The Blue Programming Environment

Reference Manual

Version 1.0

Michael Kölling
Monash University

2 December, 1998

How Do I ... ? – Introduction

About This Document

This is the "How Do I..." reference manual for the Blue Application Development Environment. This manual tries to answer questions about the Blue environment. These questions are assumed to be of the form "How do I do this-or-that?" You can replace the *do this-or-that* part with any of the section headings, and hopefully construct a question that is both close to what you wanted to ask and answered in this manual. Of course, you can also just browse through this manual, maybe after you have used the environment for a while, and possibly find out one thing or another that you did not know about the Blue environment. There are lots of tips and tricks that are not essential (you get along without them) but which are handy, neat and can speed up your work with Blue. Many of these are described in this document.

The sections in this manual contain many cross references. Cross references are shown in a distinct type style, as in the following example:

> To edit the source of a class, you must first **OPEN A CLASS (2.6)**. Depending on whether the class was compiled or not, you will see...

Here an action is mentioned ("open a class") that is itself explained in another section (2.6). The type style indicates the cross reference, which can be looked up in turn, should you be uncertain about how to perform this task..

Related Documents

This document describes only the Blue Programming Environment. The Blue Language is described in "Blue – Language Specification":

> M. Kölling and J. Rosenberg, *Blue - Language Specification, Version 1.0,* School of Computer Science and Software Engineering, Monash University, Technical Report TR97-13, November 1997.

Design decisions and related issues are discussed in various papers:

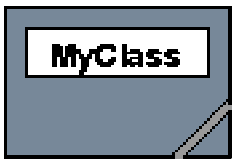
> M. Kölling, B. Koch and J. Rosenberg, *Requirements for a First Year Object-Oriented Teaching Language*, in ACM SIGCSE Bulletin, ACM, Nashville, 173-177, March 1995.
>
> M. Kölling and J. Rosenberg, *Blue - A Language for Teaching Object-Oriented Programming*, in Proceedings of 27th SIGCSE Technical Symposium on Computer Science Education, ACM, Philadelphia, Pennsylvania, 190-194, March 1996.
>
> M. Kölling and J. Rosenberg, *An Object-Oriented Program Development Environment for the First Programming Course*, in Proceedings of 27th SIGCSE Technical Symposium on Computer Science Education, ACM, Philadelphia, Pennsylvania, 83-87, March 1996.

J. Rosenberg and M. Kölling, *Testing Object-Oriented Programs: Making it Simple*, in Proceedings of 28th SIGCSE Technical Symposium on Computer Science Education, ACM, San Jose, Calif., 77-81, February 1997.

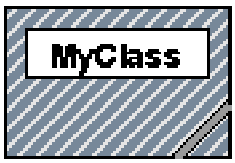
Basics

The Class Icons

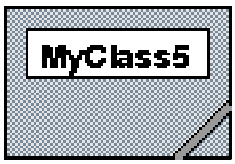
Classes are displayed in the main window in the form of icons. These icons appear in a few variations. Here is a list of possible appearance of those icons.



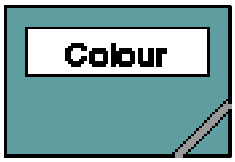
Blue. A normal class, compiled, ready for object creation.



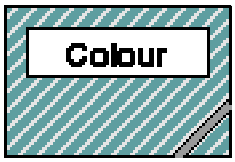
Striped. A striped icon indicates that the class is uncompiled.



Light Blue. The light colour marks abstract classes. This class is compiled, but no objects can be created, because the class is abstract.

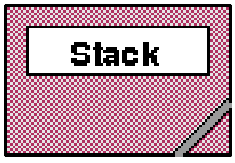


Green. This colour indicates enumeration classes.

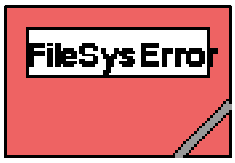


Striped green. An uncompiled enumeration class.

Red. This colour marks library classes. (This class was imported from a class library.)



Light red. An abstract library class.



Orange. A library enumeration class.

In addition to this, classes can appear greyed and have an additional lock mark if the project is a group project:

Grey. The class is not locked or taken by the current user. It cannot be edited or executed.

Blue with edit mark. This class has been taken by the current user. It may be edited and executed.

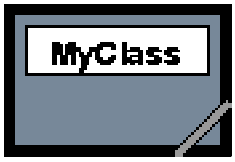
Blue with lock mark. This class has been locked by the current user. It may be executed, but not edited. It may be locked by other users as well.

Grey with edit mark. This class has been taken by another user. It cannot be taken or locked by the current user.

Grey with lock mark. This class has been locked by another user. It may be locked by the current user as well, and then used for execution.

Selecting A Class

To select a class, click on it. (If your mouse has more than one button use the left button to select a class.) A selected class is shown with a black border:



Selected class.

1. Projects

1.1 Create A New Project

menu: Project – New... *shortcut:* —

To create a new project, you first have to start Blue. If Blue is started already, and you already have a project open, you must first close that project. Blue can only have one project open at any one time.

Select New... from the menu. A file selection dialog will open. **USE THE FILE SELECTION DIALOG (8.6)** to enter a name and location for the project.

