

Nascent Java: GUI

FWCDS Pre-AP Java Programming

The API Documentation

There are a lot of ways to achieve “windowed” output with Java, but the easiest uses a Java class called `JOptionPane`. This class is used to provide simple I/O.

It is instructive to have a look at the Java API documentation. Go to <http://java.sun.com/j2se/1.4/docs/api/index.html> to find this documentation. Find the entry for the `JOptionPane` class. Java classes are packaged into related groups. Note that `JOptionPane` is in the **package** `javax.swing`.

The next information shows the **inheritance hierarchy** of the class. In this specific example, `JOptionPane` inherits certain data and methods from the class `JComponent`, which in turn inherits from the class `Container`, and so on. More about inheritance later.

Skip down to the “Method Summary.” Here you will find method `showMessageDialog()`. Notice that it is labeled as a **static** method. Recall that this means we can use it without creating an instance of class `JOptionPane`.

Allow me to digress a bit on classes. Think of a class as a blueprint of a container for data and methods that operate on that data. To actually create a usable version of a class (an **object** of that class), you normally must create a **class instance**. However, with static methods, no such instance is needed. If this doesn’t make a lot of sense just yet, hang in there.

Back to our `JOptionPane` method. Notice that there are several varieties of method `showMessageDialog()`. The Java compiler can tell which version is needed by looking at the parameter list. Methods with multiple versions differing only in the nature of their parameter lists are said to be **overloaded**.

A First GUI

Let's write the code for our simple GUI:

```
// MyFirstName MyLastName - 10 May 2002
// HelloGUI.java: a Java application to display
// the traditional message as a GUI

import javax.swing.*;

class HelloGUI
{
    public static void main(String args[])
    {
        JOptionPane.showMessageDialog(null, "Hello World!");
        System.exit(0);
    }
}
```

The `import` statement makes all classes in the `javax.swing` package, including `JOptionPane`, available for our use. **Swing** is the newest group of Java GUI classes.

Notice how the static method is invoked: the name of the class (`JOptionPane`) followed by a dot and then the name of the method (`showMessageDialog()`).

The first parameter to the method is the name of the **component** (like a window) that is to contain the dialog box. Our box will be an orphan without a parent, so we write the keyword `null` here. The second parameter is the object to be displayed in the box, in our case an object of **type** `String`.

Our code has a second statement. Class `System` contains a number of useful methods (look at the API docs), including the static method `exit()`. This method halts the **Java Virtual Machine**, which is the software that takes the metacode file (`*.class`) and interprets it into native machine code for execution on the host platform. The method takes a single parameter of type `int` (**integer**) indicating the cause of the exit. By convention, a zero indicates a normal end of application.

Drop to a command prompt and compile/run this Java application.

Learn by Playing Around

Experiment with the `JOptionPane` class using the API docs as a guide. Start by using the static method `showConfirmDialog()` in lieu of `showMessageDialog()`. Play around a bit.

Next we'll have a look at a full-fledged GUI that can take user input, process it, and display a result.