

## t Windows<sup>106</sup>

When control is transferred from the Application to the editor/interpreter environment either by the user or by the interpreter, execution is suspended. When an execution is suspended:

o

The interpreter windows (also called execution windows) are displayed with a dotted background, in contrast to the plain background of editing windows. The execution windows displayed in this environment are the Stack window and case windows. Unless otherwise specified, the term “window” is used in this section to mean “execution window.”

o

Prograph allows the user to start a new execution.<sup>106</sup>

Prograph supports multiple executions. If several executions are active, they all appear stacked in the Stack window in the order in which they were started, separated by horizontal fuzzy lines. The lowest icon in each execution indicates the top of the stack for that execution.

A case window displays the details of a case in the stack. Although editing is not allowed in such windows, Prograph allows the user to alter the status of execution and to open the editing case window from the execution case window to alter the structure of the method.<sup>106</sup>

## Stack Window<sup>107</sup>

This window displays the cases under execution as a stack of named icons, each containing a case number. The stack grows from top to bottom, so the lowest icon in the stack window represents the element at the top of the stack.

The format of the name appearing below the icon depends on whether the case belongs to a universal method, a class method (Plain, Get, Set, or Initialization) or a Local. When a case belongs to a Local, its name appears in plain text; otherwise, it is displayed in bold characters.<sup>107</sup>

rograph supports multiple executions, each execution being separated from the next by a horizontal gray fuzzy line(green on a color monitor) in the Stack window.<sup>107</sup>

## Opening<sup>108</sup>

This window is opened:

- o when Stack is selected from the Windows menu
- o when execution is proceeding in uninterrupted mode and Command- . (period) is pressed.

## Actions<sup>108</sup>

The icons in the Stack window cannot be dragged. However, the following actions are possible:

- o Pressing the Return key resumes execution., and, if in step mode, opens the execution case window of the method on top of the stack (bottom of the Stack window).
- o Double-clicking on any case icon opens its corresponding execution case window.
- o The level of debugging can be set on a selected icon or group of icons. For details about the level of debugging, please refer to the “Menus” section below in this chapter.
- o Command-clicking on an icon causes execution to be rolled back as explained below.

## Rollback<sup>108</sup>

When the user Command-clicks on an icon in the Stack window, execution is undone until the clicked icon is at the top of the stack, that is, until it is the lowest icon in the Stack window. Consequently, if several executions are in progress, all executions following the one containing the clicked icon are discarded. As icons are removed from the Stack window during this process, their execution case windows are closed.<sup>108</sup>

## Case Window<sup>109</sup>

case window shows the state of execution of a method by employing different styles for drawing operations, roots, and terminals. Each case window has one selected operation (drawn in red on a color monitor), which is the one under execution. All the operations that have executed are drawn normally (blue on a color monitor), while all the unexecuted operations are drawn in fuzzy gray (fuzzy green on a color monitor).

## Opening<sup>1109</sup>

A case window opens:

- o when an icon in the Stack window is double-clicked
- o when an error occurs in executing an operation in the case
- o when a Breakpoint set on an operation in the case is encountered during execution
- o when Trace in the Exec menu is checked
- o when Show Cases is turned on for the method

## Actions<sup>1109</sup>

The following actions are possible in the case window:

- o Pressing the Return key causes execution to continue.<sup>1109</sup>
- o Double-clicking on the dotted background opens an editable window for the same case. Changes made in the editable case window are automatically made in the execution case window, either immediately, or, with text editing of names or comments, when some event occurs that signifies the end of the text editing.<sup>1110</sup>

o

Double-clicking on an operation has the same effect as in the Editor.

o

Command-clicking on an operation selects it as the next operation to be executed. This can cause rollback or roll forward (further explained below).<sup>110</sup>

o

Double-clicking on a root of an executed operation opens a Value window. The Value window not only displays the value at the root, but the user can also modify the value, provided that the case is the top stack element.

o

Clicking on a root or terminal of an executed operation and keeping the mouse key depressed displays a popup containing the current value of that root or terminal.

For example, double-clicking on the root Side 2 in the execution case window of the universal method Pythagoras opens its Value window.<sup>110</sup>

## isplaying Values<sup>110</sup>

To display the current value of a root or terminal, single-click (holding the mouse button down) on that root or terminal, as illustrated in the following example:

n the next example, due to the synchro link, Side 2 must execute before Side 1. Yet even though the \* operation for Side 1 has not yet executed, and its terminal is filled in, you can still single-click to examine the value for this terminal, because the input bar HAS executed, and a value for the Side 1 \* operation terminal has arrived (2):<sup>1111</sup>

## pening Value Windows<sup>1111</sup>

To open a Value window for a root or terminal, double-click on the root or terminal. Click-selecting a root or terminal will open its Value window as well. The Value window operates as usual; you can view or edit the displayed data - as long as that operation has already executed. (Note that unexecuted nodes are filled in, not hollow).

To illustrate this, if you double-click on the Side 1 root in the execution case window of the following sample method Pythagoras before the input bar operation has executed, the following error dialog will result:<sup>1111</sup>

owever, if you double-click the same node after executing the input bar operation (note that the node will now be hollow), the Value window for that node will appear:<sup>1112</sup>

## ollback<sup>112</sup>

When an execution is suspended and the user edits a case that is under execution, Prograph undoes the execution of all operations in the case that are affected by the change, but preserves the executed state of other executed operations. <sup>112</sup>

Similarly, execution is rolled back if the value of a root in a case window is changed. For example, suppose that in the universal method Pythagoras, the operation \* getting its values from Side 2 is executed before the other operation \* getting its values from Side 1. <sup>113</sup>

f the value appearing at Side 2 is changed as described above, by opening the Value window when the primitive + is about to be executed, execution is undone, but only in the operation that gets its value from Side 2. The \* operation that gets its value from Side 1, however, remains executed even though it previously had been executed after the \* operation that is undone (see examples above and below). <sup>113</sup>

n general, execution in a case can rollback if: <sup>113</sup>

o

the user Command-clicks on an executed operation

- o

changes made in an editing case window affect executed operations in the window

- o

the user closes a Value window on an executed root or terminal by clicking the OK button.<sup>113</sup>

Since the state of execution within a case window changes with rollback, the case window is redrawn to reflect the change. <sup>114</sup>

## Roll Forward<sup>113</sup>

It is possible for the user to advance the state of execution in a case, avoiding the execution of some operations, by the following means:

- o

Command-clicking on an unexecuted operation selects it as the next to be executed. As a result, the operations that would have had to be executed to provide input values to the clicked operation are put into the executed state, and their roots are given the value UNDEFINED. <sup>113</sup>

- o

Pressing the Tab key places the selected operation in the executed state, giving its roots the value UNDEFINED, and selects the next operation in the normal execution sequence.

Since the state of execution within a case window changes with roll forward, the case window is redrawn to reflect the change.

## Effect of Roll Forward and Rollback on the Stack<sup>113</sup>

Certain actions that cause execution to roll forward or back in a case can also affect the stack.

- o

If the user presses the Tab key or Command-clicks to change the selected operation in a case window, and the method called by that operation is not at the top of the stack, the stack is undone until that method becomes the top stack element.

- o

If changes are made in an editing window for a case which is still on the execution stack but not at the top of the stack, the stack is undone until the earliest copy of this case becomes the top stack element. <sup>113</sup>