



User's Guide

© 1999 All Rights Reserved

interRAD Technology Pty Ltd.

*1/88 Sumners Road, Sumner Park
Queensland, Australia.*

www.interadtechnology.com

Phone 61 7 3279 5551

Fax 61 7 3279 4494

Welcome to build-IT!

Installation

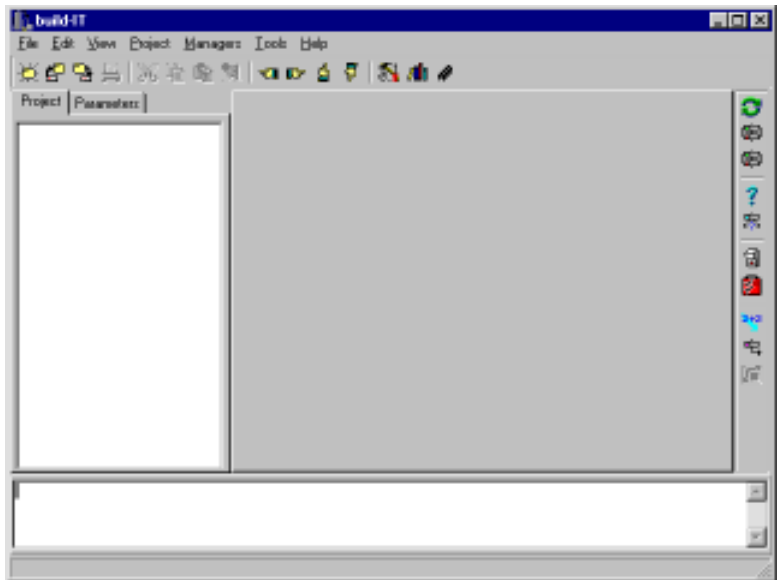
Clicking SETUP.EXE will automatically install *build-IT* on your system.

If you received the program by email, you will need to install Java JDK 1.2 separately. This is available free on the internet from www.java.sun.com. Install the JDK first, then *build-IT*. Be sure to restart your computer after installation.

Buyers who received *build-IT* on CD received the JDK libraries on the CD and these will automatically install with the program.

Once installation is complete, please double-click *build-IT's* pyramid icon on the desktop to start the program.

build-IT's main work screen opens.



The main work screen is divided into three sections:

On the left, a section of the window displays either *Project* information or *Parameter* information.

The large grey area on the right will display your flowchart.

The lower section of the window will display important status messages created by the system.

Click the **Project Tab** to view a structure chart of your current project.

Unlike Layout, *build-IT* allows programmers to manage a complex collection of flowcharts which make up a large application. The structure chart allows you to move between units in your project. When you are done and select Build to make a completed program, all the units will be included in a single completed ready to run application file. There is no longer any need to compile separate “child EXE” programs which you might previously have run from within a main application to conserve memory.

By clicking on any component of your project in the Project Window, you are able to access the flowchart for that section of your application to add to it or make changes.

Click the **Parameters Tab** to view the parameters for the currently selected element.

In Layout, you double-clicked on a flowchart element to “open” it and reveal its parameters. You could then make changes to the parameters for that element. If there were a large number of parameters, this involved scrolling the screen to find the relevant parameters.

In *build-IT*, you can still work this way if you prefer. However, the *Parameters* view on the left of the screen will always display the parameters for the currently

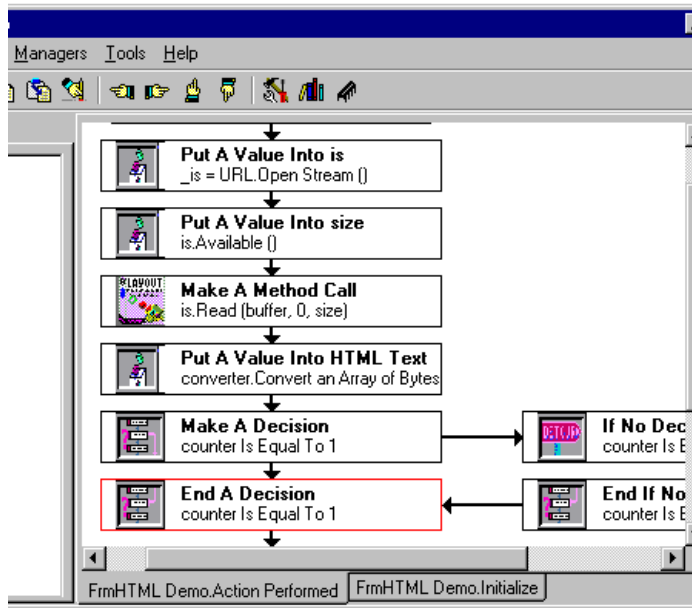
selected flowchart element and you can change these parameters by typing right into the grid, like this:

****This feature is not enabled in the Technology Preview Release***

The Main Flowchart Area

The largest part of the work window is allocated to flowchart construction. This portion of the window scrolls vertically to allow you to work with large flowcharts.

