

Swinging Into Swing

Leo S. Primero III

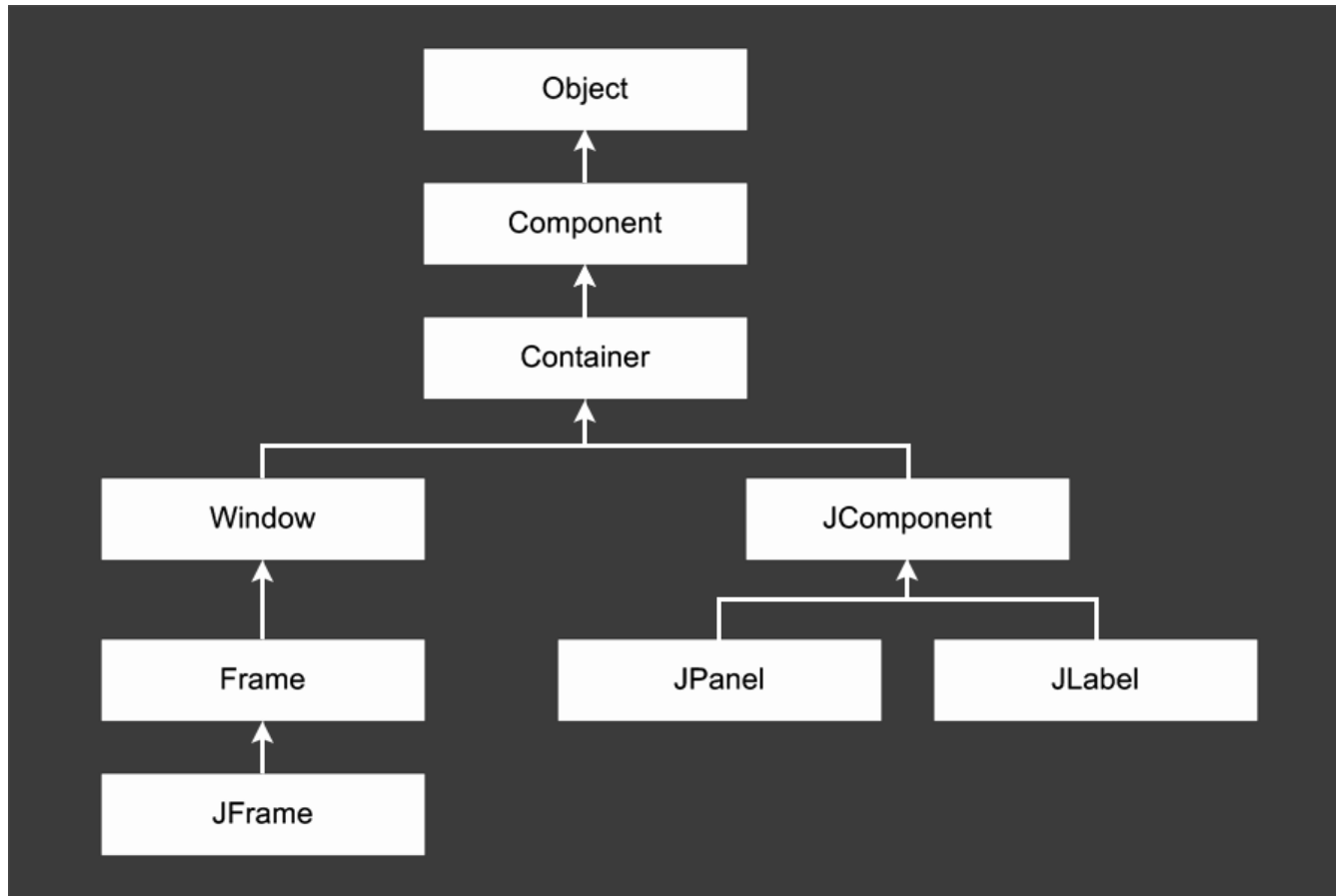
Understanding what Swing Is

- Swing is a package that lets you create applications that use a flashy Graphical User Interface (or GUI) instead of a dull console interface.
- The Swing API provides many different classes for creating various types of user interface elements.

Understanding what Swing Is (Cont' d)

- Three classes: JFrame, JPanel, and JLabel.
These classes are part of a larger collection of classes that are all related through inheritance.
- The Swing family tree splits at the Component class into one group of classes that are derived from the JComponent class, and another branch that descends from the Window class.

The Swing Class Hierarchy



Description of Classes

- Object: All classes ultimately derive from Object, thus this class is at the top of the tree.
- Component: represents an object that has a visual representation that can be shown on-screen and that can interact with users. This class defines some basic methods that are available to all Swing classes.

