
Using Objects from the Java API

In this laboratory you will use a `javax.swing.JOptionPane` object as an alternate way to do Input/Output

Declaring, Creating, and Using a JOptionPane Object

The `JOptionPane` class is defined in the package `javax.swing`. Because a `JOptionPane` is used to give us a graphical user interface (GUI) with our program, we will actually be able to see the object when we use it.

When an object is used in a program, it must first be declared and then created. To declare a `JOptionPane` variable, we use a variable declaration statement:

```
JOptionPane myGUI;
```

To create the object we use the `new` operator with a constructor, `JOptionPane()`. This combination constructs, or creates, a new `JOptionPane` object that is then assigned to the `JOptionPane` variable, `myGUI`.

```
myGUI = new JOptionPane();
```

These two lines of code can be combined into a single declaration-initialization statement:

```
JOptionPane myGUI = new JOptionPane();
```

Once an object exists, it can be used. The `JOptionPane` class, contains the method

```
public void showMessageDialog(Component c, String s)
```

which tells us that the `public` method `showMessageDialog` has two formal parameters: `Component c` and `String s`. The return type of the method is `void`, which means that the method returns nothing. To use the method, invoke it on the object in the statement

```
myGUI.showMessageDialog(null, "I love Java!!");
```

Variables that are passed to a method are called the *actual parameters*; values that are passed to a method are called *arguments*. The first argument, `null`, is assigned to the formal parameter `Component c`. Simply put, `null` indicates that this `JOptionPane` is not attached to another `Component` or window. The second argument, `"I love Java!!"`, is assigned to the formal parameter `String s`.

Experiment 5.1

Step 1. Compile and execute `J05E01.java`. Record the results.

```
import javax.swing.*;
class J05E01
{
    public static void main(String[] args)
    {
        JOptionPane myGUI = new JOptionPane();
    }
}
```

Step 2. Modify the program in Step 1 by adding this line of code to the `main` method.

```
myGUI.showMessageDialog(null, "I love Java!!");
```

Compile and execute the modified program. Describe what appears. Record what happens when the button is clicked.

Step 3. Modify the string to be printed, changing it to `"I\nlove\nJava!!!"`. Compile and execute the program. Record the results.

Step 4. Remove the statement `import javax.*;` from the program. How does the compiler inform you of this error?

Experiment 5.2

The `JOptionPane` class also defines a method that allows the user to input information. The method header

```
public String showInputDialog(String prompt)
```

tells us that, when called, the method must be passed a `String`. The returned `String` can be saved, for future use, by assigning it to a `String` variable.

Step 1. Compile and execute the program `J05E02.java`. Describe what appears.

```
import javax.swing.*;
class J05E02
{
    public static void main(String[] args)
    {
        JOptionPane myGUI = new JOptionPane();
        String answer, greeting;
        answer = myGUI.showInputDialog("Enter your name");
        greeting = "Hi " + answer;
        myGUI.showMessageDialog(null, greeting);
    }
}
```

Step 2. Enter your name and click the OK button. Record the results.

Step 3. Execute the program again, this time respond by just clicking the Cancel button. Record the results.

Step 4. Execute the program again, this time respond by entering your name and then clicking the Cancel button. Record the results.

Step 7. Execute the program, entering a non-integer value at the prompt and clicking the OK button. Record the results.

Step 8. Execute the program, entering nothing at the prompt and clicking the OK button. Record the results.
