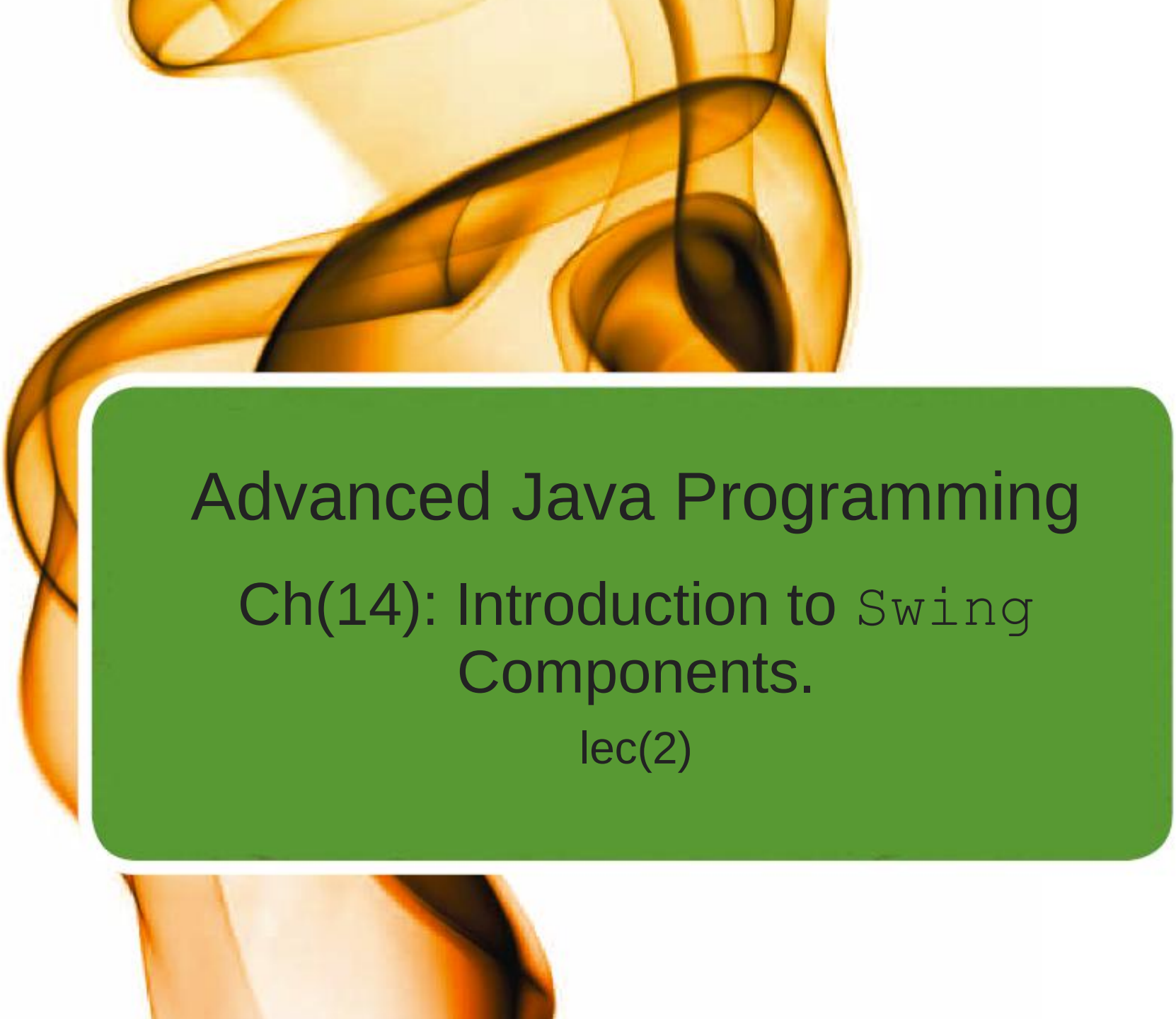


Advanced Java Programming

Ch(14): Introduction to `Swing` Components.

lec(2)

Understanding `Swing` Components

- GUI components
  - Buttons, text fields, and other components with which the user can interact
- `Swing` component
  - Descendant of `JComponent`
  - Inherits from `java.awt.Container` class
- import `javax.swing.*` package to take advantage of the Swing UI components and their methods.

