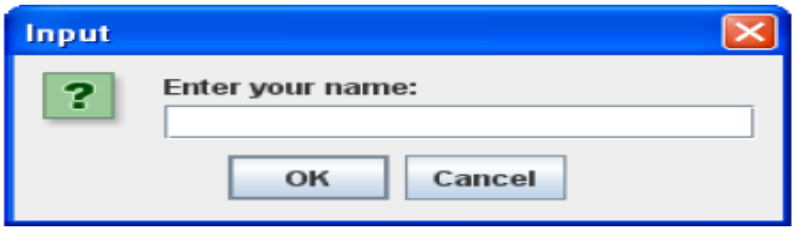
GRAPHICAL INPUT

 A GUI uses windows and point-and-click interaction rather than keyboard input.
 A GUI uses event driven programming to control the interaction.

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

 Another useful class for accepting user input, and displaying results, is the JOptionPane class. This is located in the *javax.swing* library. The JOptionPane class allows you to have input boxes like this one



- 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)
- Across many applications we will want have dialogs that convey a certain message.
- We may want to:
  - Warn the user – use the Warning Dialog
  - Report an error – use the Error Dialog
  - Ask a simple question (Yes/No query) -- Question Dialog
  - Simply Inform the user -- Information Dialog o
  - Pass a plain message form -- Message Dialog
- The JOptionPane dialog offers a fairly consistent (look-and-feel dependent) set of functional dialogs

Method	**Meaning**
String showInputDialog	Prompts for input and returns a String
void showMessageDialog	Displays a message and waits for acknowledgment
int showConfirmDialog	Asks a confirming (Yes/No) question and returns the response

Note:

1. These methods are class methods, so they are applied to JOptionPane, not to an instance of JOptionPane.
2. Each dialog possess up to 4 components as listed below:

Icon area -displays an icon.

Button area -populated with one or more buttons (Yes, No, OK, Cancel, and so on) according to an option type specified by the developer

Message area -shows a simple text string as a message.

Input area

- Content is controlled by the setWantsInput() method allowing the presence of a data entry field that requires the user to enter some text.
- There are four dialog types which can be further enhanced by by a messageType parameter in the show...Dialog() methods.
- They each have different functionality and should be used in the most applicable fashion only. They are characterised by a slightly different appearance -- buttons available, input possible etc.

showMessageDialog

- The showMessageDialog method has two versions, one with two parameters and one with four parameters.
- showMessageDialog is a void method whose sole responsibility is to display a message on the screen in a dialog box.
- We look first at the case with two parameters. The first parameter is again a parentComponent, so we simply write null there. The second parameter is the string that we wish to display as a message. For example:

JOptionPane.showMessageDialog(null, "Click OK when ready to begin entering data.");

And message boxes like this:

Example

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

public class MyNewApp

{

public static void main( String [] args )

{

String prompt, name, out;

prompt = "What's your name?";

name = showInputDialog( prompt );

out = "Welcome to Java Programming, " + name + "!";

showMessageDialog( null, out );

}
```

Example

The program below requests user to enter first and family name using the JoptionPane class. The program then outputs the names on the screen

```
package inputboxExample;

// reference the library we want to use:
 import javax.swing.JOptionPane;
public class InputBoxexample {

public static void main(String[] args) {


//get first name using inputbox


 String first_name;


first_name = JOptionPane.showInputDialog("First Name");

//get last name using inputbox


String last_name;


last _name = JOptionPane.showInputDialog("Last Name");

//Join the two together, and add some text: String
full_name;


String full_name = "Your name is " + first_name + " " + last _name;

//To display the result in a message box, add the following:


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

// **System.exit(0)** ensures that the programme exits. Also tidies up for us, removing all the created objects from memory.

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

Exercise

Input boxes and Message boxes can be formatted further. Try the following for your Input boxes:

JOptionPane.showInputDialog("First Name", "Enter Your First Name");

JOptionPane.showInputDialog("Last", "Enter Your Last Name");

Exercise

For your Message boxes try this:

JOptionPane. showMessageDialog(null, full_name, "Name", OptionPane.INFORMATION_MESSAGE);

Exercise

Instead of JOptionPane.INFORMATION_MESSAGE try these:

JOptionPane.**ERROR_MESSAGE**

JOptionPane.**PLAIN_MESSAGE**

JOptionPane.**QUESTION_MESSAGE**

JOptionPane.**WARNING_MESSAGE**

Example

Input boxes are not just used for text: they can accept numbers as well. The following program creates an input dialog, translates the returned string to an integer, adds 10 to the integer, and displays the result in a message dialog. Note the import statement at the beginning.

```
import javax.swing.*;

public class DemoDialog{

public static void main(String args[]){

String inputValue =JOptionPane.showInputDialog(null, "Please input an integer value");

int number = Integer.parseInt(inputValue) + 10;

JOptionPane.showMessageDialog(null, "Input plus 10 is " + number);

JOptionPane.showConfirmDialog( null, "Are you sure you want to Exit" );

} }
```

