Module 10

Building Java GUIs

Objectives

- Describe the Abstract Window Toolkit (AWT) package and its components
- Define the terms *containers*, *components*, and *layout managers*, and describe how they work together to build a GUI
- Use layout managers
- Use the `FlowLayout`, `BorderLayout`, and `GridLayout` managers to achieve a desired dynamic layout
- Add components to a container
- Use the `Frame` and `Panel` containers appropriately
- Describe how complex layouts with nested containers work

Relevance

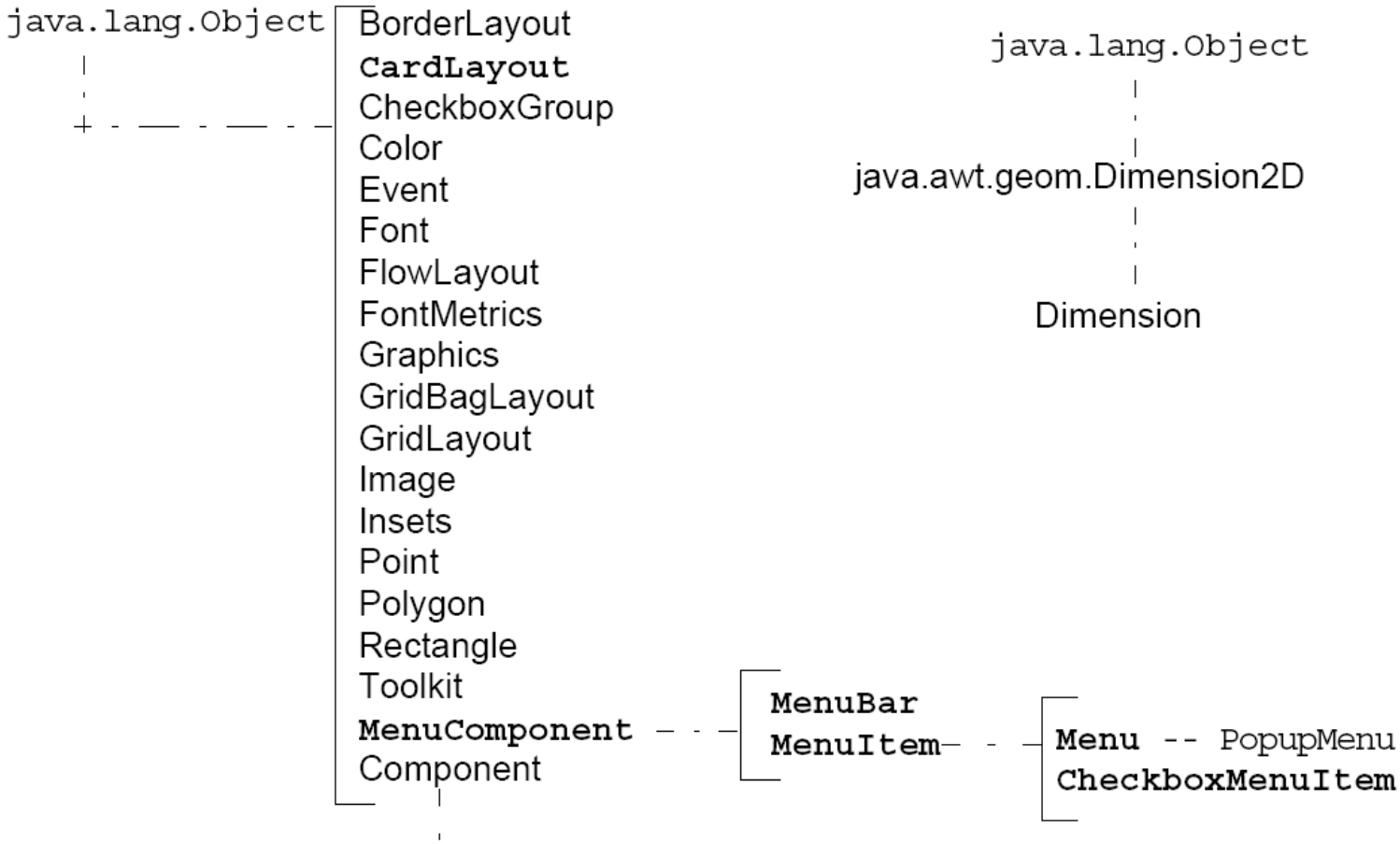
As a platform-independent programming language, how is Java technology used to make the graphical user interface (GUI) platform-independent?