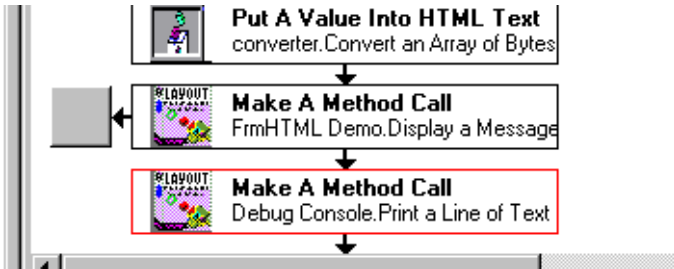
It also scrolls horizontally. This is because the flowchart is now two dimensional. That is, instead of displaying the contents of a decision statement in a vertical list between the *Start* and *End of Decision*, the elements in the decision statement are displayed to the right of the decision and the actions that will occur if a decision containing an *ELSE* is false are displayed to the right of the *Else* element. The horizontal scrolling feature allows you to readily view the elements placed to the right of the main flowchart.



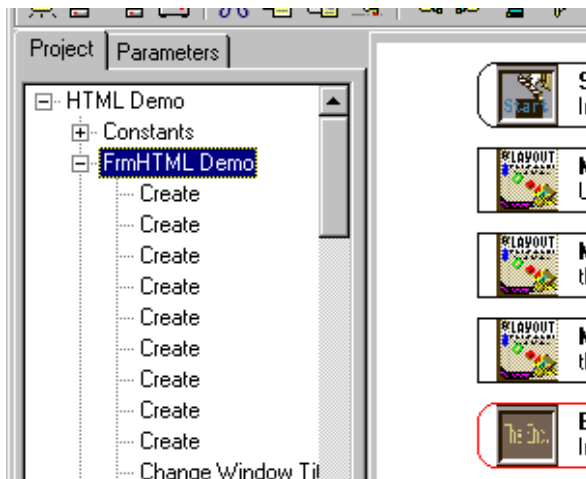
The Tabs

At the bottom of the flowchart display area, a row of tabs displays the name of each procedure in your current flowchart.

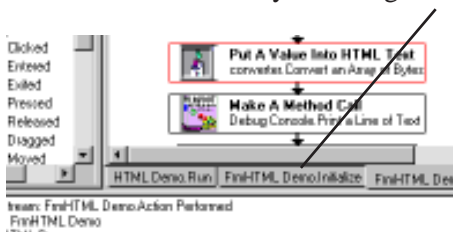
As well as being able to move to a procedure by clicking on the square beside it in the flowchart



or by double-clicking its name in the structure chart



You can also enter it by selecting its tab.



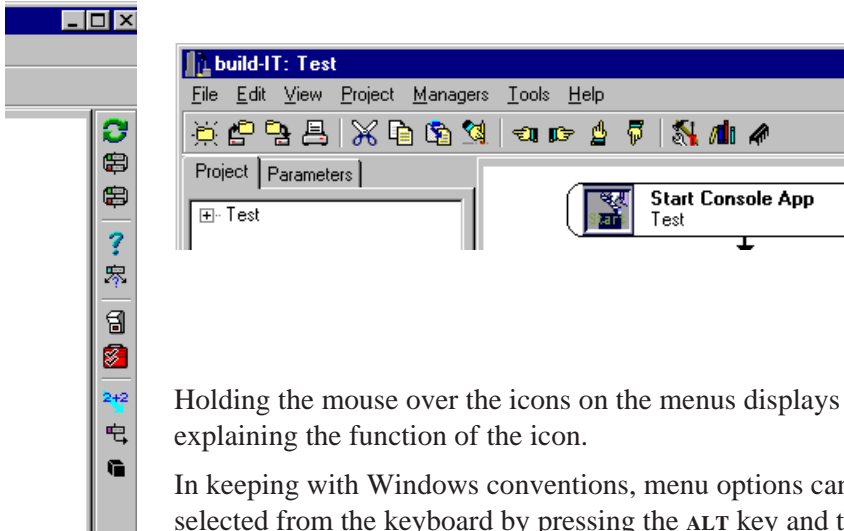
If the tab for the wanted procedure is not visible, slide the horizontal scroll bar or click the horizontal scroll arrow.

The Menus

build-IT features a series of drop-down menus, plus two icon menus—one below the main menu bar and one at the right side of the screen.

The top menu control your overall programming environment.

The side icon menu controls the construction of your flowchart. It contains commonly used tools for inserting elements into your flowchart.



Holding the mouse over the icons on the menus displays help text explaining the function of the icon.

In keeping with Windows conventions, menu options can be selected from the keyboard by pressing the **ALT** key and the underlined letter in the option description together—eg. **ALT-F** to select **F**ile, then **ALT-C** to select **N**ew **C**onsole **A**pplication when the drop-down file menu is displayed.

Other shortcut keys exist for some of the menu options. These are displayed on the right of the option description.