The second version of method showMessageDialog takes four parameters, the parentComponent, the text within the box, a text label for the box, and the messageType. Here are four examples and the boxes they produce.

```
JOptionPane.showMessageDialog(null, "Plain", "MessageType",
JOptionPane.PLAIN_MESSAGE);

JOptionPane.showMessageDialog(null, "Question", "Message Type",
JOptionPane.QUESTION_MESSAGE);

JOptionPane.showMessageDialog(null, "Warning", "Message Type",

JOptionPane.WARNING_MESSAGE);

JOptionPane.showMessageDialog(null, "Error", "Message Type",
JOptionPane.ERROR_MESSAGE);
```

showInputDialog

- The showInputDialog method creates a dialog box that asks the user to input information and press an OK button.
- This method is a value-returning method, returning the string that the user has input.
- We look at three versions of the method distinguished by their parameter lists.
    i. The first version just takes a String parameter.

    ```
    String inputValue = JOptionPane.showInputDialog("Please enter a value");
    ```

ii. A second version of method showInputDialog has one additional parameter parentComponent, which is null in the following example.

```
String inputValue = JOptionPane.showInputDialog(null, "Please input an integer value");
```

A JOptionPane dialog box can be used in conjunction with other windows. The first parameter in this version can be used to specify the window (parentComponent) over which the dialog box should be centered. Because we have not explored the use of other windows, we simply put null there, and the dialog box is then centered on the screen

iii. The third version of this method has four parameters: the parentComponent (null), a String that is the prompt to be written in the box, a String that is used to label the box, and the messageType. Here are the values that describe the messageType, which are static, named, int constants declared in JOptionPane.

The messageType parameter specifies a particular icon to display to the left of the input field. Here is an example of this method call and the dialog box it produces. Note that messageType refers to class constants, so the fourth parameter is the class name, dot, and then the named constant.

```
String inputValue =JOptionPane.showInputDialog(null, // Parent Component (null = the screen)
 "Please input an integer value", // Prompt message in box
 "My Input Pane", // Label for top of box
 JOptionPane.QUESTION_MESSAGE);       // Message Type
```

Exercise

Write a programme that prompts the user for two numbers, the length of a rectangle and the width of a rectangle. Use a message box to calculate the area of the rectangle. (Remember: the area of a rectangle is its length multiplied by the width.)

```
package arearectangle;

import javax.swing.JOptionPane;

public classAreaRectangle {

static void main(String[] args) {

String length;

length = JOptionPane.showInputDialog("Rectangle length");

String width;

width = JOptionPane.showInputDialog("Rectangle width");

//convert string to integer

Integer.parseInt( width );

Integer.parseInt(length );

int area = Integer.parseInt( length ) * Integer.parseInt( width);

//display the result in a message box

JOptionPane.showMessageDialog( null, "The area of the rectangle is  " + area );

System.exit(0);

 }
```

Input Data Validation

We can use a try-catch statement to ensure that correct numeric values are entered through a dialog box. In the first JOptionPane example, we had the user enter a number and the code added a value to the number and displayed the result. Let's enclose that code in a do statement, in which the loop keeps repeating until a valid number is input. If the value that is entered is not an integer value, then parseInt throws a NumberFormatException. The catch clause can alert the user that the number was invalid and prompt for the number again. A boolean variable numberOK is the while expression; it is set to true at the beginning of the try clause and set to false in the catch clause.

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

```
public class DemoTryCatchDialog {
public  static    void main(String args[])         {
String inputValue;
boolean numberOK;
 int number;
 do
 {
inputValue =JOptionPane.showInputDialog(null, "Please input an integer value");
 numberOK = true;
 try
 {
number = Integer.parseInt(inputValue) + 10;
JOptionPane.showMessageDialog(null, "Input plus 10 is " + number);
}
catch(NumberFormatException except) {
JOptionPane.showMessageDialog(null, "You did not enter a valid number");
numberOK = false;
}
} while(!numberOK);
}
}
```

Exercise

1. Write a program that asks user to enter their weight (in kg)/ and height (in metres) using

Inputbox. The program should then calculate the person's Body Mass Index (BMI) using the formula BMI = weight height and output it in MessageDialog;

2. Using the graphical input, write a program that requests user to input name, course, year of study, and marks for 5 subjects undertaken by a student. The program should then calculate and display the total and average.