27. GUI Programming

Java

Summer 2008

Instructor: Dr. Masoud Yaghini

GUI Programming

- Until now, you have only used dialog boxes and the command window for input and output.
- You used JOptionPane.showInputDialog to obtain input, and JOptionPane.showMessageDialog and System.out.println to display results.
- These approaches have limitations and are inconvenient.
- For example, to read ten numbers, you have to open ten input dialog boxes.

GUI Programming

- Starting with this chapter, you will learn Java GUI programming.
- You will create custom graphical user interfaces (GUI, pronounced goo-ee) to obtain input and display output in the same user interface.
- This chapter introduces the basics of Java GUI programming.
- Specifically, it discusses GUI components and their relationships, containers and layout managers, colors, fonts, borders, and tool tips.

Outline

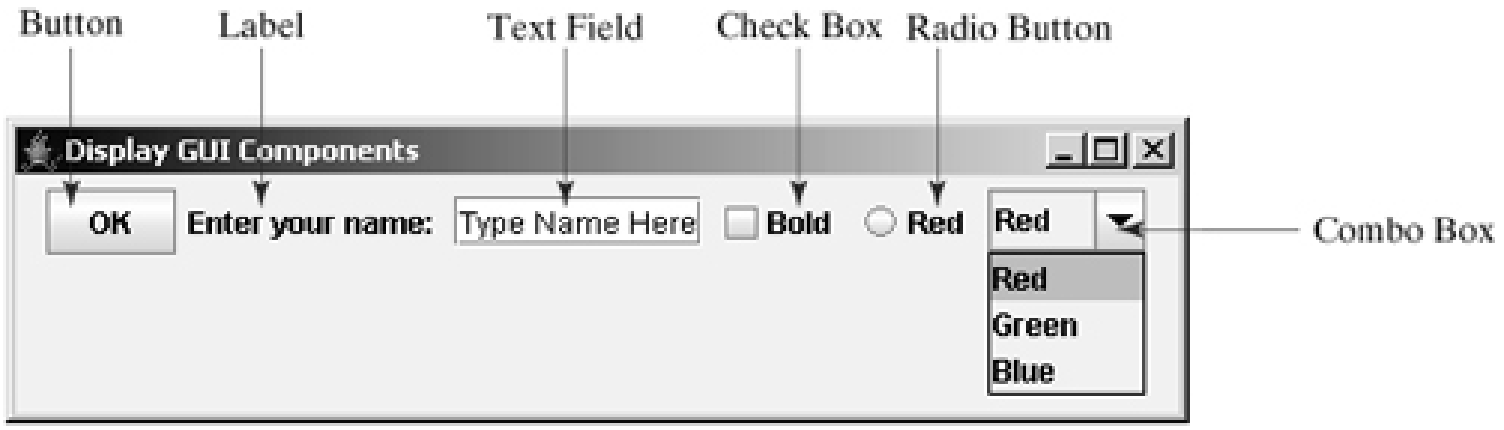
- The Java GUI API
- Frames
- Layout Managers
- The FlowLayout Class
- The GridLayout Class
- The BorderLayout Class
- References