The File Menu:

New Window Application	F2	Starts a new Windows-style application
New Console Application	F3	Starts a new text-based application (similar to the old text-based applications used in the DOS environment). <i>Note: It is possible to create a console application which uses a graphical window, but Java coding</i>
New Java Applet	F4	Starts a new Java Applet to attach to a web page. Java applets enrich the functionality of HTML. They must be run within a web browser. They have limited functionality. The interface is the web page to which the applet is attached.
Load Application		Loads the flowchart for an existing application
Save Application		Saves the flowchart for an application
Close Application		Closes the current application
Load Unit		Loads an existing unit
Save Unit		Saves the currently open unit
Close Unit		Closes the current unit
Exit		Exits build-IT

The Edit Menu

Undo	CTRL-Z	Restores the flowchart to its condition before the last action
Repeat		Re-does the last operation that was undone
Cut	CTRL-X	Cuts the currently highlighted element from the flowchart
Copy	CTRL-C	Copies the currently highlighted flowchart element
Paste	CTRL-V	Pastes the contents of the clipboard into the flowchart
Delete	DEL	Deletes the currently highlighted flowchart element
Find		
Replace		Finds a specified element in the flowchart
Go To		Finds and replaces a specified element in the flowchart Goes to a selected element or procedure

The View Menu:

Note: This menu will differ in the final release

Parameter View

Refresh Screen **F5** Redraws the currently displayed flowchart screen

The Project Menu:

Build

Compiles the current application to a Java class file (the Java equivalent of an EXE)

Run

Runs the current flowchart

Note: Currently builds the application before executing. In the final product the Run option will operate as in previous versions of Layout.

Generate Java and View Source

In the Tech Preview edition and Professional Edition and above only. This option generates editable Java source code and opens the user's default Java editor

The Managers Menu:

- | | |
|-------------------------|--|
| Class Wizard | A tool to define a Java class (ie. a collection of blackboxes.) |
| Blackbox Manager | A tool for adding to, removing from, and rearranging your collection of blackboxes |
| Variable Manager | |

The Tools Menu

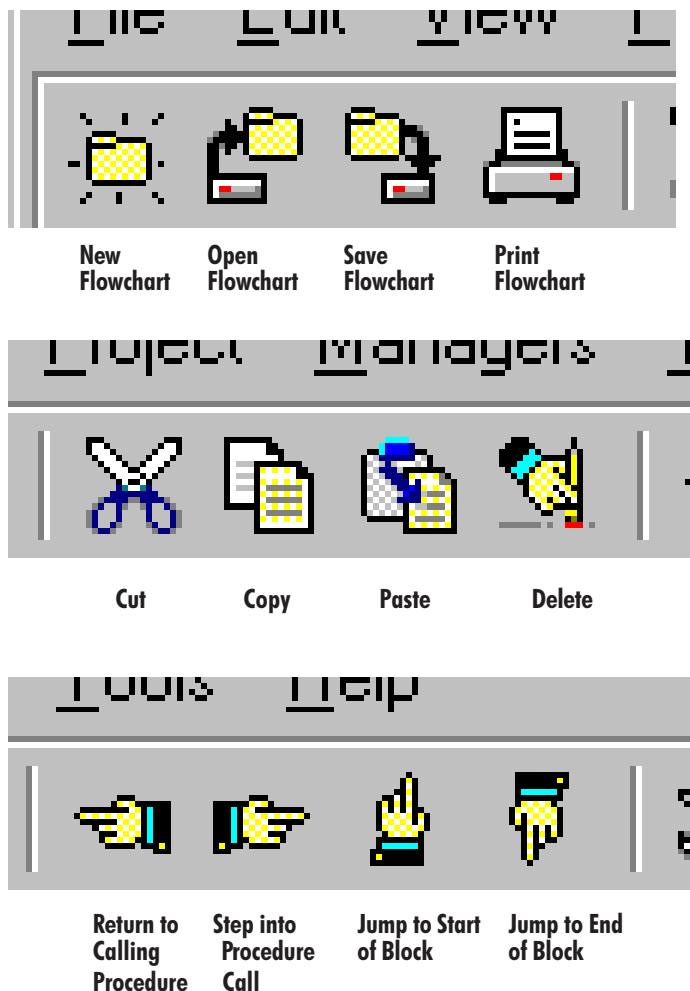
- | | |
|--------------------------------|--|
| Make a New Variable | Opens the Variable Manager to allow creation of a new variable |
| Rename a Variable | Opens the Variable Manager to allow renaming of a variable |
| Blackbox Wizard | Converts a Java class to a build-IT (.JLH) blackbox |
| Advanced Properties | Not yet active |
| Customize Tools | Allows user to customise menus and add his/her own tools |
| Configure Shortcut Keys | Allows user to set up shortcut keys |

Help Menu

- | | |
|------------------------------|---|
| Contents | HELP is not available in the Technology Preview Edition |
| Index | A tool for creating, deleting and renaming variables used in your application |
| Search for Help On... | |
| Commands | |
| Blackboxes | |
| About | |

The Icon Menu

The icon menu is divided into 4 parts, and includes the most often used options of the top drop-down menus:



Return to Calling Procedure works like the red UP arrow on the side of Start and End of Procedure elements in *Layout*. It exits the current procedure and returns to the previous procedure. The advantage of the menu icon is that you can exit up several layers without scrolling.

