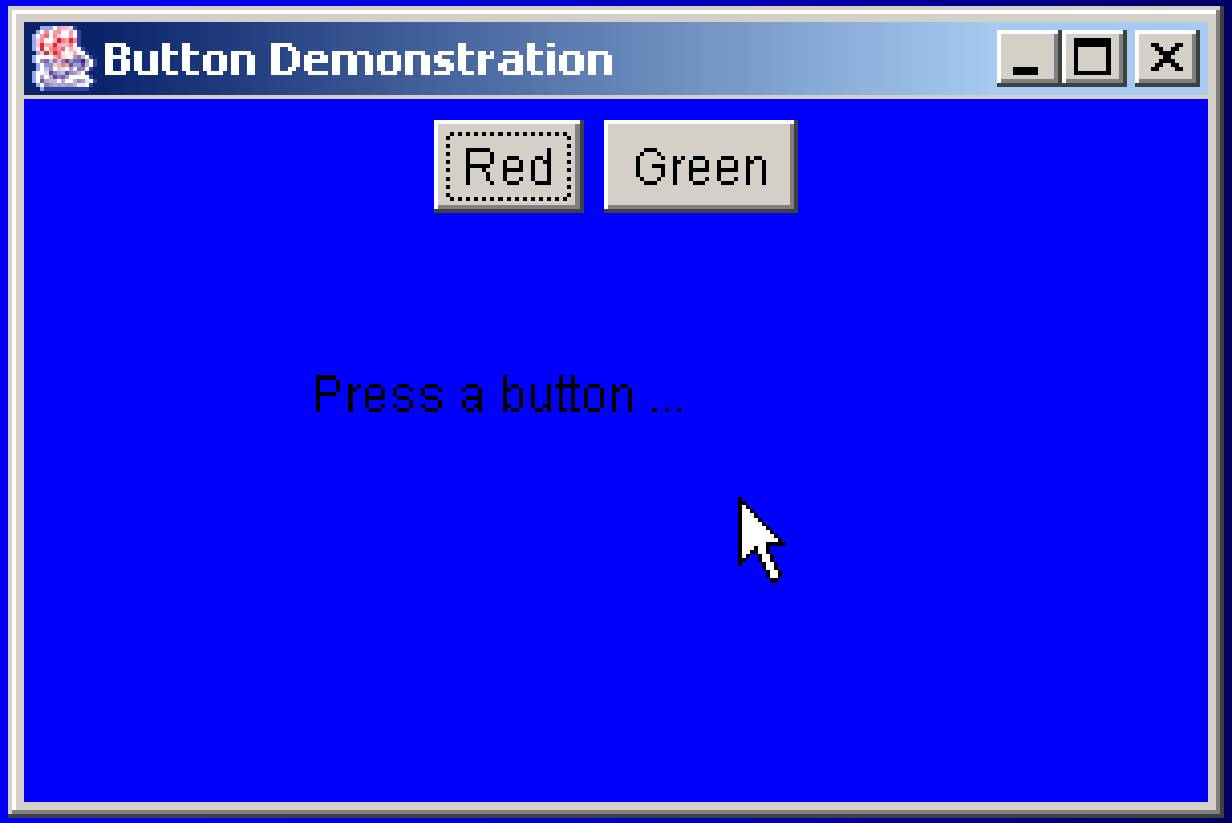
```
Button stopButton = new Button("Red");
    stopButton.addActionListener(this);
    add(stopButton);

    Button goButton = new Button("Green");
    goButton.addActionListener(this);
    add(goButton);
}

public void paint(Graphics g) {
    g.drawString(theText, 75, 100);
}
```

