

Remember "Reviewing Java" has many more examples and topics.

Tutorial 6 ***JOptionPane for Input/Output***

What is a JOptionPane?

A JOptionPane (spelled as is) is part of the java swing library. It requires an import statement at the top of the program. Here is what the statement is:

```
import javax.swing.JOptionPane;
    OR
import javax.swing.*;
```

Either of those is acceptable. The difference is that the latter will import the entire library as denoted by the star. The first will just import the JOptionPane.

The pane will allow a user to do a few things. The user can enter data, display messages or a combination of those.

Wrapper Classes

A JOptionPane can be used to input numbers or strings. By default, anything entered in a JOptionPane is stored as a string in Java. To fix this, we need what is called a **wrapper class**. This allows us to convert the string into the desired number data type (int, float, double, long or short). Here are the wrapper class names:

```
Integer
    -for the int type
Double
    -for the double type
Float
    -for the float type
Short
    -for the short type
Long
    -for the long type
Character
    -for the char type
Boolean
    -for the boolean type
Void
    -for the void type
Byte
    -for the byte type
```

Remember "Reviewing Java" has many more examples and topics.

A very important thing to observe is that the names of these wrapper classes are all capital letters. This is how they are differentiated from some data types.

With the exception of the Character class, each of the above has a method called parseXXX where the XXX is the primitive type name (int, float, double, short, long or boolean). This method will actually change the values into the correct type. Let's see a code snippet:

```
int x = Integer.parseInt(args[0]);  
  
double d = Double.parseDouble(args[1]);  
  
etc...
```

Input

To use a JOptionPane for input, you use the following method:

```
JOptionPane.showInputDialog(null, "[message]");
```

where in the above, JOptionPane is required to access the methods, null is a keyword representing nothing, and the [message] is the appropriate message that will be displayed in the JOptionPane dialog window. The above will be used inside of a parseXXX method if you are trying to input a numeric type.

Here is an example showing a dialog box for input and then displaying the results to the console. The below program will ask for a number and then output it's triple:

Example 1: JOptionPane input example

```
import javax.swing.*;  
  
public class InputExample{  
  
    public static void main(String args[]){  
  
        int x = Integer.parseInt(  
            JOptionPane.showInputDialog(  
                null, "Enter any number: "));  
  
        System.out.println("Here is the triple: " + x*3);  
    }  
}
```

Remember "Reviewing Java" has many more examples and topics.

No command line arguments are needed as the input is coming from the input dialog. The output goes to the console via the `println` statement.

But what if we wanted the output to be done strictly on `JOptionPane`s? We can do that as well.

Output

To do output on a `JOptionPane`, use the following statement:

```
JOptionPane.showMessageDialog(null, "[message]");
```

This time, there is no data entry being performed. It simply displays a message on the window.

Here is a modification of the above program to show the output to a `JOptionPane` message dialog:

Example 2: JOptionPane output example

```
import javax.swing.*;

public class OutputExample{

    public static void main(String args[]){

        int x = Integer.parseInt(
            JOptionPane.showInputDialog(
                null, "Enter any number: "));

        JOptionPane.showMessageDialog(null,
            "The triple is: " + x*3);
    }
}
```

Instead of concatenating strings in the method, you could have used a `String` variable to display the message. Either way is acceptable.