Understanding Swing Components (cont'd.)

- Container
  - Type of component that holds other components
  - Can treat group as single entity
  - Defined in `Container` class
  - Often takes form of window
    - Drag
    - Resize
    - Minimize
    - Restore
    - Close

Understanding Swing Components (cont'd.)

- `Window` class
  - Child of `Container`
  - Does not have title bars or borders
  - Rarely used
  - Instead use subclasses
    - `Frame`
    - `JFrame`

Using the `JFrame` Class

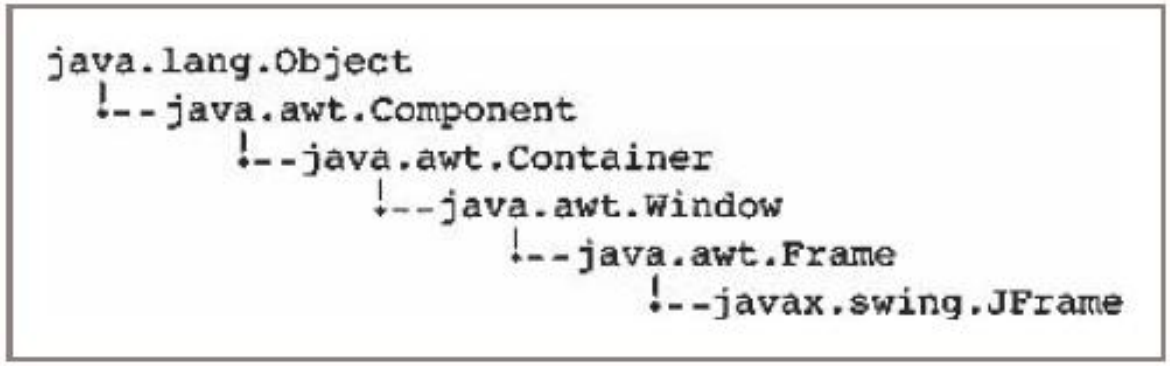


Figure 14-1 Relationship of the `JFrame` class to its ancestors

Using the `JFrame` Class (cont'd.)

- `JFrame` class
  - a place to other objects for display
- Constructors
  - **`JFrame()`** //constructs a new frame that initially is invisible and has no title.
  - **`JFrame(String title)`** //creates a new, initially invisible JFrame with the specified title.
  - Many other constructors see chapter (14).

Method	Purpose
`void setTitle(String)`	Sets a `JFrame`'s title using the `String` argument
`void setSize(int, int)`	Sets a `JFrame`'s size in pixels with the width and height as arguments
`void setSize(Dimension)`	Sets a `JFrame`'s size using a `Dimension` class object; the `Dimension(int, int)` constructor creates an object that represents both a width and a height
`String getTitle()`	Returns a `JFrame`'s title
`void setResizable(boolean)`	Sets the `JFrame` to be resizable by passing `true` to the method, or sets the `JFrame` not to be resizable by passing `false` to the method
`boolean isResizable()`	Returns `true` or `false` to indicate whether the `JFrame` is resizable
`void setVisible(boolean)`	Sets a `JFrame` to be visible using the `boolean` argument `true` and invisible using the `boolean` argument `false`
`void setBounds(int, int, int, int)`	Overrides the default behavior for the `JFrame` to be positioned in the upper-left corner of the computer screen's desktop; the first two arguments are the horizontal and vertical positions of the `JFrame`'s upper-left corner on the desktop, and the final two arguments set the width and height

Table 14-1 Useful methods inherited by the `JFrame` class

Using the `JFrame` Class (cont'd.)

```
import javax.swing.*;
public class JFrame1
{
   public static void main(String[] args)
   {
      JFrame aFrame = new JFrame("First frame");
      aFrame.setSize(250, 100);
      aFrame.setVisible(true);
   }
}
```

Figure 14-2 The JFrame1 application

Using the `JFrame` Class (cont'd.)

