Using JOptionPanes for graphical communication with our programs.

The + String operation

Using + to join `Strings` to other things is most often seen in printing, as in:

```
System.out.println("Welcome " + myName + " to Java");
```

You don't have to join only strings to other strings:

```
int myNumber = 1;
System.out.println("you are number " + myNumber + "!!!");
```

And you can also use this for variables if you want:

```
String myFirstName = "Fintan";
String mySecondName = "Costello";
String myFullName = myFirstName + " " + mySecondName;
```


Communicating with our programs

We've already seen one way our programs can communicate with us:

`System.out.print("…")` (or `println("…")`)

It would be good if **we** could communicate with our programs: if they could ask for values to put in variables, and we could answer.

For example, in our "Welcome to java" program we had to type in the `myName` value as part of the program. It would be better if the program asked for the users name and gave a greeting using that value.

Communicating using a pop-up window (a 'JOptionPane')

An "OptionPane" is a small window that pops up to ask us a question or give us some information

Java lets us use OptionPanes in our programs: they are called

'`JOptionPane`' (the `J` at the beginning stands for Java)

Drawing and running little pop-up windows is a complicated process

Our program has to 'import' some extra help to do it

Program using JOptionPanes to say Hello to the user in a pop-up window

```
/* this program creates a pop-up JOptionPane to ask your name
and say Hello.  It imports 'javax.swing.JOptionPane' to allow us
use JOptionPane methods                                    */

import javax.swing.JOptionPane;

public class Hello
{
 public static void main(String[] args)
 {
  String name;

  name = JOptionPane.showInputDialog(null, "What's your name?");

  JOptionPane.showMessageDialog(null, "Hello, " + name);

  System.exit(0);
 }
}
```

comments

class name

main part of class

declare a `String` var

Put something in the variable `name` (what?)

New components of this program

```
import javax.swing.JOptionPane;

public class Hello
{
 public static void main(String[] args)
 {
  String name;

  name=JOptionPane.showInputDialog(null,"What's your name?");

  JOptionPane.showMessageDialog(null, "Hello, " + name);
  System.exit(0);
}
}
```

Import the class that let us give instructions to `JOptionPanes`

Ask the users name using an `InputDialog` window, and put the answer in variable `name`.

Create a `MessageDialog` window showing the message "`Hello`" followed by the users name (in the `name` variable)

Tell the `System` that this program is `exit`ing (finishing), so that it can shut down the Dialog windows properly.

What does `import javax.swing.JOptionPane;` mean?

Java provides a number of extra "packages" to allow us to do complicated things (such as drawing windows on the screen). These packages are java code written by expert programmers. Some of these packages are named `javax` (`java` extras).

One of these packages is for doing things with all sorts of windows. This package is called `swing`. In the `swing` package there is a class (a program) for doing things with OptionPanes: the `JOptionPane` class.

When we put the command `import javax.swing.JOptionPane;` at the start of our program, this tells the java compiler to include the code for the `JOptionPane` class in our program. This lets us "call" the `JOptionPane` program and tell it to do things with `OptionPanes`.

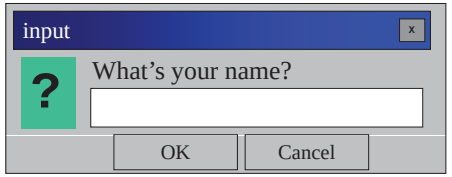
What does `JOptionPane.showInputDialog(null,"What's your name?");` mean?

```
JOptionPane.showInputDialog(null,"What's your name?");
```

This asks the `JOptionPane` program to call its `showInputDialog` method.

The `showInputDialog` method is a series of commands which `show` an `input dialog` to the user (draw a window asking user to input something)

The `showInputDialog` method draws this on the screen:



and waits for the user to type something and click "ok"

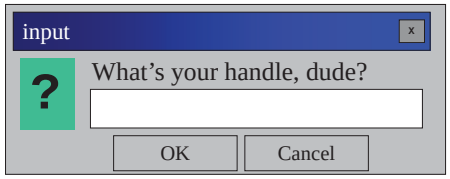
The arguments to `JOptionPane.ShowInputDialog`

When we call a method, we give *arguments*. The method is going to do things (e.g. draw a window): the arguments are what we want it to do things **with.**

```
JOptionPane.showInputDialog(null,"What's your name?");
```

In an input dialog, the most important argument is the question we want to ask the user in the dialog; the question they will give an answer to.

```
JOptionPane.showInputDialog(null,"What's your handle, dude?");
```



If we change that argument, the dialog will ask a different question.

Other arguments to `JOptionPane.ShowInputDialog`

The main argument for ShowInputDialog is the question we're asking the user to respond to.

```
JOptionPane.showInputDialog(null,"What's your name?");
```

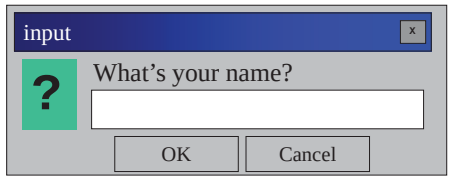
The other argument indicates where we want to put the pop-up window that `ShowInputDialog` will draw. Often we want these windows to pop up inside a "frame" (a larger window doing other things).

In our simple example, we're not using any larger windows, so we just say "null": this means the window will just pop up in the middle of the screen.

```
JOptionPane.showInputDialog(null,"What's your name?");
```

class method frame query message

Where does the answer to `JOptionPane.ShowInputDialog` go?



When someone uses this, they answer the question (type their name) and click ok. Where does their name go? We can see where in the code:

```
String name;
name = JOptionPane.showInputDialog(null,"What's your name?");
```

This means that the answer the user gives to the `showInputDialog` box is placed into the `String` variable `name`.

The `showInputDialog` 'returns' or 'evaluates to' the answer the user types in the dialog box. This is always a `String`, so it must go in a `String` variable.

What about the MessageDialog?

```
String name;
name = JOptionPane.showInputDialog(null,"What's your name?");
```

This asks the `showInputDialog` method of the `JOptionPane` program to make an InputDialog window (not inside any other window) asking "What's your name", and puts the `String` that the user types as answer in the `name` variable.

```
JOptionPane.showMessageDialog(null, "Hello, " + name);
```

What does this line do? Asks the `showMessageDialog` method of `JOptionPane` to make a window (not inside any other window), saying `"Hello, "+name`. No answer is returned, so no variable assignment is needed.

And finally: System.exit(0)

`System.exit(0);` is the last line in our program. What does it do?

This line sends a message to the computer's operating `System`, telling it that this program (the `Hello` program) is `exit`ing (that is, shutting down).

The exit value `0` tells the `System`, that the program ran without errors.

When `System` gets this exit message, it makes sure that the `JOptionPane` program (which was called by our `Hello` program) is also shut down.

The `System.exit(0);` line is like a politeness to the operating system program, telling it that the `Hello` program is finished using the computer now, thanks very much.

In future, you should **end all your programs** with `System.exit(0);`
This will be especially important if other programs are calling your programs: the `System.exit(0);` tells other programs when your program is finished.

An online tutorial

You can see a (very detailed, perhaps too advanced) tutorial on optionpanes and other popup windows for communicating with java programs here:

http://download.oracle.com/javase/tutorial/uiswing/components/dialog.html

(apologies for the pale blue display: one of the quirks of powerpoint).

This tutorial comes from Oracle, the makers of java: they have a number of tutorials on other topics in java programming also. All are slightly advanced, but very well worth looking at.