Description of Classes (Cont' d)

- Container: builds on the basic visual capabilities of the Component class by adding the ability to hold other containers.
- Window: a specialized type of container object that has a border, a title bar, buttons that minimize, maximize, and close the window, and that can be repositioned and possibly even resized by the user.

Description of Classes (Cont' d)

- Frame: a type of Window that serves as the basis for Java GUI applications. Frame is an AWT class that has been improved upon by the JFrame class.
- JFrame: the Swing version of the older Frame class. Most of the Swing applications include at least one JFrame object.
- JComponent: is the basis for all other Swing components except for frames.

Description of Classes (Cont' d)

- JPanel: used to organize and control the layout of other components such as labels, buttons, text fields, etc. In most Swing applications, one or more panels are added to a frame. Then, when the frame is displayed, the components that were added to its panels are made visible.
- JLabel: creates a label that displays a simple text value.

Useful JFrame Constructors and Methods

Constructor	Description
<code>JFrame ()</code>	Creates a new frame with no title.
<code>JFrame (String title)</code>	Creates a new frame with the specified title.
Method	Description
<code>void add (Component c)</code>	Adds the specified component to the frame.

Useful JFrame Constructors and Methods (Cont' d)

Method	Description
<code>JMenuBar getJMenuBar ()</code>	Gets the menu for this frame.
<code>void pack ()</code>	Adjusts the size of the frame to fit the components added to it.
<code>void remove (Component c)</code>	Removes the specified component from the frame.

Useful JFrame Constructors and Methods (Cont' d)

Method	Description
<code>void remove (Component c)</code>	Removes the specified component from the frame.
<code>void setDefaultCloseOperation</code>	Sets the action taken when the user closes the frame. Always specify JFrame.EXIT ON CLOSE.

Useful JFrame Constructors and Methods (Cont' d)

Method	Description
<code>void setIconImage (Icon image)</code>	Sets the icon displayed when the frame is minimized.
<code>void setLayout (LayoutManager layout)</code>	Sets the layout manager used to control how components are arranged when the frame is displayed. The default is the BorderLayout manager.

Useful JFrame Constructors and Methods (Cont' d)

Method	Description
<code>void setLocation (int x, int y)</code>	Sets the x and y position of the frame on-screen. The top-left corner of the screen is 0, 0.
<code>void setLocationRelativeTo (Component c)</code>	Centers the frame on-screen if the parameter is null.

Useful JFrame Constructors and Methods (Cont' d)

Method	Description
<code>void setResizable (boolean value)</code>	Sets whether or not the size of the frame can be changed by the user. The default setting is true (the frame can be resized).

Useful JFrame Constructors and Methods (Cont' d)

Method	Description
<code>void setSize (int width, int height)</code>	Sets the size of the frame to the specified width and height.
<code>void setJMenuBar (JMenuBarMenu)</code>	Sets the menu for this frame.

Using the JPanel Class

- A panel is a type of container that's designed to hold a group of components so they can be displayed on a frame. The normal way to display a group of controls such as text fields, labels, buttons, and other GUI widgets is to add those controls to a panel, and then add the panel to the frame.
- You can bypass the panel and add the controls directly to the frame if you want, but using a separate panel to hold the frames control is almost always a good idea.

Useful JPanel Constructors and Methods

Constructor	Description
<code>JPanel ()</code>	Creates a new panel.
<code>JPanel (boolean isDoubleBuffered)</code>	Creates a new panel. If the parameter is true, the panel uses a technique called double-buffering.

Useful JPanel Constructors and Methods (Cont' d)

Constructor	Description
<code>JPanel (LayoutManager layout)</code>	Creates a new panel with the specified layout manager. The default layout manager is <code>FlowLayout</code> .

Useful JPanel Constructors and Methods (Cont' d)

Method	Description
<code>void add (Component c)</code>	Adds the specified component to the panel.
<code>void remove (Component c)</code>	Removes the specified component from the panel.

Useful JPanel Constructors and Methods (Cont' d)

Method	Description
<code>void setLayout (LayoutManager layout)</code>	Sets the layout manager used to control how components are arranged when the panel is displayed. The default is the FlowLayout manager.

Useful JPanel Constructors and Methods (Cont' d)

Method	Description
<code>void setLocation (int x, int y)</code>	Sets the x and y position of the frame-screen. The top-left corner of the screen is 0, 0.

Useful JPanel Constructors and Methods (Cont' d)

Method	Description
<code>void setSize (int width, int height)</code>	Sets the size of the frame to the specified width and height.
<code>void setToolTipText (String text)</code>	Sets the tooltip text that's displayed if the user rests the mouse over an empty part of the panel.