The Java GUI API

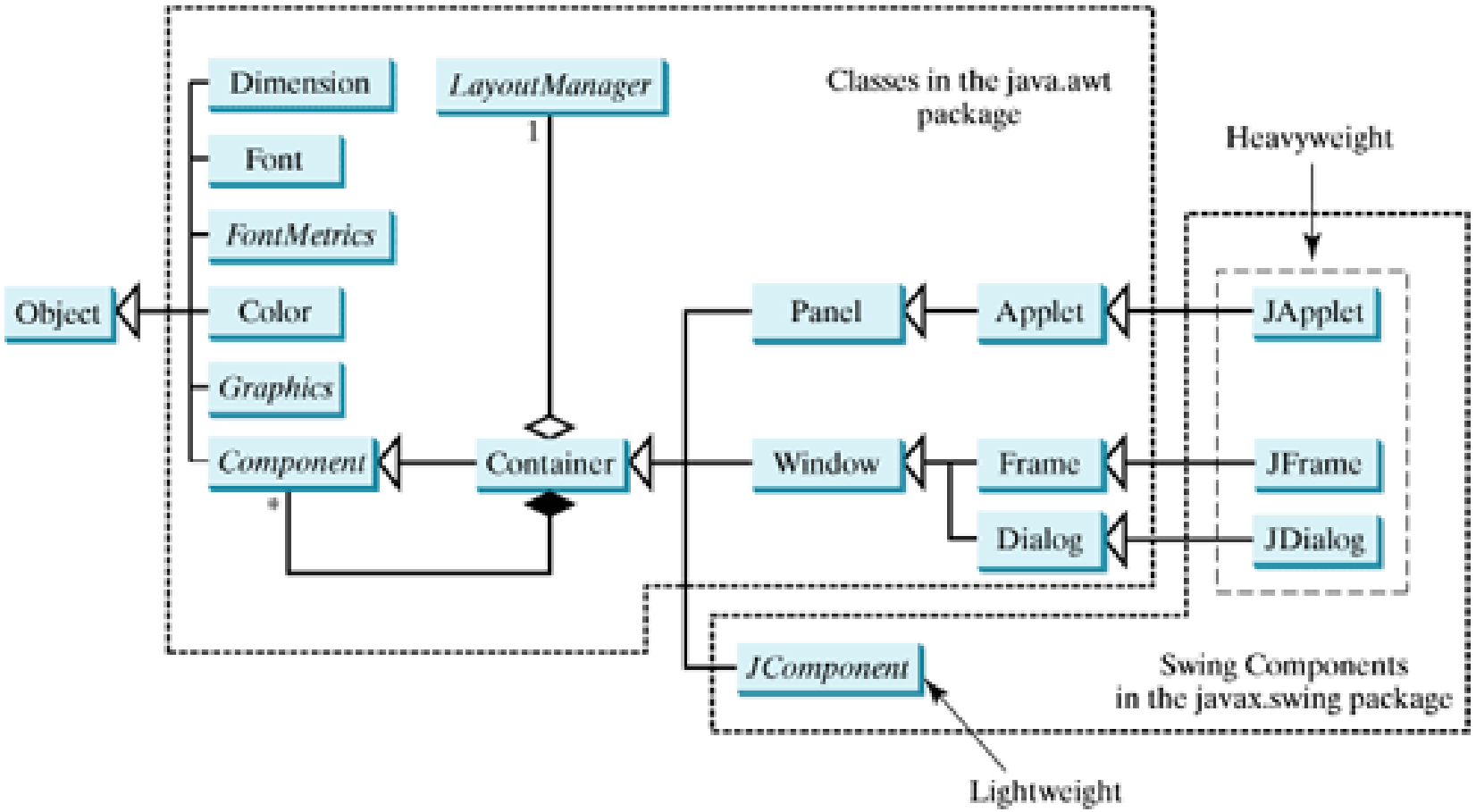
GUI Components

- You create graphical user interfaces using GUI objects such as buttons, labels, text fields, check boxes, radio buttons, and combo boxes.
- Each type of GUI object is defined in a class, such as JButton, JLabel, JTextField, JCheckBox, JRadioButton, and JComboBox.
- Each GUI component class provides several constructors that you can use to create GUI component objects.

```
// Create a button with text OK
JButton jbtOK = new JButton("OK");

// Create a label with text "Enter your name: "
JLabel jlblName = new JLabel("Enter your name: ");

// Create a text field with text "Type Name Here"
JTextField jtfName = new JTextField("Type Name Here");

// Create a check box with text bold
JCheckBox jchkBold = new JCheckBox("Bold");

// Create a radio button with text red
JRadioButton jrbRed = new JRadioButton("Red");

// Create a combo box with choices red, green, and blue
JComboBox jcboColor = new JComboBox(new String[]{"Red",
  "Green", "Blue"});
```