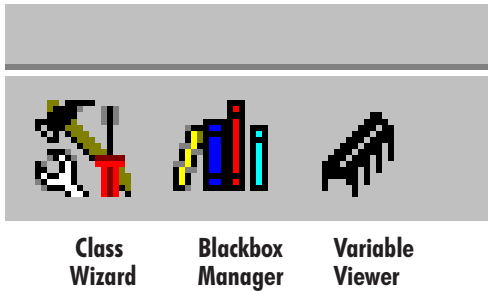
build-IT User's Guide

Step into Procedure works like the green in arrow on the side of Procedure elements in *Layout*. This option is also available in build-IT

Jump to Start of Block takes you to the start of a Decision or Repeat block

Jump to End of Block takes you instantly to the end of the Decision or Repeat

In a *Match Statement*, Jump will move you up or down one element.













The Toolbar

Your flowcharting tools are located on the right of the window.



Holding the mouse over any icon displays a HELP message describing the tool function.

	For Loop	Repeats (Replaces Repeat No. of Times and adds extra functionality)
	Repeat Until	Repeats until a Specified Condition exists
	Repeat While	Repeats whilever a specified condition exists
	Insert If	(Make a Decision) Inserts a decision
	Insert If/Else	(Make a Decision with an Else clause) Inserts a decision with an Else
	Insert Find a Match For	Inserts the start and end of a Match Statement
	Insert Case	Inserts an additional option in a Match Statement
	Put Value into Variable	Sets the value for a selected variable
	Insert Procedure Call	Calls an existing procedure or creates a new procedure
	Make a call to a Blackbox	Allows you to select a "blackbox" or "method" and insert it into your flowchart within a Class

Notes for Layout users:

The *Repeat Number of Times* in Layout has been changed to *Repeat For*, giving increased flexibility to, for example, repeat every third increment of a number. This conforms more closely to programming conventions used in other languages.

The *Make a Decision Tool* has been changed. *build-IT* provides a separate tool for decisions which contain an *ELSE*, rather than a checkbox option within the decision.

In the final release version, a right mouse click on a *Make a Decision* element will allow you to change an *IF* to an *IF/ELSE*.

The *Find a Match Tool* now specifies only the selection to match. The choices which were previously listed inside the *Start of Match* element, and were sometimes difficult to edit and add to, are now inserted individually as needed using the *Insert Case* tool. This provides greater flexibility to position the matches as desired, add to the options, edit, and remove options.

Notes For Layout Users

What has happened to Layout?

Layout was a great product for the DOS and Windows 3.x world. Today, we live in a rapidly changing 32-bit/64-bit world with an increasingly diverse range of computer hardware and operating systems to choose from, new standards in software development, and a strong focus on communications, the internet, and multi-user computing. Platform-independence has become the catch-cry of programmers and users alike!

build-IT was designed to embody the best of *Layout* technology, but expand on it to meet the needs of today's information technology world.

Is *Layout* dead? That decision has yet to be made. Based on response to the *build-IT* preview, inteRAD will decide whether or not it is appropriate to upgrade *Layout for Windows*, or to develop alternate development tools based on the *Layout* technology.

The "Back End"

The original *Matrix Layout* was developed primarily in Assembler and output a compiled EXE program or Basic, C, or Pascal Code. As the computing world moved to Windows, the product evolved to one largely based on C++ and outputting C and C++ code for the Windows environment.

build-IT was written largely in C++, though it will be eventually ported to Java. It outputs Java byte code and distributable Java-based applications as well as "stub" exe programs for the MS Windows environment.

The reasoning behind the move to Java is simple. The computing world needs and wants platform independence. Being able to develop software which can be used on an Apple, Unix, Linux, or Windows machine without having to be extensively re-written for each operating system makes programmers' lives easier, expands their markets, increases their profits, and decreases software maintenance problems. For users, it means a greater availability of a selection of software to suit their needs and freedom of choice in hardware, operating systems and software.

Many computer users question the popularity of Java, assuming that Windows is the "industry standard" and will remain so, and that there is no need to develop for any other environment. The rapid growth in popularity of *Linux* disproves this theory. There are predictions that this growth will continue as a rate so rapid as to cause *Linux* to displace Windows as the most used operating system within a few years. Even if its growth slows, it is an inescapable fact that more and more computer users are migrating to non-Windows environments or expressing a desire to do so.

The internet is a further important reason for the popularity of *Java*. As the use of the internet grows, platform independence and hardware independence increase in importance.

build-IT's designers recognised this trend, and wanted to deliver to users the benefits of platform-independent computing ahead of our competitors.

The Flowchart Structure

Layout users will immediately recognise similarities between *build-IT* and *Layout*—but may be a little overwhelmed by some very significant changes.

The changes were designed to achieve three major goals:

- To increase the power and flexibility of the tool and its suitability for high end professional programmers' needs
- To conform to current industry standards to maintain compatibility with other tools and methods and to ensure that learning *build-IT* was an investment in learning industry-standard methods

- To ensure that the tool's foundation provided a base for updated product releases which could reflect changes in technology trends and could implement new features without requiring a further major rewrite of the product for some time to come. This ensures product upgrades are made available in a timely fashion.

The Tools

Most of the tools with which you are familiar are still available. Minor changes have been made to make tools more usable. New tools have been added to expand your programming capabilities.

The Blackboxes

Although some of the familiar blackboxes are not available in the Technology Preview release, they will all be included in the final product release. A vast range of additional “blackboxes” or “libraries” have been added through the inclusion of the entire *Java* API. Anything you can do in Java, you can now do in *build-IT*—faster and easier.