Using Labels

- A label is a component that simply displays text. Labels are used for a variety of purposes: to display captions for other controls such as text fields or combo boxes, to display informational messages, or to show the results of a calculation or a database lookup.

Using Labels

- A label can also display an image, or it can display both an image and some text. And you have complete control over the appearance of the text.
- You can specify the font, size, whether the text is bold, italic, or underlined, what color the text is displayed as, and so on.

Useful JLabels Constructors and Methods

Constructor	Description
<code>JLabel ()</code>	Creates a new label with no initial text.
Method	Description
<code>String getText ()</code>	Returns the text displayed by the label.
<code>void setText (String text)</code>	Sets the text displayed by the label.

Useful JLabels Constructors and Methods (Cont' d)

Method	Description
<code>void setToolTipText (String text)</code>	Sets the tooltip text that's displayed if the user rests the mouse over the label for a few moments.
<code>void setVisible (boolean value)</code>	Shows or hides the label.

Creating Buttons

- Next to labels, the Swing component used most is the JButton component which creates a button the user can click.
- The constructors of the JButton class are similar to the constructors for the JLabel class. You can either create an empty button or a button with text.

Useful JPanels Constructors and Methods

Constructor	Description
<code>JButton ()</code>	Creates a new button with no initial text.
<code>JButton (String text)</code>	Creates a new button with the specified text.

Useful JPanels Constructors and Methods (Cont' d)

Method	Description
<code>doClick ()</code>	Triggers an action event for the button as if the user clicked it.
<code>String getText ()</code>	Returns the text displayed by the button.

Useful JPanels Constructors and Methods (Cont' d)

Method	Description
<code>void setBorderPainted (boolean value)</code>	Shows or hides the button's border. The default setting is true (the border is shown).
<code>void setContentAreaFilled (boolean value)</code>	Specifies whether or not the button's background should be filled or left empty. The default setting is true (the back-ground is filled in).

Useful JPanels Constructors and Methods (Cont' d)

Method	Description
<code>void setContentAreaFilled (boolean value)</code>	Specifies whether or not the button's background should be filled or left empty. The default setting is true (the back-ground is filled in).
<code>void setEnabled (boolean value)</code>	Enables or disables the button. The default setting is true (enabled).

Useful JPanels Constructors and Methods (Cont' d)

Method	Description
<code>void setRolloverEnabled (boolean value)</code>	Enables or disables the rollover effect, which causes the border to get thicker when the mouse moves over the button. The default setting is true (rollover effect enabled).

Useful JPanels Constructors and Methods (Cont' d)

Method	Description
<code>void setText (String text)</code>	Sets the text displayed by the button.
<code>void setToolTipText (String text)</code>	Sets the tooltip text that's displayed if the user lets the mouse rest over the button.
<code>void setVisible (boolean value)</code>	Shows or hides the button. The default setting is true (the button is visible).

A Word on the Layout of Components

- The layout of components on a panel (or frame) is controlled by a layout manager, which determines the final placement of each component.
- The layout manager takes the size of the component, the size of the panel, and the position of other nearby components into account when it makes its decisions.

A Word on the Layout of Components (Cont' d)

- Swing provides seven different layout managers to choose from. Each has its own way of deciding where each component goes.
- The default layout manager for panels is called FlowLayout. It places components one after another in a row, and starts a new row only when it gets to the end of the panel (or the frame that contains it).

A Word on the Layout of Components (Cont' d)

- With FlowLayout (and with the other layout managers too), the layout changes if the user changes the size of the frame. The size of the frame makes a big difference in how FlowLayout arranges controls.
- You can always call the frame's `setResizable(false)` method to prevent the user from resizing the frame.

A Word on the Layout of Components (Cont' d)

- For many (if not most) Swing applications, one can use more than one panel to display the components with each panel having a different layout manager. With this technique, one can create complex layouts with lots of components arranged in any way wanted.

A Word on the Layout of Components (Cont' d)

- If needed, one can always turn off the layout manager altogether. To do that, call the panel's `setLayout` method with `null` set as the parameter. Then, use absolute positioning, which allows setting the x and y position and the size of each component by calling its `setBounds` method.

A Word on the Layout of Components (Cont' d)

- Controlling the layout of components on a panel is one of the hardest things about using Swing. But following the key points outlined above will make life of a Java developer more efficient.

References

Deitel, H.M., Dietel, P.J. Java How To Program
7th Ed. Upper Saddle River: Pearson, 2007.

Lowe, Doug. Java All-In-One Desk Reference for
Dummies. Hoboken: Wiley, 2005.

McGrath, Mike. Java In Easy Steps. New York:
Barnes & Nobles, 2005.