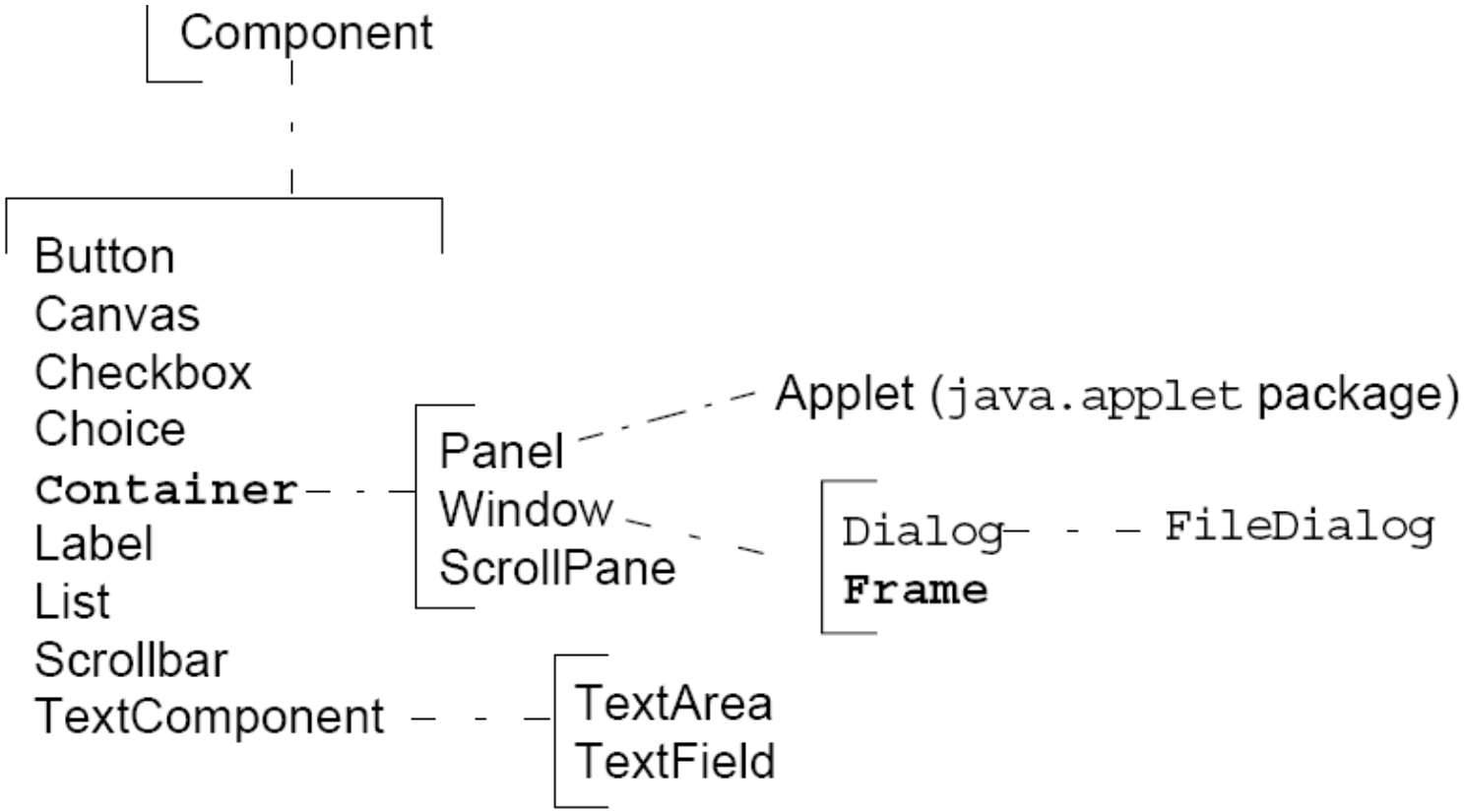
Abstract Window Toolkit

The AWT performs the following:

- Provides GUI components that are used in all Java applets and applications
- Contains classes that can be composed or extended; classes can also be abstract
- Ensures that every GUI component that is displayed on the screen is a subclass of the abstract class `Component` or `MenuComponent`
- Has `Container`, which is an abstract subclass of `Component` and includes two subclasses:
  - `Panel`
  - `Window`

The `java.awt` Package





Exceptions – `AWTException`

Errors – `AWTError`

Containers

- Add components with the `add()` method.
- The two main types of containers are `Window` and `Panel`.
- A `Window` is a free floating window on the display.
- A `Panel` is a container of GUI components that must exist in the context of some other container, such as a window or applet.