The Terminologies

In the *Technology Preview Edition*, the names of blackboxes and descriptions of functions are not all the “plain English” *Layout* users are accustomed to. This will change. However, in recognition of the fact that different people think differently and have had different learning experiences, we've designed *build-IT* so that descriptions and terminologies can be changed to suit individual preferences.

What's Missing in the Preview Release?

The *Technology Preview* release of *build-IT* is intended to be a fully usable and capable tool which will deliver sound value for money. However, it is a long way from being the finished product envisioned by *build-IT's* designers.

There are two significant reasons for its release at this time. Firstly, we want to show a committed and loyal user base who have patiently waited for a *Layout* upgrade for a very long time that the product they love does indeed have a future.

Secondly, any good software product is shaped by listening to its users. We want you to help us ensure the final design and features of *build-IT* make it the ultimate tool for your programming needs. By releasing the *Preview Edition*, we allow you to trial the product, evaluate its capabilities, and tell us exactly what you like and dislike and what you would prefer to see added or changed.

This is your tool! Help us shape it to your liking!

The "Card" Tool - The Interface Designer

The *build-IT Technology Preview Edition* lacks an interface design tool.

For those familiar with *Layout*, the *Card Tool* has been discarded.

The new interface design tool will be a much more powerful and full-featured tool with capabilities along the lines of those in *VisualBasic*, *Delphi* and *C-builder's* Form Designer.

The Database

The *build-IT Technology Preview Edition* does not contain a proprietary database.

For those familiar with *Layout's* elegant Cardfile Database, a similar but slightly more capable database will be included in the full release of *build-IT*.

However, in response to popular demand, *build-IT* has been designed to allow the use of any SQL-compliant database and to connect to any ODBC database. This connectivity is transparent. It no longer requires the user to design cards to facilitate a “connection” between the database and the card interface.

Using *build-IT* Tools

The Blackbox Manager

The Blackbox Manager replaces, and is similar to, the *Layout* Blackbox Manager:

- It features folders for grouping blackboxes;
- It allows you to add blackboxes to your system;
- It allows you to re-arrange your blackboxes, change their names, or delete unwanted blackboxes.

Blackbox files in *build-IT* take one of three forms:

.CLS files are blackbox files shipped from interRAD

.JLH files are blackbox files constructed with *build-IT's* Blackbox Wizard which converts Java class files to *build-IT* blackboxes

.BFC files are blackbox files generated using the Class Wizard in *build-IT*

When adding a blackbox to *build-IT*, you must select a folder in which to store the blackbox. You may select an existing folder, or create a new folder.

If you do not select a folder, the system will select a folder for the blackbox based on information contained in the blackbox file.

Adding a Package

The Blackbox Manager also allows you to Add a Package.

A Package is a group of .JLH files contained in a compressed file built with *build-IT's* file compression utility. When adding a package, you do not need to select a folder. The package itself is a folder.

To create your own blackboxes within *build-IT*, select the Blackbox Manager and choose a folder in which to store the new blackbox.

Select Add Blackbox.

Fill in the flowchart for the blackbox.

The Variable Manager

Selecting the Variable Manager opens a floating window in which you can manage all the variables in your current project.

Note: Choosing *Make a Variable* or *Rename a Variable* from the menu invokes the Variable Manager. These options are included on the menu for backward compatibility only.

The floating window may be left open so that variables are always accessible.

build-IT Variable Types

build-IT uses the standard variable types available in the C++ and Java programming language. This may confuse some Layout users at first, but the variable types you are accustomed to are all available—and many more—presented in a slightly different way.

The standard variable types are:

Boolean **True/False** **A checkbox variable which returns T if the statement is true, otherwise F**

Byte **Small Number** **A number between 0 and 255**

Char **A character** **A single text character**

Short **A +/-32767 number, or short integer**

Integer **A +/-2 billion number, or long integer**

Float **A decimal number**

String characters **A line of text or text message containing up to 64,000 characters**

Note that this replaces Layout's text line which could contain only 255 characters and only 79 if the variable was not on a card

Array **A collection or group of variables**
An array of 20 bytes, for example, is a set of 20 variables containing numbers between 0 and 255.

You don't need to learn the names above. You can change the variable names displayed to any name that is easy for you to understand and remember.

Understanding Classes

C++ and Java programmers use classes. Layout programmers might want to think of classes as being like cards. Classes can contain collections of variables (like objects on cards) and methods (blackboxes attached to cards). Classes, however, have substantially greater capabilities than Layout's cards.

First, they feature a quality called inheritance. To understand inheritance, think about a situation where you need a card which is identical to a previously created card but adds additional objects. In Layout, you might simply copy the card to one with a new name and add the extra objects. That uses substantial resources. With classes, you can define a class which is a sub-class of another class—that is, a class which contains everything the original class contains but extends on it—without using all the additional resource needed to actually copy the original class.

To understand inheritance, think of a Window as being a class. The window has a standard border and title bar. Now think of a decorative window with a three-dimensional border. This new window can be created by calling the Window class and creating a sub-class which defines the changes in the appearance of the window.