- Close `JFrame`
  - Click Close button
  - The default behavior : `JFrame` becomes hidden and application keeps running.
  - To change this behavior
    - Use `setDefaultCloseOperation(Arg)` method
    - example:
      myframe.setDefaultCloseOperation(**JFrame.EXIT_ON_CLOSE**);

Customizing a `JFrame`'s Appearance

- Window decorations
  - Icon and buttons in the frame's title bar
  - The coffee-cup icon, Minimize, Restore, and Close buttons
- Look and feel
  - Default appearance and behavior of user interface supplied by the operating system.
  - `setDefaultLookAndFeelDecorated()` method
    - Set `JFrame`'s look and fee
  - You can provide a custom icon
    - http://java.sun.com and search for "How to Make Frames."

Customizing a `JFrame`'s Appearance (cont'd.)

```
import javax.swing.*;
public class JFrame2
{
   public static void main(String[] args)
   {
      JFrame.setDefaultLookAndFeelDecorated(true);
      JFrame aFrame = new JFrame("Second frame");
      aFrame.setSize(250, 100);
      aFrame.setVisible(true);
   }
}
```

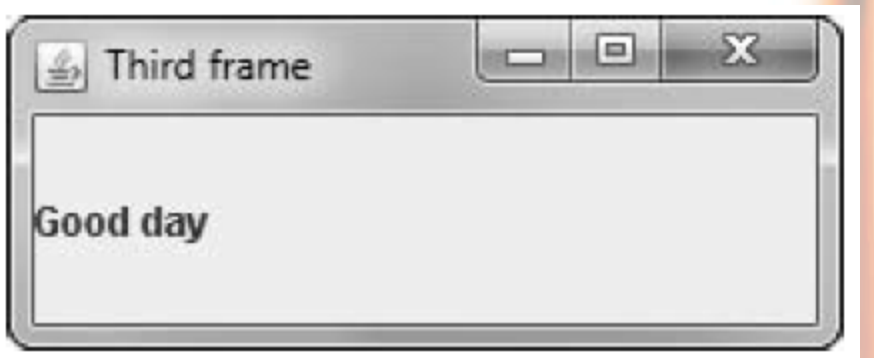
Figure 14-4 The JFrame2 class

Using a `JLabel`

- `Jlabel` **`class`**
  - Holds text you can display.
- Methods
  - `JLabel();` `//`creates a JLabel instance with no text/image.
  - `JLabel(Icon image);//`creates a JLabel instance with the specified image.
  - `JLabel(String text);//`creates a JLabel instance with the specified text.
  - `setText();//` change the label text
  - `getText();//` retrieve the label text

Using a `JLabel`

```
import javax.swing.*;
public class JFrame3
{
   public static void main(String[] args)
   {
      final int FRAME_WIDTH = 250;
      final int FRAME_HEIGHT = 100;
      JFrame aFrame = new JFrame("Third frame");
      aFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
      aFrame.setVisible(true);
      aFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JLabel greeting = new JLabel("Good day");
      aFrame.add(greeting);
   }
}
```



Changing a `JLabel`'s Font

- `setFont()` method
  - Requires a `Font` object argument
- `Font` class
  - Creates an object that holds typeface and size information
  - To construct a `Font` object Arguments:
    - Typeface: Arial, Century, Monospaced, and Times New Roman.
    - Style: Font.PLAIN, Font.BOLD, or Font.ITALIC.
    - point size: integer value. e.g: printed text is commonly 12 points; a headline might be 30 points.