Positioning Components

- The position and size of a component in a container is determined by a layout manager.
- You can control the size or position of components by disabling the layout manager.

  You must then use `setLocation()`, `setSize()`, or `setBounds()` on components to locate them in the container.

Frames

Frames have the following characteristics:

- Are a subclass of `Window`
- Have title and resizing corners
- Are invisible initially; use `setVisible(true)` to expose the frame
- Have `BorderLayout` as the default layout manager
- Use the `setLayout` method to change the default layout manager

The `FrameExample` Class

```
import java.awt.*;

public class FrameExample {
  private Frame f;
  public FrameExample() {
    f = new Frame("Hello Out There!");
  }

  public void launchFrame() {
    f.setSize(170,170);
    f.setBackground(Color.blue);
    f.setVisible(true);
  }

  public static void main(String args[]) {
    FrameExample guiWindow = new FrameExample();
    guiWindow.launchFrame();
  }
}
```

Example `Frame`

Solaris OS

Microsoft Windows

Panels

- Panels provide a space for components.
- This enables subpanels to have their own layout manager.

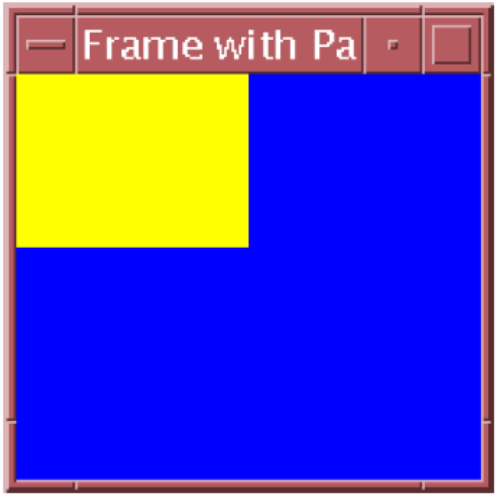
The `FrameWithPanel` Class

```
import java.awt.*;

public class FrameWithPanel {
  private Frame f;
  private Panel pan;

  public FrameWithPanel(String title) {
    f = new Frame(title);
    pan = new Panel();
  }
```

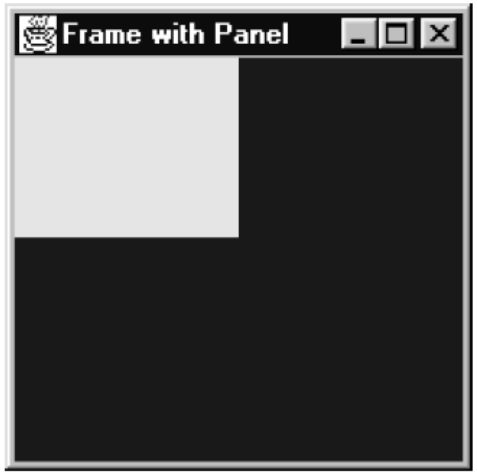
The `FrameWithPanel` Class

```
11
12    public void launchFrame() {
13      f.setSize(200,200);
14      f.setBackground(Color.blue);
15      f.setLayout(null);  // Use default layout
16
17      pan.setSize(100,100);
18      pan.setBackground(Color.yellow);
19      f.add(pan);
20      f.setVisible(true);
21    }
22
23    public static void main(String args[]) {
24      FrameWithPanel guiWindow =
25          new FrameWithPanel("Frame with Panel");
26      guiWindow.launchFrame();
27    }
28  }
```