Second, classes can have private members. To understand private members, think of a card. Any procedure in your flowchart can change the content of variables on the card. Now imagine the card is contained in a blackbox. Only actions that occur in the blackbox can change the content of the variable unless the card is accessible from the main program. When using a class, you can define the class as public (like a card in your flowchart—accessible from anywhere in the program) or private—meaning that only actions which take place within the class can alter the content of variables in the class.

Finally, classes allow overloading. Overloading refers to the use of multiple classes of the same name to perform different functions, by passing a different parameter. This is not an advantage to end program users. It simply increases the reusability of code for other programmers. It means your blackboxes can perform a wider range of functions from a single blackbox.

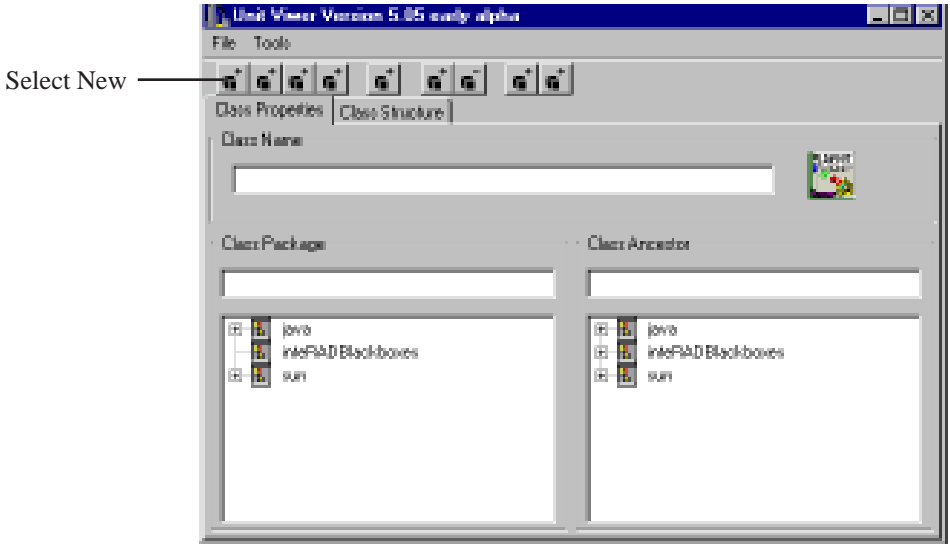
Don't be concerned if you don't fully understand the concept of classes. You really don't have to understand to use *build-IT* just as you used *Layout*. In any case, as with *Layout*, using the product will help you increase your understanding of it over time. The important thing to understand about classes is that classes are variables. So, when you think of the variables in *Layout* that don't appear in the above variables list—cards, cardfiles, groups of checkboxes, pictures, etc.—these variables are available to you by using classes.

In addition, you can create your own customised variables—like cards, but with more functionality and flexibility—by creating classes.

Naturally, C++ and *Java* programmers will welcome the changes and instantly understand them. *Layout* programmers may be a little overwhelmed by them at first, but *build-IT* doesn't really require any additional skill or knowledge to use. You can use it just like you used *Layout*. Over time, you will find yourself experimenting with the extra features and as your understanding develops from experimentation, you will welcome the additional functionality they provide.

Creating a Blackbox - Step by Step

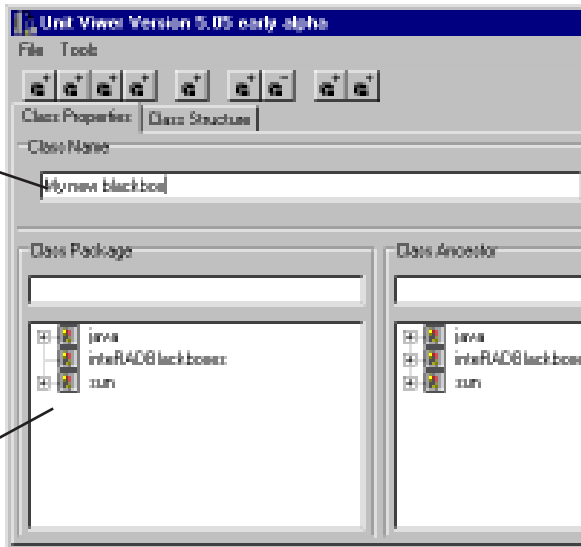
Select the *Class Wizard* from the drop-down **Managers** menu. The Class Wizard screen appears.