Changing a `JLabel`'s Font (cont'd.)

```
import javax.swing.*;
import java.awt.*;
public class JFrame4
{
   public static void main(String[] args)
   {
      final int FRAME_WIDTH = 250;
      final int FRAME_HEIGHT = 100;
      Font headlineFont = new Font("Arial", Font.BOLD, 36);
      JFrame aFrame = new JFrame("Fourth frame");
      aFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
      aFrame.setVisible(true);
      aFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JLabel greeting = new JLabel("Good day");
      greeting.setFont(headlineFont);
      aFrame.add(greeting);
   }
}
```

Figure 14-9 The JFrame4 program

Adding multiple components

```
import javax.swing.*;
import java.awt.*;
public class JFrame5
{
   public static void main(String[] args)
   {
      final int FRAME_WIDTH = 250;
      final int FRAME_HEIGHT = 100;
      JFrame aFrame = new JFrame("Fifth frame");
      aFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
      aFrame.setVisible(true);
      aFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JLabel greeting = new JLabel("Hello");
      JLabel greeting2 = new JLabel("Who are you?");
      aFrame.add(greeting);
      aFrame.add(greeting2);
   }
}
```

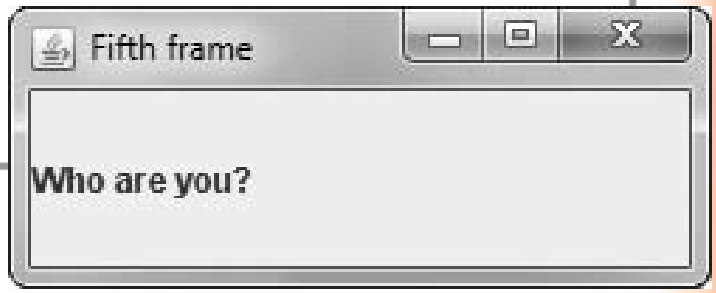


Figure 14-11 The JFrame5 program

Using a Layout Manager

- Layout manager
  - Class that controls component positioning
- `BorderLayout`
  - Normal (default) behavior of a `JFrame`
  - Divides a container into regions
  - if the region was not specified all the components are placed in the same region, and they hide each other.

Using a Layout Manager

- setLayout() method to set the JFrame layout.
  - e.g: aFrame.setLayout(flow);
- `FlowLayout` class
  - Places components in a row and when a row is filled components automatically spill into the next row.
- There are 3 positions in each row, defined using the following constants :
  - FlowLayout.LEFT, FlowLayout.RIGHT, and FlowLayout.CENTER.
  - e.g: FlowLayout flow = new FlowLayout(FlowLayout.RIGHT);

```
import javax.swing.*;
import java.awt.*;
public class JFrame6
{
   public static void main(String[] args)
   {
      final int FRAME_WIDTH = 250;
      final int FRAME_HEIGHT = 100;
      JFrame aFrame = new JFrame("Sixth frame");
      aFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
      aFrame.setVisible(true);
      aFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      JLabel greeting = new JLabel("Hello");
      JLabel greeting2 = new JLabel("Who are you?");
      aFrame.setLayout(new FlowLayout());
      aFrame.add(greeting);
      aFrame.add(greeting2);
   }
}
```

Figure 14-13 The JFrame6 program

Adding `JTextFields`

- `JTextField`
  - Component into which a user can type a single line of text data
  - Several constructors
  - Methods
    - `setText()//` change the text in a JTextField.
    - `getText()//` retrieve the String of text in a JTextField.
    - `setEditable()//`allows entering data in a JTextField.

JTextFields Constructors

- JTextField(); // constructs a new JTextField.
- JTextField(int columns); //constructs a new, empty JTextField with a specified number of columns.
- JTextField(String text); //constructs a new JTextField initialized with the specified text.
- JTextField(String text, int columns) ;//constructs a new JTextField initialized with the specified text and columns.