Example `Panel`



Solaris OS

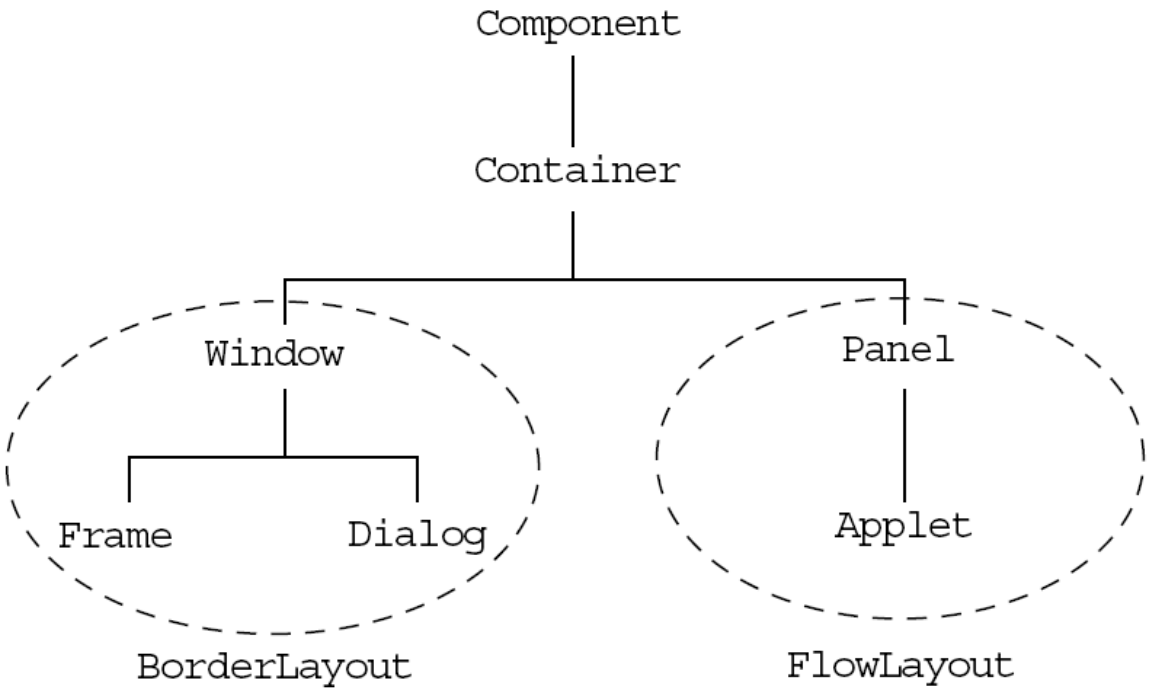


Microsoft Windows

Layout Managers

- `FlowLayout`
- `BorderLayout`
- `GridLayout`
- `CardLayout`
- `GridBagLayout`

Default Layout Managers



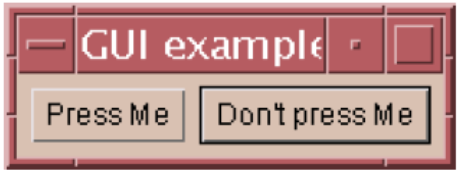
A Simple `FlowLayout` Example

```
import java.awt.*;

public class LayoutExample {
  private Frame f;
  private Button b1;
  private Button b2;

  public LayoutExample() {
    f = new Frame("GUI example");
    b1 = new Button("Press Me");
    b2 = new Button("Don't press Me");
  }
```

A Simple `FlowLayout` Example

```
13
14    public void launchFrame() {
15      f.setLayout(new FlowLayout());
16      f.add(b1);
17      f.add(b2);
18      f.pack();
19      f.setVisible(true);
20    }
21
22    public static void main(String args[]) {
23      LayoutExample guiWindow = new LayoutExample();
24      guiWindow.launchFrame();
25    }
26
27  } // end of LayoutExample class
```

Example of `FlowLayout`



Solaris OS

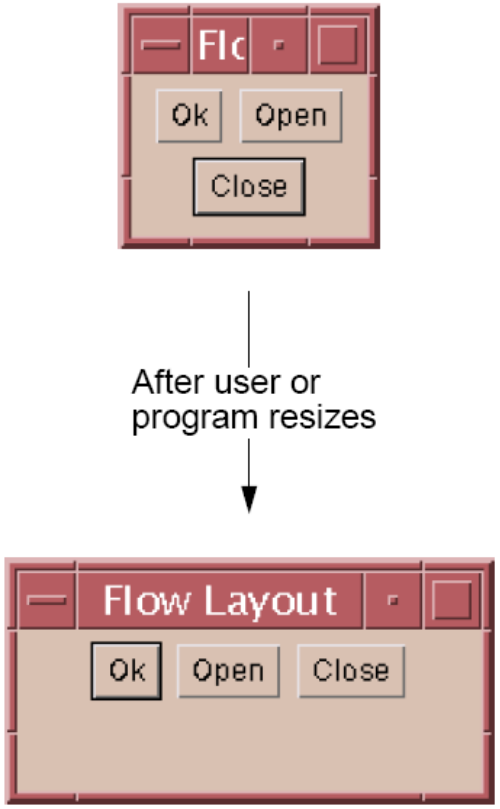
Microsoft Windows

The `FlowLayout` Manager

The `FlowLayout` manager has the following characteristics:

- Forms the default layout for the `Panel` class
- Adds components from left to right
- Alignment default is centered
- Uses components' preferred sizes
- Uses the constructor to tune behavior

The `FlowLayout` Resizing



Solaris OS

The `FlowExample` Class

```
import java.awt.*;

public class FlowExample {
  private Frame f;
  private Button button1;
  private Button button2;
  private Button button3;

  public FlowExample() {
    f = new Frame("Flow Layout");
    button1 = new Button("Ok");
    button2 = new Button("Open");
    button3 = new Button("Close");
  }
```

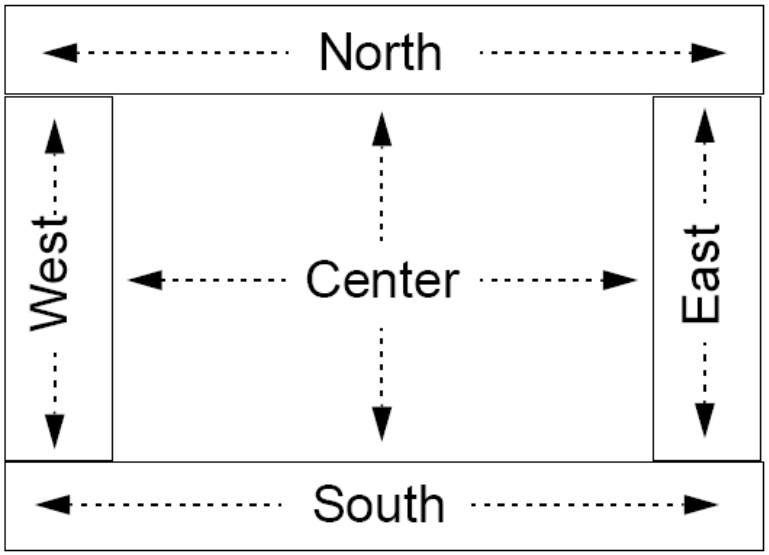
The `FlowExample` Class

```
15
16    public void launchFrame() {
17      f.setLayout(new FlowLayout());
18      f.add(button1);
19      f.add(button2);
20      f.add(button3);
21      f.setSize(100,100);
22      f.setVisible(true);
23    }
24
25    public static void main(String args[]) {
26      FlowExample guiWindow = new FlowExample();
27      guiWindow.launchFrame();
28    }
29  }
```

The `BorderLayout` Manager

- The `BorderLayout` manager is the default layout for the `Frame` class.
- Components are added to specific regions.
- The resizing behavior is as follows:
  - North, South, and Center regions adjust horizontally
  - East, West, and Center regions adjust vertically