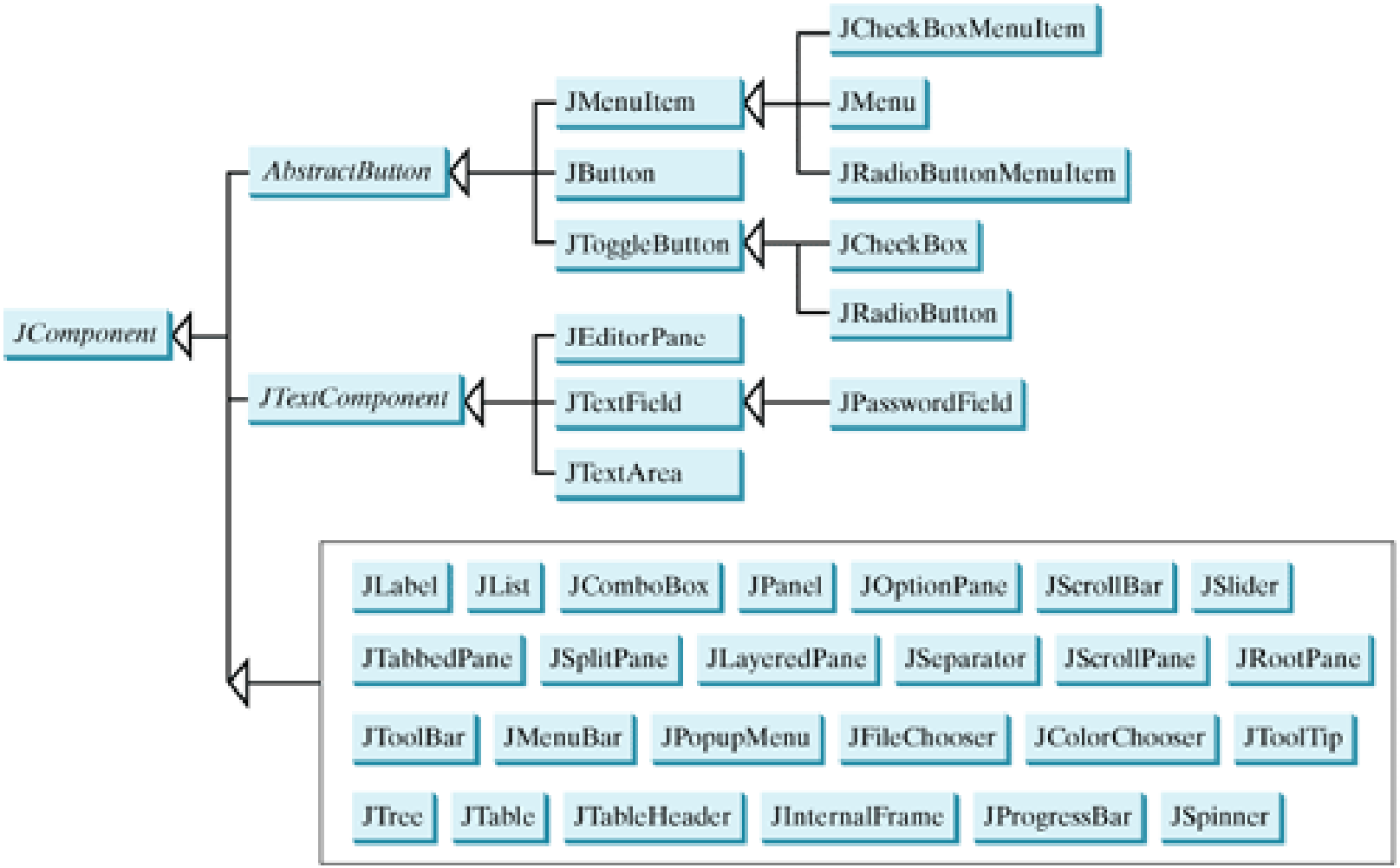


GUI Class Hierarchy



AWT: Abstract Windows Toolkit

JComponent and its subclasses



The Java GUI API

- The GUI classes can be classified into three groups:
  - ***Swing Component classes***.
    - such as JButton, JTextField, JTextArea, JComboBox, JList, JRadioButton, and JMenu,
    - are subclasses of JComponent.
  - ***Container classes***:
    - such as JFrame, JPanel, and JApplet,
    - are used to contain other components.
  - ***Helper classes***:
    - such as Graphics, Color, Font, FontMetrics, and Dimension,
    - are used to describe the properties of GUI components, such as graphics context, colors, fonts, and dimension.

Swing GUI Components

- Component is a superclass of all the user-interface classes, and JComponent is a superclass of all the lightweight Swing components
- Since JComponent is an abstract class, you cannot use new JComponent() to create an instance of JComponent.
- You can use the constructors of concrete subclasses of JComponent to create JComponent instances.

Frames

Creating a Frame

- Frame is a window that is not contained inside another window.
- Frame is the basis to contain other user interface components in Java GUI applications.
- The JFrame class can be used to create windows.

