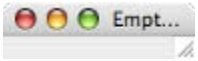
SWING BASICS IN JAVA

Most programs that people use from day to day involve a graphical user interface, a term which is abbreviated as *GUI* and often pronounced *gooey*. We're going to look at how to build a GUI using a Java library called Swing.

Swing isn't the only Java library for GUIs: There is another prominent one included with Java called AWT (for Abstract Window Toolkit), and some developers instead use SWT, a GUI library that doesn't come with Java. In fact, Swing is built using AWT, and we'll see that Swing programs often manipulate some AWT classes.

Historically, AWT came first. It was an attempt to provide a set of classes that interfaced with the graphical elements provided by each operating system. Thus, if we create a button using AWT, it corresponds to a MacOS X button when the program is run in MacOS X, a Linux button when run under Linux, and a Windows button when run under Windows. This leads to difficulties due to slight differences in behavior under different operating systems. Rather than continue to deal with the resulting headaches, Java's developers created Swing, which implements all the graphical elements entirely in Java.

The simplest GUI



The simplest Swing program, in Figure 24.1, simply displays an empty window. It uses just one Swing class, `JFrame`, which represents a window on the screen.

Figure 24.1: The `EmptyWindow` program.

```
1  import javax.swing.JFrame;
2
3  public class EmptyWindow {
4      public static void main(String[] args) {
5          JFrame frame = new JFrame("Empty Window");
6          frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
7          frame.pack();
8          frame.setVisible(true);
9      }
10 }
```

A `JFrame` has just a few methods worth talking about for now.

```
JFrame(String title)
```

(Constructor) Creates an object representing a window with `title` in its title bar. Note that the window doesn't itself appear on the screen yet; that is done using its`setVisible` method.

```
void setDefaultCloseOperation(int value)
```

Configures how this window behaves when the user clicks on the close-window button in the title bar. For simple programs you will want this to be `JFrame.EXIT_ON_CLOSE`. The default behavior is usually `JFrame.HIDE_ON_CLOSE`, where the window disappears from the screen but the base program continues to execute.

(More complex programs often want specialized behavior, such as showing a confirmation dialog. To do this, they use `JFrame.DO_NOTHING_ON_CLOSE` with this method and also add a `WindowListener` to the `JFrame` with a `windowClosing` method. We'll get to talking about listeners later.)

```
void pack()
```

Resizes this window to accommodate the components within it.

```
void setVisible(boolean value)
```

Configures whether this window is visible on the screen.

```
Container getContentPane()
```

Returns an object into which you can place GUI elements that should appear in the window.

Inserting components

A `Container` is an object used for holding elements which might appear in a graphical user interface. We specify which elements go into it using either of its two methods named`add`.

```
void add(Component what)
```

Inserts `what` into this container.

```
void add(Component what, Object info)
```

Inserts `what` into this container, using `info` as information about where the object should be placed.

The first parameter for each method is a `Component` representing an element that appears in a window. You'll never write `new Component()` to create a `Component` instance. Instead, you'll create `Component`'s subclasses. Right now, we'll look at two subclasses: `JButton` and `JTextField`.

`JButton(String label)`

(Constructor) Creates a button holding the string `label`. A **button** is an area the user can click for an action to occur; you see them most often at the bottom of dialog boxes containing labels such as OK or Cancel. We'll see how to get the buttons to do something in Section 24.4.

`JTextField(int columns)`

(Constructor) Creates a text field, sized to contain `columns` characters. A **text field**is an area where the user can type a line of text.

`String getText()`

Returns the current text inside this field.

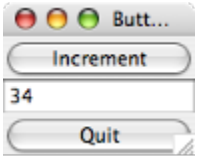
`void setText(String text)`

Changes this field's current text to `text`.

The second `add` method takes a second parameter. This parameter is important: Without it, each component is placed on top of each other in the window's center, and the window isn't very useful. The possible values for this second parameter include five values indicating to place the added component on one of the window's four edges or in the window's center. (Section 24.5.2 will describe more ways to place components within a window.)

`BorderLayout.NORTH`

`BorderLayout.WEST` `BorderLayout.CENTER` `BorderLayout.EAST`



`BorderLayout.SOUTH`

Using `JButton` and `JTextField`, we can build a program that creates a window containing a text field sandwiched by two buttons labeled Increment and Quit. field. We're going to work on making the buttons *do* something soon. For the moment, all the program does is display the components.

Figure 24.2: The `ButtonExample` program.

```
1  import java.awt.BorderLayout;  import java.awt.Container;
2  import javax.swing.JButton;    import javax.swing.JFrame;
3  import javax.swing.JTextField;
4
5  public class ButtonExample {
6      public static void main(String[] args) {
7          JButton incrButton = new JButton("Increment");
8          JButton quitButton = new JButton("Quit");
9          JTextField numberField = new JTextField();
10
11         JFrame frame = new JFrame("Button Example");
12         frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
13         Container contents = frame.getContentPane();
14         contents.add(incrButton, BorderLayout.NORTH);
15         contents.add(numberField, BorderLayout.CENTER);
16         contents.add(quitButton, BorderLayout.SOUTH);
17         frame.pack();
18         frame.setVisible(true);
19     }
20 }
```

This program has two differences from the `EmptyWindow` program from before. In lines 7 to 9 we create objects corresponding to the two buttons and text field. This simply creates objects, but they are not yet part of the window. Lines 13 to 16 adds the components into the window's content pane. Notice that as we add each component, we include a second parameter indicating where to place it within the window.

Source : http://www.toves.org/books/java/ch24-swing/index.html