Organization of the Border Layout Components



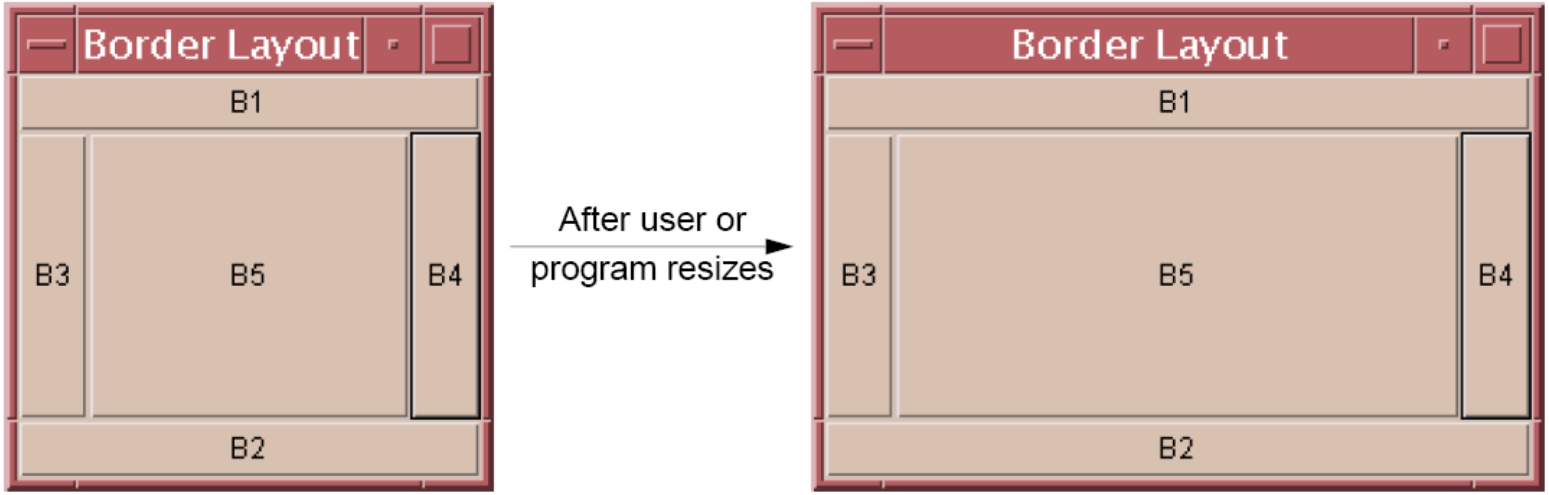
The `BorderExample` Class

```
import java.awt.*;

public class BorderExample {
  private Frame f;
  private Button bn, bs, bw, be, bc;

  public BorderExample() {
    f = new Frame("Border Layout");
    bn = new Button("B1");
    bs = new Button("B2");
    bw = new Button("B3");
    be = new Button("B4");
    bc = new Button("B5");
  }
```

The `BorderExample` Class

```
15
16    public void launchFrame() {
17      f.add(bn, BorderLayout.NORTH);
18      f.add(bs, BorderLayout.SOUTH);
19      f.add(bw, BorderLayout.WEST);
20      f.add(be, BorderLayout.EAST);
21      f.add(bc, BorderLayout.CENTER);
22      f.setSize(200,200);
23      f.setVisible(true);
24    }
25
26    public static void main(String args[]) {
27      BorderExample guiWindow2 = new BorderExample();
28      guiWindow2.launchFrame();
29    }
30  }
```

Example of `BorderLayout`



Solaris OS

The `GridLayout` Manager

- Components are added from left to right, and from top to bottom.
- All regions are sized equally.
- The constructor specifies the rows and columns.

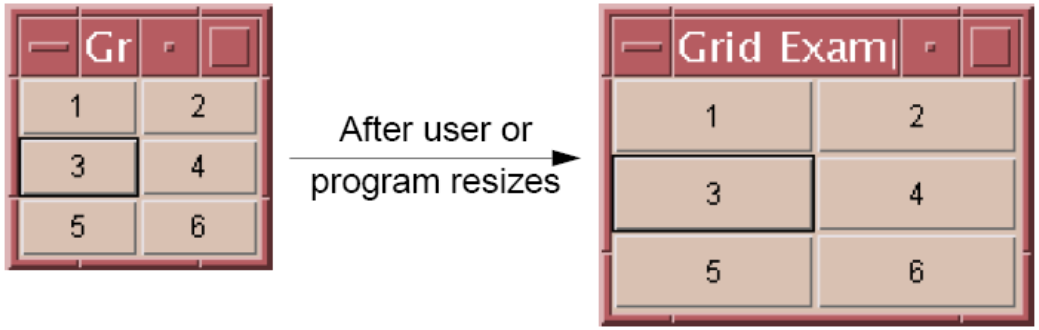
The `GridExample` Class

```
import java.awt.*;

public class GridExample {
  private Frame f;
  private Button b1, b2, b3, b4, b5, b6;

  public GridExample() {
    f = new Frame("Grid Example");
    b1 = new Button("1");
    b2 = new Button("2");
    b3 = new Button("3");
    b4 = new Button("4");
    b5 = new Button("5");
    b6 = new Button("6");
  }
```