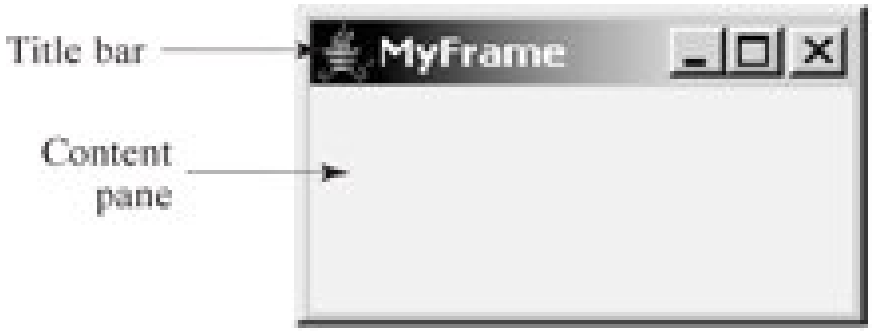
javax.swing.JFrame

- +JFrame()
  - Creates a default frame with no title.
- +JFrame(title: String)
  - Creates a frame with the specified title.
- +getSize(width: int, height: int): void
  - Specifies the size of the frame.
- +setLocation(x: int, y: int): void
  - Specifies the upper-left corner location of the frame.
- +setVisible(visible: boolean): void
  - Sets true to display the frame.

javax.swing.JFrame

- +setDefaultCloseOperation(mode: int): void
  - Specifies the operation when the frame is closed.
- +setLocationRelativeTo (c: Component): void
  - Sets the location of the frame relative to the specified component. If the component is null, the frame is centered on the screen.
- The setSize method is defined in the Component class, and is inherited by the JFrame class.

```
1  package chapter12;
2
3  import javax.swing.*;
4
5  public class MyFrame {
6    public static void main(String[] args) {
7      JFrame frame = new JFrame("MyFrame"); // Create a frame
8      frame.setSize(400, 300); // Set the frame size
9      frame.setLocationRelativeTo(null); // centers the frame on the screen
10     // to terminate program when the frame is closed
11     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
12     frame.setVisible(true); // Display the frame
13   }
14 }
```



```
1  package chapter12;
2
3  import javax.swing.*;
4
5  public class MyFrameWithComponents {
6    public static void main(String[] args) {
7      JFrame frame = new JFrame("MyFrameWithComponents");
8
9      // Add a button into the frame
10     JButton jbtOK = new JButton("OK");
11     frame.add(jbtOK);
12
13     frame.setSize(400, 300);
14     frame.setVisible(true);
15     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
16     frame.setLocationRelativeTo(null); // centers the frame on the screen
17   }
18 }
```

Layout Managers

Layout Managers

- Java's layout managers provide a level of abstraction to automatically map your user interface on all window systems.
- The GUI components are placed in containers.
- Each container has a layout manager to arrange the GUI components within the container.
- Layout managers are set in containers using the setLayout(LayoutManager) method in a container.

Layout Managers

- Some kinds of Layout Managers:
  - FlowLayout class
  - GridLayout class
  - BorderLayout class

Layout Managers

- Layout managers are set in containers using the setLayout(LayoutManager) method.
- For example, you can use the following statements to create an instance of FlowLayout and set it in a container:

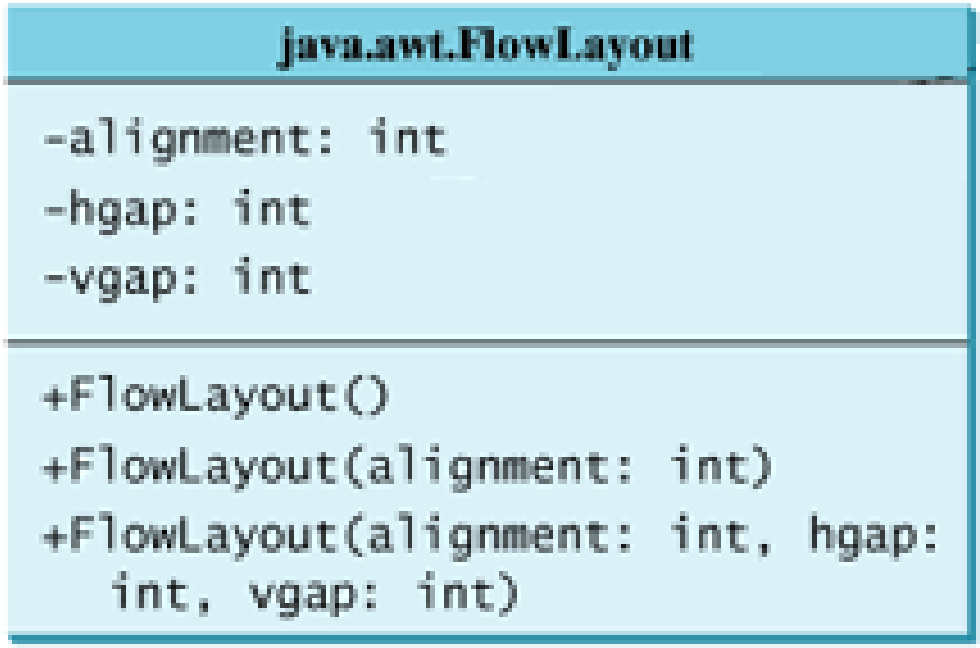
```
LayoutManager layoutManager = new FlowLayout();
container.setLayout(layoutManager);
```

