**Messages and Methods for Window Controls**

Let us begin with a quick review of some basic terminology. Programs manipulate window controls by sending them messages. For instance,

fahrenheit = degreesFahrenheitField.getNumber();

and

degreesCentigradeField.setNumber (centigrade);

Asking a control for information or telling it what to do is called sending the control amessage. Sending a control a message activates in the control a method that accomplishes the desired task. Methods are implemented by Java code. Thus, with respect to the statement

fahrenheit = degreesFahrenheitField.getNumber();

we can state the following:

|  | The getNumber message is sent to the degreesFahrenheitField control. |
| --- | --- |
|  | The degreesFahrenheitField control implements a getNumber method. |
|  | The degreesFahrenheitField control understands the getNumber message. |
|  | The getNumber method returns an integer, which in this example is assigned to the variable fahrenheit. |

Notice also:

|  | The message and the method have the same name. |
| --- | --- |
|  | In response to a message a control may or may not return a value. |
|  | getNumber returns a value, whereas setNumber returns nothing (also called void). |
|  | When a control is sent a message, sometimes additional information is required as part of the message. |
|  | Each piece of additional information is called a ***parameter.*** |
|  | Thus, the variable centigrade is a parameter in the following message: |

degreesCentigradeField.setNumber (centigrade);

|  | In general, several parameters may be included or passed at once, for instance: |
| --- | --- |

someControl.someMethod (parm1, parm2, . . ., parm*n*);

|  | Sometimes no parameters are needed, for instance, |
| --- | --- |

fahrenheit = degreesFahrenheitField.getNumber();

|  | If a method expects parameters, they must always be included and must always appear in the same order. |
| --- | --- |

The following table provides a list of the most useful methods implemented by some basic window controls (we will encounter more controls later):

| **Type of Control** | **Name of Method** | **What the Method Does** |
| --- | --- | --- |
| IntegerField | setNumber(anInteger)  returns void | Displays anInteger on the screen. Nothing is returned. |
|  | getNumber()  returns double | If the number on the screen is a valid integer, then reads the number and returns it. If the number is not a valid integer, then sets it to 0 and returns 0. |
|  | isValid()  returns boolean | Determines if the number entered on the screen is a valid integer. More about Booleans later. |
| DoubleField | setNumber(aDouble)  returns void | Displays aDouble on the screen. Nothing is returned. |
|  | getNumber()  returns double | If the number on the screen is a valid double, then reads the number and returns it. If the number is not a valid double, then sets it to 0 and returns 0. |
|  | isValid()  returns boolean | Determines if the number entered on the screen is a valid double. |
|  | setPrecision(anInteger)  returns void | Sets the control’s precision to the specified number.  When a double is displayed on the screen, the number of digits after the decimal point is equal to the precision.  If the precision is never set or if it is set to -1, then the format of the number on the screen varies and depends on the size of the number; however, at most six digits will be displayed. |
|  | getPrecision()  returns int | Returns the value of the precision |
| Label | getText()  returns String | Returns the text of the label. Such text is called a string. |
|  | setText(aString)  returns void | Sets the text of the label to the specified string |
| Button | setLabel(aString)  returns void | Sets the text written on the button to the specified string |
|  | getLabel()  returns String | Returns the text written on the button. |
| All Window Controls | setVisible(true/false)  returns void | If false then hides the control so that the user can no longer see it on the screen.  If true then makes the control visible again. |
| setEnabled (true/false)  returns void | If false then disable the control for user input.  If true then enable the control for user input.  Default is enabled. |
| requestFocus()  returns void | Moves the focus to the control |