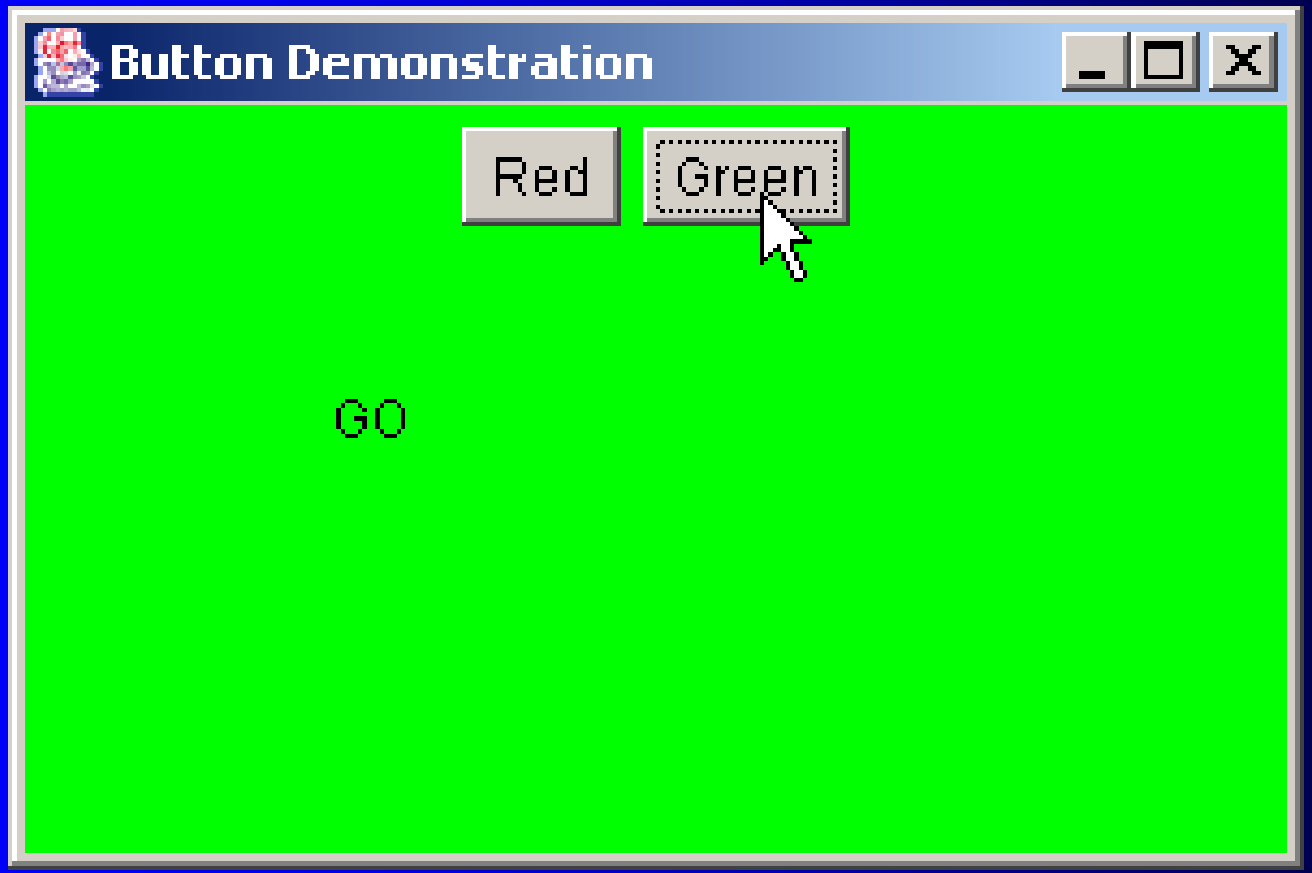
```
public void actionPerformed(ActionEvent e) {
        if (e.getActionCommand().equals("Red")) {
            setBackground(Color.red);
            theText = "STOP";
        }
        else if (e.getActionCommand().equals("Green")) {
           setBackground(Color.green);
           theText = "GO";
        }
        else
            theText = "Error in button interface.";
        repaint(); //force color and text change
    }
}
```

```
// WindowDestroyer.java
import java.awt.*;
import java.awt.event.*;
/* End the program and close the Frame if the user click
    the close window button. */
public class WindowDestroyer extends WindowAdapter {
    public void windowClosing(WindowEvent e) {
        System.exit(0);
    }
}
```

Program Execution

Java ButtonDemo




Button Demonstration
Red
Green
GO

Some Notes

- Frame is a subclass of Window in API.
  Normally, a programmer extends from Frame (rather than Window) to build his/her own windows. A Frame object contains a border, title and a number of buttons.

- Frame sizes are defined in pixels, and start at the top left corner which is (0, 0).
- Below is a list of methods associated with Frame.

AddWindowListener()
setSize()
setLocation()
setVisible()
setTitle()
setForground()
setBackgroup()

- The setLocation() function takes two coordinates and uses these to set the location of an interface window, on the screen.
- The two parameters are integers and represent pixel coordinates.
- If this function is not used, default (0, 0) positioning applies. This leads to the GUI title bar being inconveniently placed in the very top left hand corner of the screen.