The FlowLayout Class

The FlowLayout Class

- Java.awt.FlowLayout is the simplest layout manager.
- The components are arranged in the container from left to right in the order in which they were added.

Java.awt.FlowLayout



- The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The FlowLayout Class

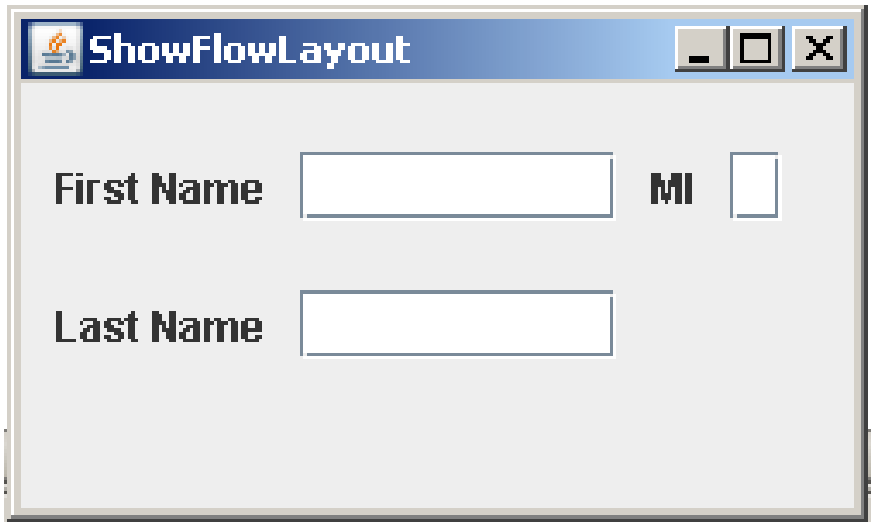
- Data fields:
  - -alignment: int
    - The alignment of this layout manager (default: CENTER).
  - -hgap: int
    - The horizontal gap of this layout manager (default: 5 pixels).
  - -vgap: int
    - The vertical gap of this layout manager (default: 5 pixels).

The FlowLayout Class

- Constructors:
  - +FlowLayout()
    - Creates a default FlowLayout manager.
  - +FlowLayout(alignment: int)
    - Creates a FlowLayout manager with a specified alignment.
  - +FlowLayout(alignment: int, hgap: int, vgap: int)
    - Creates a FlowLayout manager with a specified alignment, horizontal gap, and vertical gap.

The FlowLayout Class

- Write a program that adds three labels and text fields into the content pane of a frame with a FlowLayout manager.



```
1  package chapter12;
2
3  import javax.swing.JLabel;
4  import javax.swing.JTextField;
5  import javax.swing.JFrame;
6  import java.awt.FlowLayout;
7
8  public class ShowFlowLayout {
9
10   /** Main method */
11   public static void main(String[] args) {
12
13     JFrame frame = new JFrame("ShowFlowLayout");
14
15     FlowLayout layout =  new FlowLayout(FlowLayout.LEFT, 10, 20);
16     frame.setLayout(layout);
17
18     // Add labels and text fields to the frame
19     frame.add(new JLabel("First Name"));
20     frame.add(new JTextField(8));
21     frame.add(new JLabel("MI"));
22     frame.add(new JTextField(1));
23     frame.add(new JLabel("Last Name"));
24     frame.add(new JTextField(8));
25
26     frame.setLocationRelativeTo(null); // centers the frame on the screen
27     frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
28     frame.setSize(250, 150);
29     frame.setVisible(true);
30   }
31 }
```

```
1  package chapter12;
2
3  import javax.swing.JLabel;
4  import javax.swing.JTextField;
5  import javax.swing.JFrame;
6  import java.awt.FlowLayout;
7
8  public class ShowFlowLayout2 extends JFrame {
9    public ShowFlowLayout2() {
10     // Set FlowLayout, aligned left with horizontal gap 10
11     // and vertical gap 20 between components
12     setLayout(new FlowLayout(FlowLayout.LEFT, 10, 20));
13
14     // Add labels and text fields to the frame
15     add(new JLabel("First Name"));
16     add(new JTextField(8));
17     add(new JLabel("MI"));
18     add(new JTextField(1));
19     add(new JLabel("Last Name"));
20     add(new JTextField(8));
21   }
```

```
22
23      /** Main method */
24      public static void main(String[] args) {
25
26          ShowFlowLayout2 frame = new ShowFlowLayout2();
27
28          frame.setTitle("ShowFlowLayout2");
29          frame.setLocationRelativeTo(null); // centers the frame on the screen
30          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31          frame.setSize(200, 200);
32          frame.setVisible(true);
33      }
34  }
```