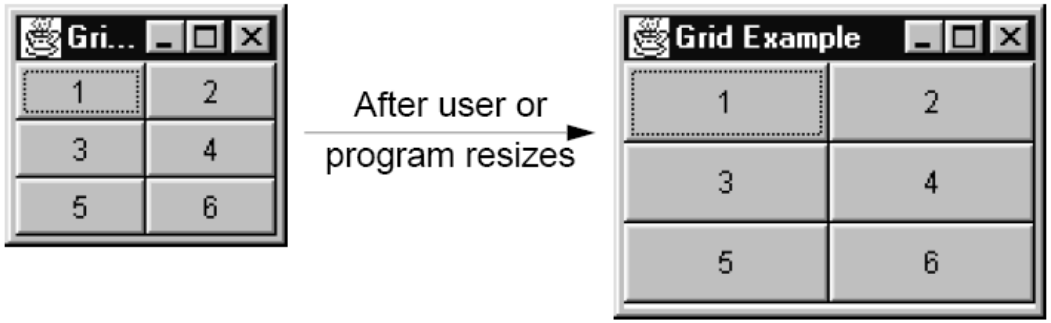
The `GridExample` Class

```
16
17    public void launchFrame() {
18      f.setLayout (new GridLayout(3,2));
19      f.add(b1);
20      f.add(b2);
21      f.add(b3);
22      f.add(b4);
23      f.add(b5);
24      f.add(b6);
25      f.pack();
26      f.setVisible(true);
27    }
28
29    public static void main(String args[]) {
30      GridExample grid = new GridExample();
31      grid.launchFrame();
32    }
33  }
```

Example of `GridLayout`



Solaris OS



Microsoft Windows

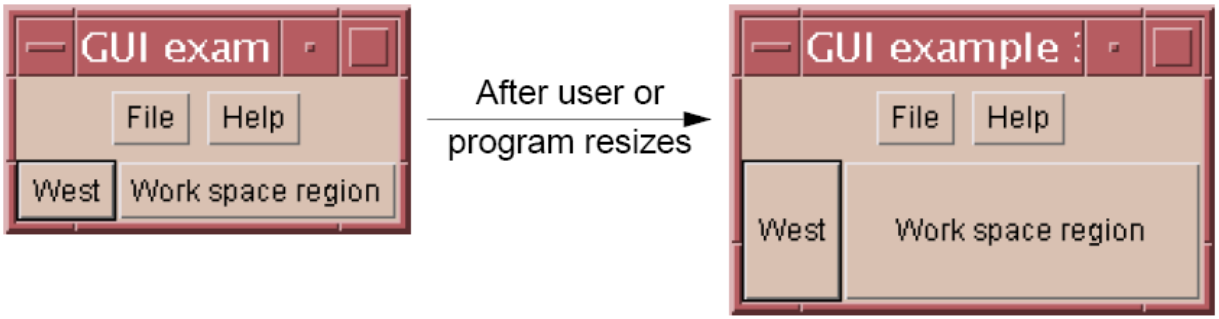
The `ComplexLayoutExample` Class

```
import java.awt.*;

public class ComplexLayoutExample {
  private Frame f;
  private Panel p;
  private Button bw, bc;
  private Button bfile, bhelp;

  public ComplexLayoutExample() {
    f = new Frame("GUI example 3");
    bw = new Button("West");
    bc = new Button("Work space region");
    bfile = new Button("File");
    bhelp = new Button("Help");
  }
```

The `ComplexLayoutExample` Class

```
public void launchFrame() {
    // Add bw and bc buttons in the frame border
    f.add(bw, BorderLayout.WEST);
    f.add(bc, BorderLayout.CENTER);
    // Create panel for the buttons in the north border
    p = new Panel();
    p.add(bfile);
    p.add(bhelp);
    f.add(p, BorderLayout.NORTH);
    // Pack the frame and make it visible
    f.pack();
    f.setVisible(true);
  }

  public static void main(String args[]) {
    ComplexLayoutExample gui = new ComplexLayoutExample();
    gui.launchFrame();
  }
}
```

Combining Layout Managers



Solaris OS

Drawing in AWT

- You can draw in any `Component` (although AWT provides the `Canvas` and `Panel` classes just for this purpose).
- Typically, you create a subclass of `Canvas` or `Panel` and override the `paint` method.
- The `paint` method is called every time the component is shown (for example, if another window overlapped the component and was then removed).
- Every component has a `Graphics` object.
- The `Graphics` class implements many drawing methods.

Various Shapes Drawn by the `Graphics` Object