- Using setLocation with values of 20 or greater will ensure that the GUI title bar is clear of the top of the screen.

```
setLocation(30, 40);
```

- The Color class (in API) is defined to handle colours for containers, components etc.

  brighter()
  darker()
  equals()
  getColor()

  are some frequently used methods.

  The class defines a number of colours (such as red, black, white, blue and grey etc) as static variables. Therefore we can use

  Color.red

- setLayout() adds a layout manager into the frame. The interface LayoutManager is defined in API. It has a number of subclasses (such as FlowLayout, BorderLayout, BoxLayout etc) which defines different layout for the components being added into the frame.

- The Button class (in API) can be used to define Button objects. Each object has a label and is linked with a listener. Once clicked, the listener triggers an action. In the example,

```
Button stopButton = new Button("Red");
stopButton.addActionListener(this);
add(stopButton);
```

Panel

- The following example displays how to add a panel into the frame, and add items onto the panel.

Example

```
//PanelDemo.java
import java.awt.*;
import java.awt.event.*;

/* put buttons in a panel (Adapted from Savitch) */
public class PanelDemo extends Frame implements
    ActionListener {
    public static final int WIDTH = 300;
    public static final int HEIGHT = 200;
    public static final int X = 20;
    public static final int Y = 20;

    private String theText = “Press a button …";
```

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

    public PanelDemo() {
        setTitle("Panel Demonstration");
        setSize(WIDTH, HEIGHT);
        setLocation(X,Y);
        setBackground(Color.blue);
        addWindowListener(new WindowDestroyer());
        Panel buttonPanel = new Panel();
        buttonPanel.setBackground(Color.white);
        buttonPanel.setLayout(new FlowLayout());
```

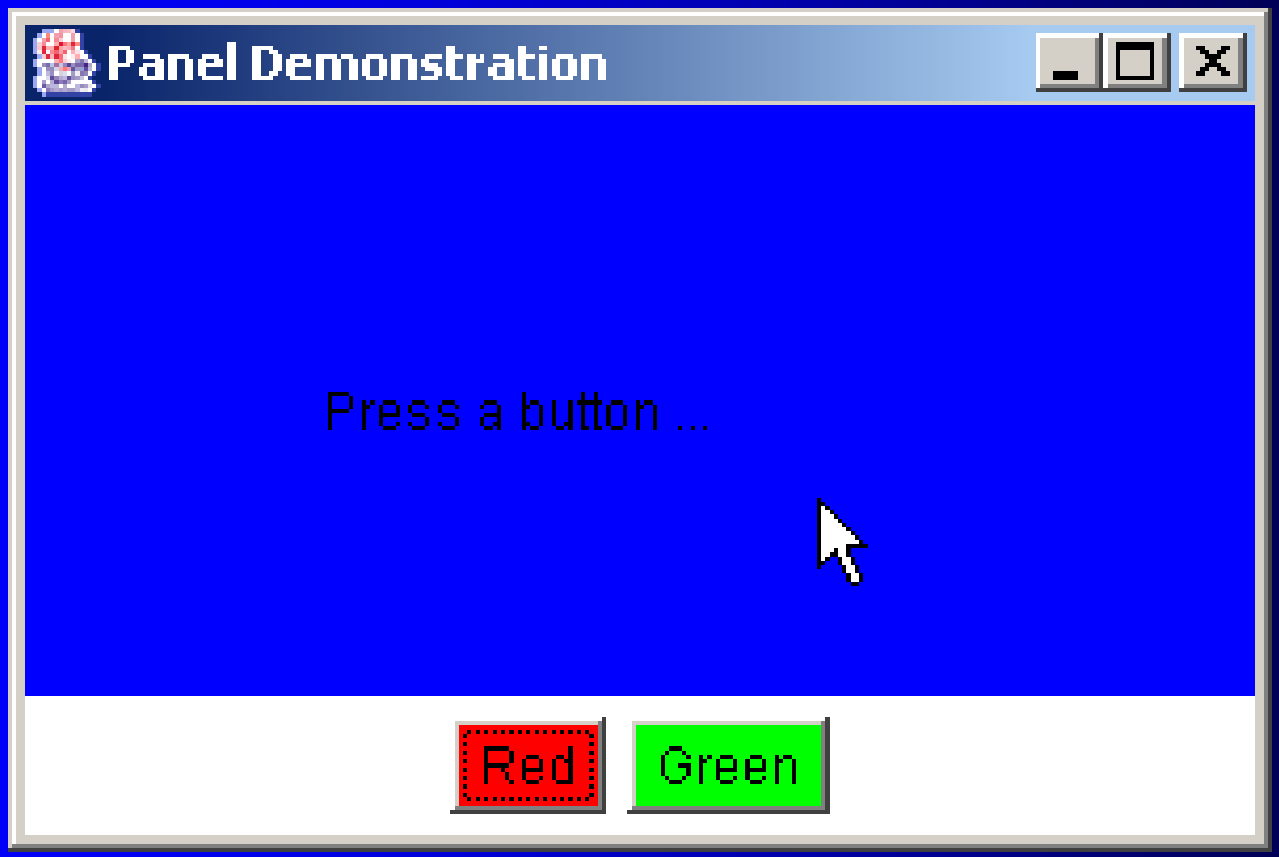
```
        Button stopButton = new Button("Red");
        stopButton.setBackground(Color.red);
        stopButton.addActionListener(this);
        buttonPanel.add(stopButton);