The FlowLayout Class

- The constructor ShowFlowLayout() does not explicitly invoke the constructor JFrame(), but the constructor JFrame() is invoked implicitly.

- The setTitle method is defined in the java.awt.Frame class. Since JFrame is a subclass of Frame, you can use it to set a title for an object of JFrame.

The FlowLayout Class

- The ShowFlowLayout can be easily reused.
- For example, you can create multiple frames by creating multiple instances of the class.

The FlowLayout Class

- FlowLayout has alignment, hgap, and vgap properties.
- You can use the setAlignment, setHgap, and setVgap methods to specify the alignment and the horizontal and vertical gaps.

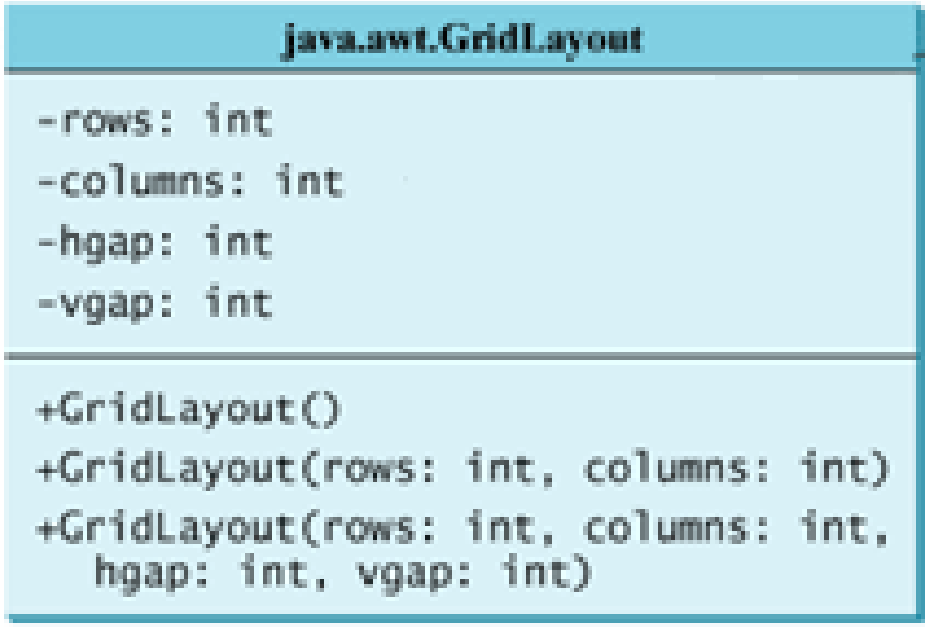
```
// Create a layout manager
FlowLayout flowLayout = new FlowLayout();
// Set layout properties
flowLayout.setAlignment(FlowLayout.RIGHT);
flowLayout.setHgap(10);
flowLayout.setVgap(20);
```

The GridLayout Class

The GridLayout Class

- The GridLayout manager arranges components in a grid (matrix) formation with the number of rows and columns defined by the constructor.
- The components are placed in the grid from left to right, starting with the first row, then the second, and so on, in the order in which they are added.