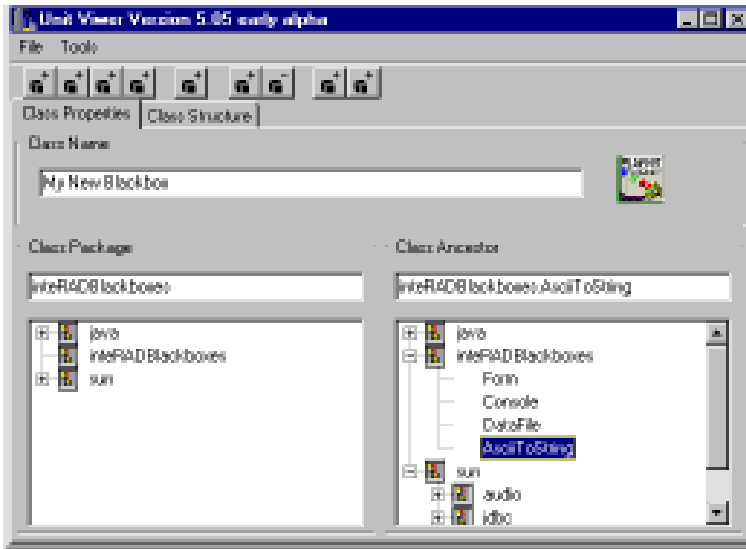
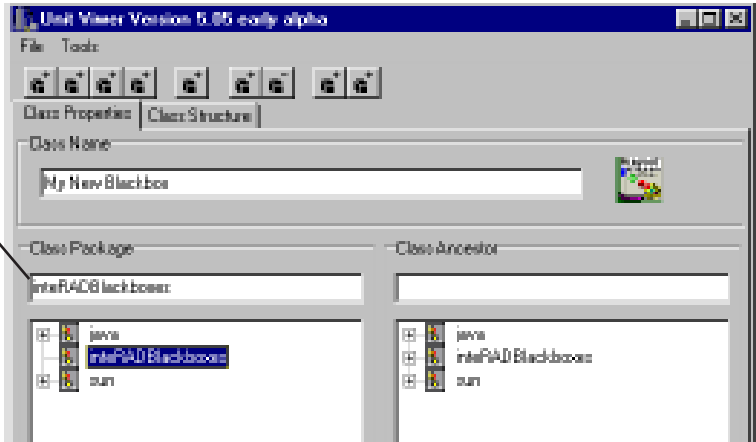
New Class appears in the Class Name dialog box.

Double click and change this to the name you want to use for your new blackbox.

Click on the folder you want to use to store the blackbox.



The folder name appears in the Class Package dialog box.



If you want this class to be a sub-class of an existing class (i.e. to inherit the characteristics of a class), select the parent class in the list on the right side of the window.

The parent class name appears in the Class Ancestor edit box.

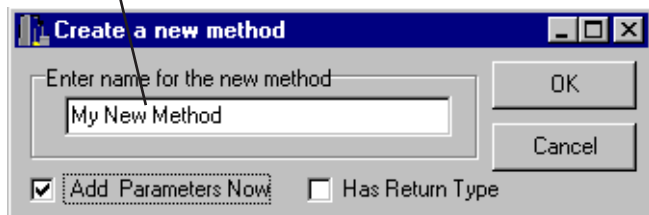
If this is not a sub-class of an existing class, leave the Class Ancestor edit box blank.

Click the *Class Structure* Tab. The display changes to show the Class Structure.

Select the *Add new Method* icon



Enter a name for the method



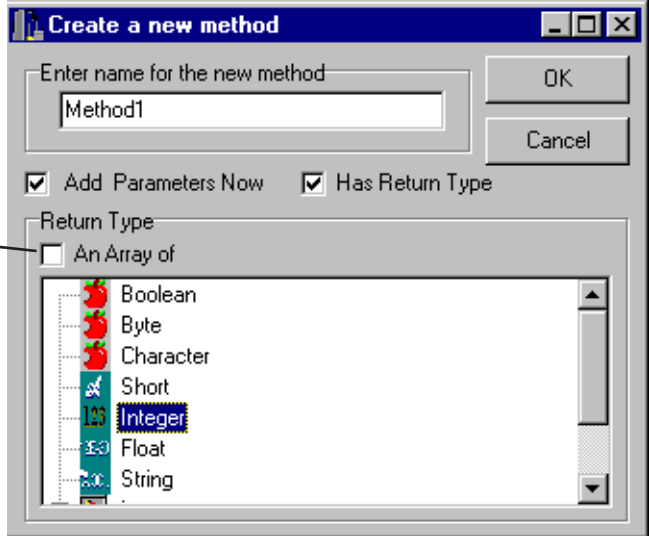
If the blackbox has inputs or outputs, check the *Add Parameters Now* checkbox.

If the blackbox has a return value, check the *Has Return Type* checkbox.

If *Has Return Type* is selected, the following screen displays:

Select the *Return Type* from the displayed list.

If the *Return Type* is an *Array*, check the *Array* box.



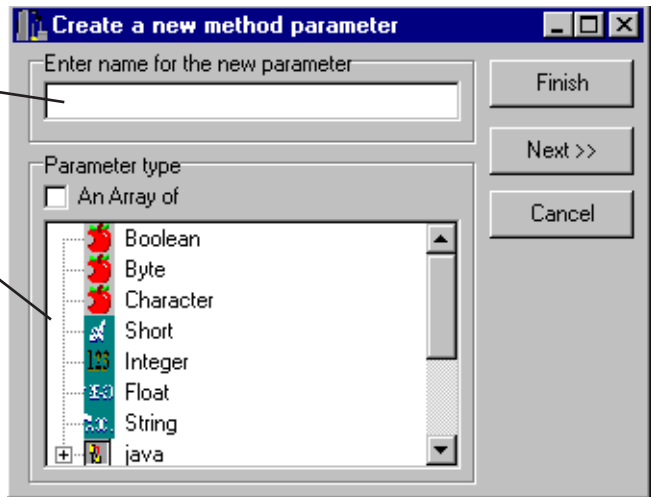
Click the **OK** button.