        Button goButton = new Button("Green");
        goButton.setBackground(Color.green);
        goButton.addActionListener(this);
        buttonPanel.add(goButton);

        setLayout(new BorderLayout());
        add(buttonPanel, "South");
    }
```

```
public void paint(Graphics g) {
        g.drawString(theText, 75, 100);
    }

public void actionPerformed(ActionEvent e) {
       if (e.getActionCommand().equals("Red")) {
            setBackground(Color.red);
            theText = "STOP";
        }
        else if (e.getActionCommand().equals("Green")) {
           setBackground(Color.green);
           theText = "GO";
        }
```

```
else
            theText = "Error in button interface.";

        repaint(); //force colour and text change
    }
}
```

Program execution

Java PanelDemo



Some notes

- Panel is a container class. A Panel object can be added into a frame. We can add objects such as buttons, checkboxes into a panel. In the example

```
Panel buttonPanel = new Panel();
… …
Button goButton = new Button("Green");
… …
buttonPanel.add(goButton);
```

TextArea and TextField

- The following example demonstrates the way we use text area and text field for input and output.

Example

```
//TextAreaDemo.java
/* get/set text from/into a TextArea object
    (Adapted from Savitch) */
import java.awt.*;
import java.awt.event.*;

public class TextAreaDemo extends Frame implements
    ActionListener{
    public static final int WIDTH = 600;
    public static final int HEIGHT = 300;
    public static final int X = 20;
    public static final int Y = 20;

    private Panel textPanel;
    private TextArea theText;
    private String memo1 = "No Memo 1.";
    private String memo2 = "No Memo 2.";
```

```
public TextAreaDemo() {
        setTitle("Memo Saver");
        setLayout(new BorderLayout());
        setSize(WIDTH, HEIGHT);
        setLocation(X, Y);
        addWindowListener(new WindowDestroyer());

        Panel buttonPanel = new Panel();
        buttonPanel.setBackground(Color.white);
        buttonPanel.setLayout(new FlowLayout());

        Button memo1Button = new Button("Save Memo 1");
        memo1Button.addActionListener(this);
        buttonPanel.add(memo1Button);
```