The GridLayout Class



- The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The GridLayout Class

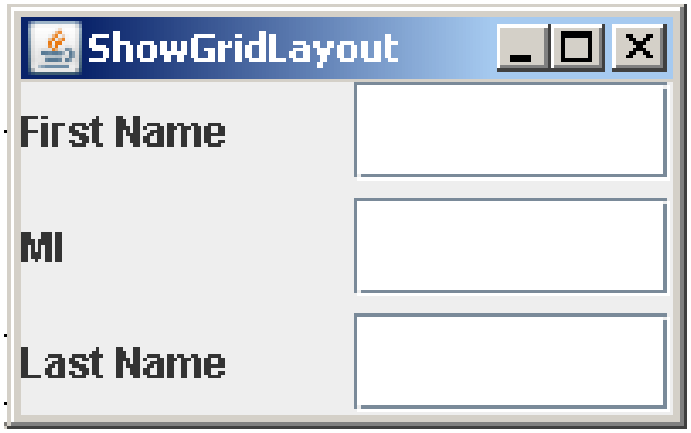
- Data fields:
  - -rows: int
    - The number of rows in this layout manager (default: 1).
  - -columns: int
    - The number of columns in this layout manager (default: 1).
  - -hgap: int
    - The horizontal gap of this layout manager (default: 5 pixels).
  - -vgap: int
    - The vertical gap of this layout manager (default: 5 pixels).

The GridLayout Class

- Constructors:
  - +GridLayout()
    - Creates a default GridLayout manager.
  - +GridLayout(rows: int, columns: int)
    - Creates a GridLayout with a specified number of rows and columns.
  - +GridLayout(rows: int, columns: int, hgap: int, vgap: int)
    - Creates a GridLayout manager with a specified number of rows and columns, horizontal gap, and vertical gap.

The GridLayout Class

- Rewrite the program in the preceding example using a GridLayout manager instead of a FlowLayout manager to display the labels and text fields.



```
1  package chapter12;
2
3  import javax.swing.JLabel;
4  import javax.swing.JTextField;
5  import javax.swing.JFrame;
6  import java.awt.GridLayout;
7
8  public class ShowGridLayout extends JFrame {
9    public ShowGridLayout() {
10     // Set GridLayout, 3 rows, 2 columns, and gaps 5 between
11     // components horizontally and vertically
12     setLayout(new GridLayout(3, 2, 5, 5));
13
14     // Add labels and text fields to the frame
15     add(new JLabel("First Name"));
16     add(new JTextField(8));
17     add(new JLabel("MI"));
18     add(new JTextField(1));
19     add(new JLabel("Last Name"));
20     add(new JTextField(8));
21   }
```

```
22
23      /** Main method */
24      public static void main(String[] args) {
25
26          ShowGridLayout frame = new ShowGridLayout();
27
28          frame.setTitle("ShowGridLayout");
29          frame.setLocationRelativeTo(null); // centers the frame on the screen
30          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
31          frame.setSize(200, 125);
32          frame.setVisible(true);
33      }
34  }
```

The GridLayout Class

- All components are given equal size in the container of GridLayout.
- In FlowLayout and GridLayout, the order in which the components are added to the container is important.
- It determines the location of the components in the container.

The GridLayout Class

- GridLayout has the rows, columns, hgap, and vgap properties.

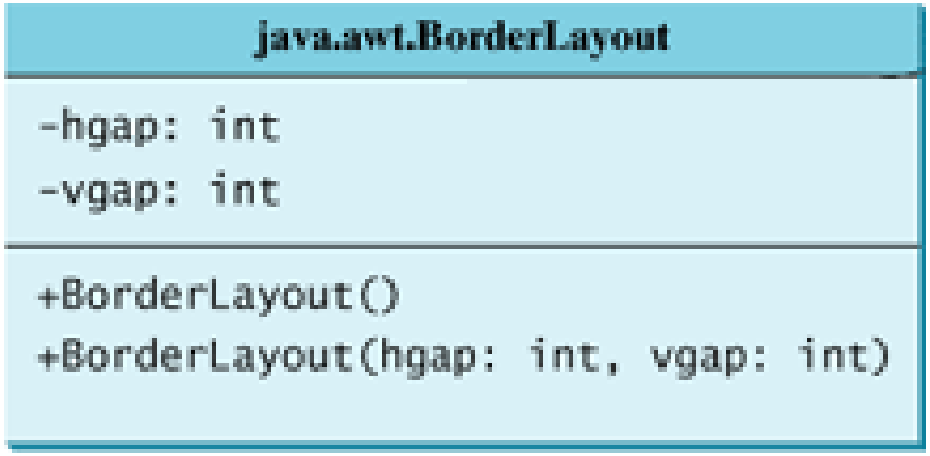
- You can use the setRows, setColumns, setHgap, and setVgap methods to specify the number of rows, the number of columns, and the horizontal and vertical gaps.