The *New Method Parameter* window opens:

Enter the name you wish to use for an input variable.

Select the type for the variable by clicking on the correct type in the list.

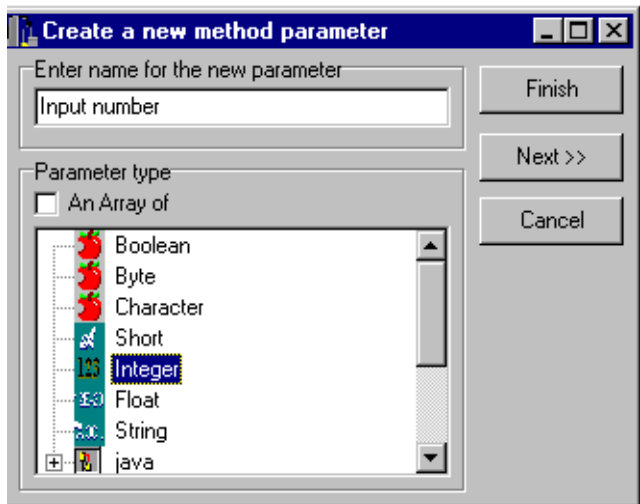
Note that if the type is an array, you should check the *An Array of* checkbox as well as selecting the type.



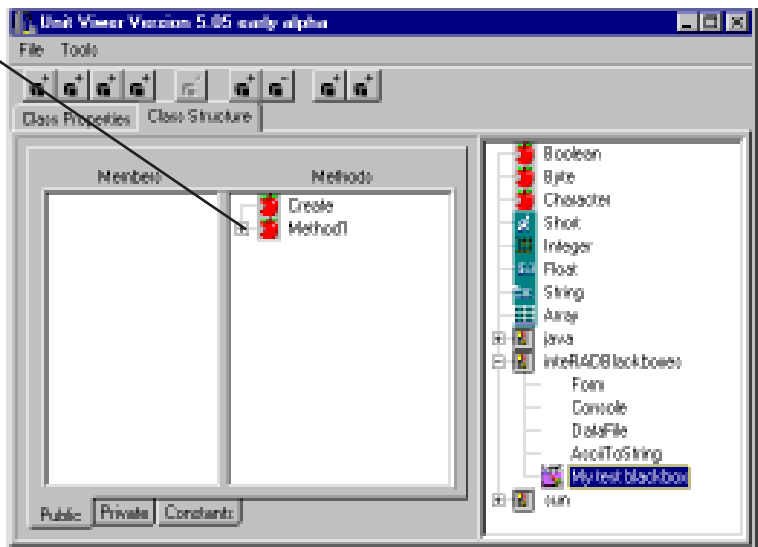
Click the **NEXT** button to add another input variable.

Click the **FINISH** button when you have added all input variables.

The Class Structure window is again displayed.

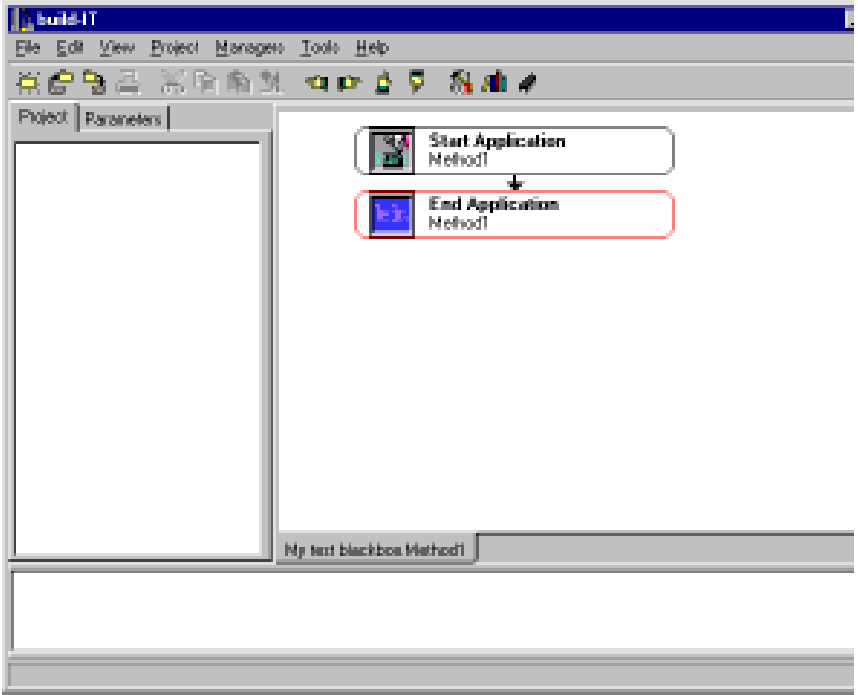


Double click on the blackbox name in the methods list.



build-IT User's Guide

The flowchart screen appears to allow you to create the flowchart for the blackbox.



Once the flowchart is complete, switch to the Class Wizard Window which is still open in the background and select the *Register Icon* to add the blackbox to your blackbox library so that it can be reused in other programs.

