1.9 (GUI) Displaying Text in a Message Dialog Box

JOptionPane

showMessageDialog

The program in Listing 1.1 displays the text on the console, as shown in Figure 1.12. You can rewrite the program to display the text in a message dialog box. To do so, you need to use the `showMessageDialog` method in the `JOptionPane` class. `JOptionPane` is one of the many predefined classes in the Java system that you can reuse rather than "reinventing the wheel." You can use the `showMessageDialog` method to display any text in a message dialog box, as shown in Figure 1.13. The new program is given in Listing 1.4.

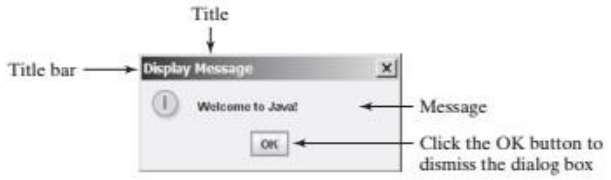

FIGURE 1.13 "Welcome to Java!" is displayed in a message box.

LISTING 1.4 WelcomeInMessageDialogBox.java

block comment

import

main method

display message

```
1 /*  This application program displays Welcome to Java!
2  *  in a message dialog box.
3  */
4 import javax.swing.JOptionPane;
5
6 public class WelcomeInMessageDialogBox {
7   public static void main(String[] args) {
8     // Display Welcome to Java! in a message dialog box
9     JOptionPane.showMessageDialog(null, "Welcome to Java!");
10  }
11 }
```

package

This program uses a Java class `JOptionPane` (line 9). Java's predefined classes are grouped into packages. `JOptionPane` is in the `javax.swing` package. `JOptionPane` is imported to the program using the `import` statement in line 4 so that the compiler can locate the class without the full name `javax.swing.JOptionPane`.

Note

If you replace `JOptionPane` on line 9 with `javax.swing.JOptionPane`, you don't need to import it in line 4. `javax.swing.JOptionPane` is the full name for the `JOptionPane` class.

The `showMessageDialog` method is a *static* method. Such a method should be invoked by using the class name followed by a dot operator (`.`) and the method name with arguments. Methods will be introduced in Chapter 5, "Methods." The `showMessageDialog` method can be invoked with two arguments, as shown below.


```
JOptionPane.showMessageDialog(null,
  "Welcome to Java!");
```

The first argument can always be `null`. `null` is a Java keyword that will be fully introduced in Chapter 8, "Objects and Classes." The second argument is a string for text to be displayed.

There are several ways to use the `showMessageDialog` method. For the time being, you need to know only two ways. One is to use a statement, as shown in the example:

two versions of `showMessageDialog`

```
JOptionPane.showMessageDialog(null, x);
```

where `x` is a string for the text to be displayed.

The other is to use a statement like this one:

```
JOptionPane.showMessageDialog(null, x,
  y, JOptionPane.INFORMATION_MESSAGE);
```

where `x` is a string for the text to be displayed, and `y` is a string for the title of the message box. The fourth argument can be `JOptionPane.INFORMATION_MESSAGE`, which causes the icon ( ) to be displayed in the message box, as shown in the following example.

Note

There are two types of `import` statements: *specific import* and *wildcard import*. The *specific import* specifies a single class in the import statement. For example, the following statement imports `JOptionPane` from package `javax.swing`.

specific import

```
import javax.swing.JOptionPane;
```

The *wildcard import* imports all the classes in a package. For example, the following statement imports all classes from package `javax.swing`.

wildcard import

```
import javax.swing.*;
```

The information for the classes in an imported package is not read in at compile time or runtime unless the class is used in the program. The import statement simply tells the compiler where to locate the classes. There is no performance difference between a specific import and a wildcard import declaration.

no performance difference

Note

Recall that you have used the `System` class in the statement `System.out.println("Welcome to Java");` in Listing 1.1. The `System` class is not imported because it is in the `java.lang` package. All the classes in the `java.lang` package are *implicitly* imported in every Java program.

`java.lang`
implicitly imported

LISTING 2.1 ComputeArea.java

```
1 public class ComputeArea {
2   public static void main(String[] args) {
3     double radius; // Declare radius
4     double area; // Declare area
5
6     // Assign a radius
7     radius = 20; // New value is radius
8
9     // Compute area
10    area = radius * radius * 3.14159;
11
12    // Display results


13    System.out.println("The area for the circle of radius " +
14      radius + " is " + area);
15  }
16 }
```