The BorderLayout Class

The BorderLayout Class

- The BorderLayout manager divides the window into five areas: East, South, West, North, and Center.
- Components are added to a BorderLayout by using add(Component, index), where index is a constant:
  - BorderLayout.EAST,
  - BorderLayout.SOUTH,
  - BorderLayout.WEST,
  - BorderLayout.NORTH, or
  - BorderLayout.CENTER.

The BorderLayout Class



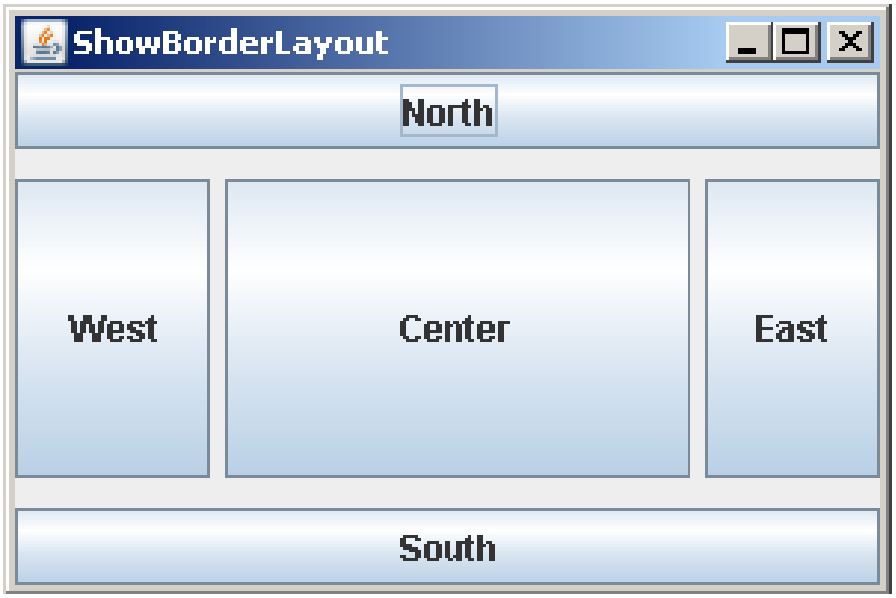
- The get and set methods for these data fields are provided in the class, but omitted in the UML diagram for brevity.

The BorderLayout Class

- Data fields:
  - -hgap: int
    - The horizontal gap of this layout manager (default: 0).
  - -vgap: int
    - The vertical gap of this layout manager (default: 0).
- Constructors:
  - +BorderLayout()
    - Creates a default BorderLayout manager.
  - +BorderLayout(hgap: int, vgap: int)
    - Creates a BorderLayout manager with a specified number of horizontal gap, and vertical gap.

The BorderLayout Class

- Write a program that demonstrates border layout.
- The program adds five buttons labeled East, South, West, North, and Center into the frame with a BorderLayout manager.



```
1  package chapter12;
2
3  import javax.swing.JButton;
4  import javax.swing.JFrame;
5  import java.awt.BorderLayout;
6
7  public class ShowBorderLayout extends JFrame {
8    public ShowBorderLayout() {
9      // Set BorderLayout with horizontal gap 5 and vertical gap 10
10     setLayout(new BorderLayout(5, 10));
11
12     // Add buttons to the frame
13     add(new JButton("East"), BorderLayout.EAST);
14     add(new JButton("South"), BorderLayout.SOUTH);
15     add(new JButton("West"), BorderLayout.WEST);
16     add(new JButton("North"), BorderLayout.NORTH);
17     add(new JButton("Center"), BorderLayout.CENTER);
18   }
```

```
19
20      /** Main method */
21      public static void main(String[] args) {
22          ShowBorderLayout frame = new ShowBorderLayout();
23          frame.setTitle("ShowBorderLayout");
24          frame.setLocationRelativeTo(null); // centers the frame on the screen
25          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
26          frame.setSize(300, 200);
27          frame.setVisible(true);
28      }
29  }
```

The BorderLayout Class

- BorderLayout has the hgap and vgap properties.
- You can use the setHgap and setVgap methods to specify the horizontal and vertical gaps.

References

References

- Y. Daniel Liang, **Introduction to Java Programming**, Sixth Edition, Pearson Education, 2007. (Chapter 12)

The End