Adding `JButtons`

- `JButton`
  - Click with a mouse to make a selection
  - Constructors : slides
  - Methods
    - `setText()`
    - `getText()`
- `add()` method
  - Adds a `JButton` to a `Jframe`
- When clicked
  - No resulting actions occur
  - Code has not yet been written to handle user-initiated events

`Jbuttons` Constructors

- `JButton();` creates a button with no set text.
- `JButton(Icon icon);` creates a button with an icon of type Icon or ImageIcon.
- `JButton(String text);` creates a button with text.
- `JButton(String text, Icon icon);` creates a button with initial text and an icon of type Icon or ImageIcon.

```
import javax.swing.*;
import java.awt.*;
public class JFrame7
{
public static void main(String[] args)
{
final int FRAME_WIDTH = 250;
final int FRAME_HEIGHT = 100;
JFrame aFrame = new JFrame("Sixth frame");
aFrame.setSize(FRAME_WIDTH, FRAME_HEIGHT);
aFrame.setVisible(true);
aFrame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
JLabel heading = new JLabel("This frame has many components");
heading.setFont(new Font("Arial", Font.BOLD, 16));
JLabel namePrompt = new JLabel("Enter your name:");
JTextField nameField = new JTextField(12);
JButton button = new JButton("Click to continue");
aFrame.setLayout(new FlowLayout());
aFrame.add(heading);
aFrame.add(namePrompt);
aFrame.add(nameField);
aFrame.add(button);
}
}
```

End

Extending the `JFrame` Class

- Create class that descends from `JFrame` class
- Advantage
  - Set `JFrame`'s properties within object's constructor
  - When `JFrame` child object created
    - Automatically endowed with specified features
- Create child class using keyword `extends`
- Call parent class's constructor method
  - Using keyword `super`

Extending the `JFrame` Class (cont'd.)

```
import javax.swing.*;
public class JMyFrame extends JFrame
{
   final int WIDTH = 200;
   final int HEIGHT = 120;
   public JMyFrame()
   {
      super("My frame");
      setSize(WIDTH, HEIGHT);
      setVisible(true);
      setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
   }
}
```

Figure 14-15 The JMyFrame class

```
public class CreateTwoJMyFrameObjects
{
    public static void main(String[] args)
    {
        JMyFrame myFrame = new JMyFrame();
        JMyFrame mySecondFrame = new JMyFrame();
    }
}
```

Figure 14-16 The CreateTwoJMyFram

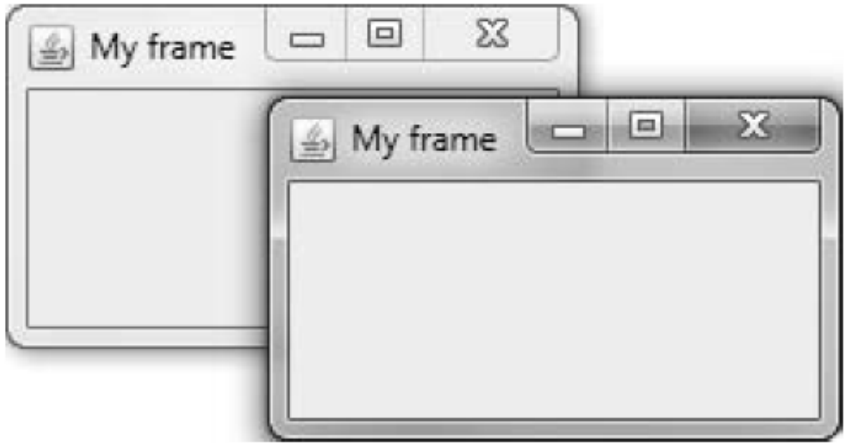


Figure 14-17 Output of the CreateTwoJMyFrameObjects application after dragging the top frame