```
Button memo2Button = new Button("Save Memo 2");
memo2Button.addActionListener(this);
buttonPanel.add(memo2Button);
Button clearButton = new Button("Clear");
clearButton.addActionListener(this);
buttonPanel.add(clearButton);
Button get1Button = new Button("Get Memo 1");
get1Button.addActionListener(this);
buttonPanel.add(get1Button);
Button get2Button = new Button("Get Memo 2");
get2Button.addActionListener(this);
buttonPanel.add(get2Button);
```

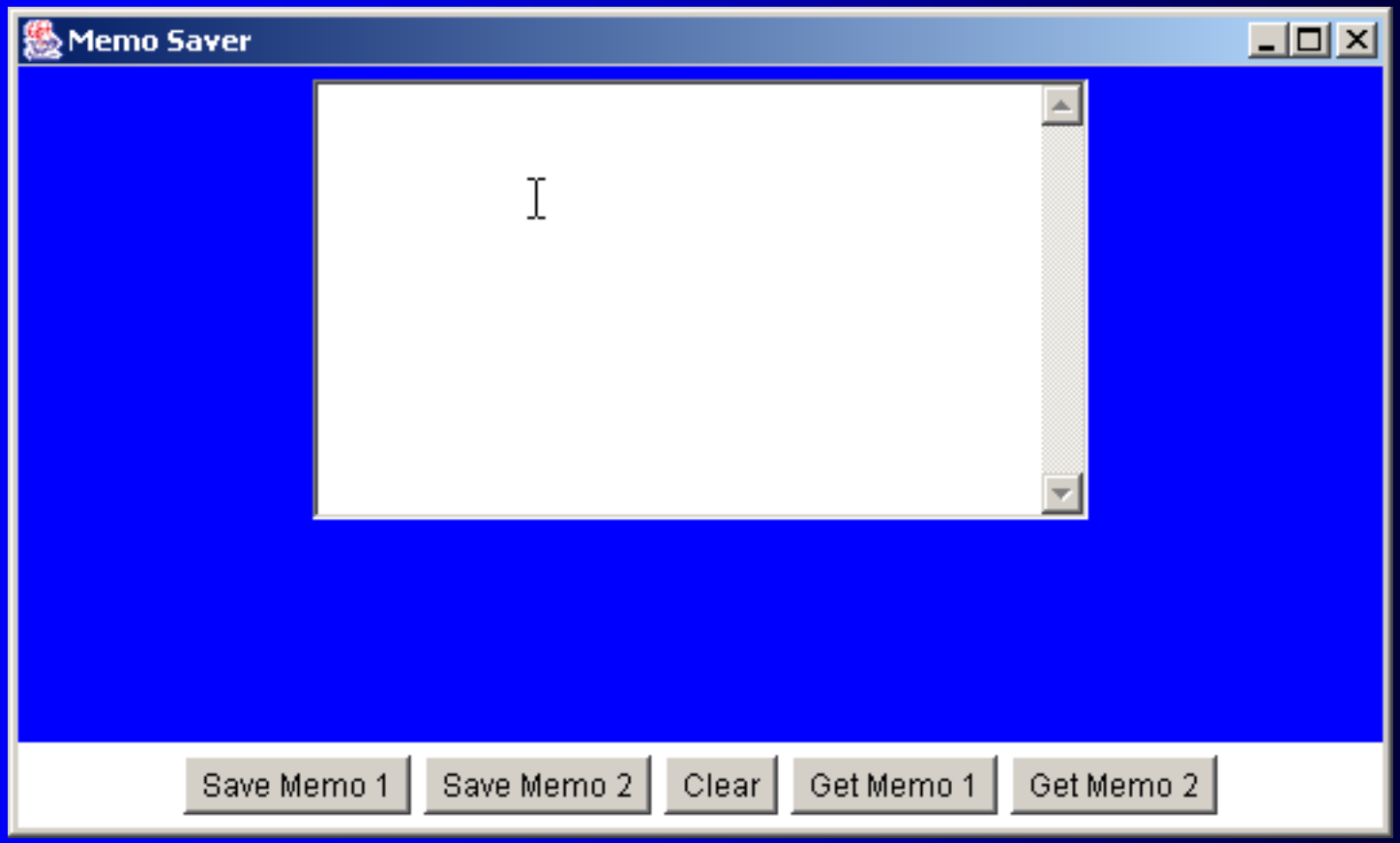
```
        add(buttonPanel, "South");

        textPanel = new Panel();
        textPanel.setBackground(Color.blue);
        theText = new TextArea(10, 40);
        theText.setBackground(Color.white);
        textPanel.add(theText);
        add(textPanel, "Center");
    }
```

```
public void actionPerformed(ActionEvent e) {
        String actionCommand = e.getActionCommand();
        if (actionCommand.equals("Save Memo 1"))
            memo1 = theText.getText();
        else if (actionCommand.equals("Save Memo 2"))
            memo2 = theText.getText();
        else if (actionCommand.equals("Clear"))
            theText.setText("");
        else if (actionCommand.equals("Get Memo 1"))
            theText.setText(memo1);
        else if (actionCommand.equals("Get Memo 2"))
            theText.setText(memo2);
        else
            theText.setText("Error in memo interface");
```

```
    textPanel.repaint();//Shows changes in textPanel
     }

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

Program Execution

Java TextAreaDemo



- Some notes

- Both TextArea and TextField are subclasses of java.awt.TextComponent. They both can be used to add text to a frame. The major difference between the two classes is that a TextArea object represents a two dimensional text area (*ie.* we can specify the number of chars per line and the number of lines in the area) while a TextField object is one dimensional (*i.e.* we can only specify the length of the field).

For example

```
TextArea ta = new TextArea(“This is a text area”, 10, 40);
TextField tf = new TextField(“This is a text field”, 20);
```