A project name always ends in *.bp*. If you type a name that does not end with this suffix, Blue will automatically append it to the name you type.

Click Ok, and the new project is created and opened.

1.2 Open A Project

menu: Project – Open... *shortcut:* Alt-o

There are two ways to open a project: from the command line of a shell (before Blue is started) type

```
blue <project-name>
```

to start Blue and automatically open the project <project-name>.

If Blue is already running, use the Open... command from the menu. If another project is open already, you need to close that project first. A file selection dialog will open. **USE THE FILE SELECTION DIALOG (8.6)** to select a project to open. Click Ok, and the project is opened.

1.3 Find Out What A Project Does

menu: Tools – Show *shortcut:* Alt-e *toolbar button:* Show

A project is described in the project description. The project description is shown as a note icon close to the top left corner of the main screen:

To read the project description, select the note by clicking on it, and select the Show command from the menu or the toolbar. A double click on the note icon is a shortcut to the same function.

1.4 Copy A Project

To copy a project, you first **OPEN A PROJECT (1.2)**. This is the original you want to copy. Then you select Save As... from the project menu to save the project under a different name. The Save As... will open a file selection dialog. **USE THE FILE SELECTION DIALOG (8.6)** to enter a new name and location for the project.

If the original project is read-only for you, all classes will appear greyed out. This means that you cannot modify or execute the original project. This is not a problem – as soon as you save the project under a new name, the classes will appear normal and you will have modify and execute rights.

1.5 Delete A Project

menu: Project – Remove A Project On Disk...

To remove a project on disk, choose the Remove A Project On Disk... command from the project menu. **USE THE FILE SELECTION DIALOG (8.6)** to select a project and click Ok. The project will be deleted.

Warning: Once a project is removed from disk, it cannot be recovered!

1.6 Compile A Project

menu: Tools – Compile *shortcut:* Alt-c *toolbar button:* Compile

To compile a project, select Compile from the menu or toolbar. This function will make an analysis of the current project, check which classes need recompilation, check dependencies, and then compile all classes that need compilation in the appropriate order.

This function will always try to compile all uncompiled classes. It aborts as soon as in error is detected in one of them. To compile one or more specific classes without compiling them all, see **COMPILE A CLASS (2.4)**.

1.7 Document A Project

To document a project properly, you should at least do the following: **WRITE A PROJECT DESCRIPTION (3.10)**, comment all classes and comment all routines.

Class comments and routine comments are both entered in the class's source code. **OPEN A CLASS (2.6)** to enter these comments. See [1] for a more detailed description of class comments.

In addition to this, you can also **WRITE A PROJECT COMMENT (3.11)**.

1.8 Execute A Project

To execute a project, you first have to find out which routine you want to execute. Some projects have a hierarchical structure with one top-level class. The top-level class can have one or more start routines. Some projects have a more open structure in which you call various functions from different classes to perform the various tasks the project offers. If

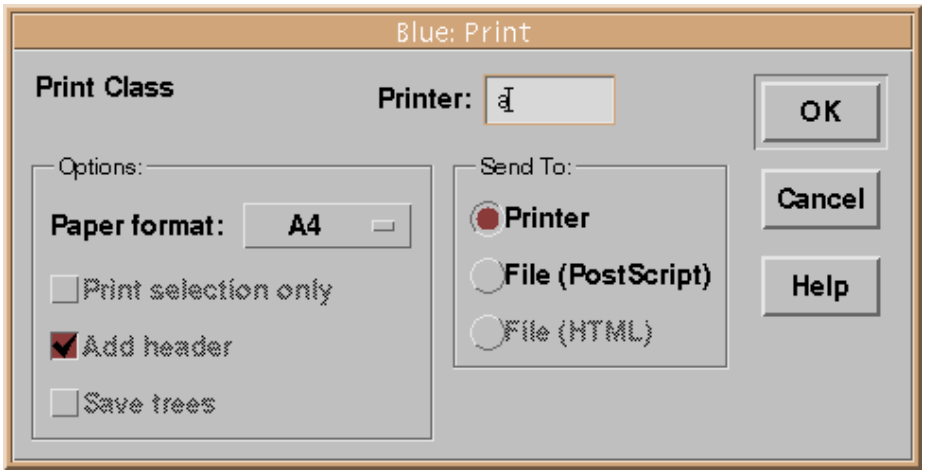
you do not know which class or which routine you should use have a look at the project description note. It should be described there. The project description note is further explained in **FIND OUT WHAT A PROJECT DOES (1.3)**. As a rule-of-thumb, top-level classes are usually placed in the project window near the top-left corner.

Once you know what you want to execute, you have to **CREATE AN OBJECT (5.1)** first. You cannot execute code without creating an object. Once you have created that object, you can **CALL A ROUTINE (5.2)**.

1.9 Print The Project Window

menu: Project – Print... *shortcut:* Alt-p

To print a project, select Print... from the project menu. A dialog appears that looks like this:



Insert the name of the printer in the text field. You can print directly to the printer, create a PostScript file for later printing or viewing or create an HTML page with the project overview on it.

The “Print selection only” option is always disabled for printing the project window (it applies only to the printing of text). “Add headers” adds a header to each page, “Save trees” prints in reduced format to save paper.

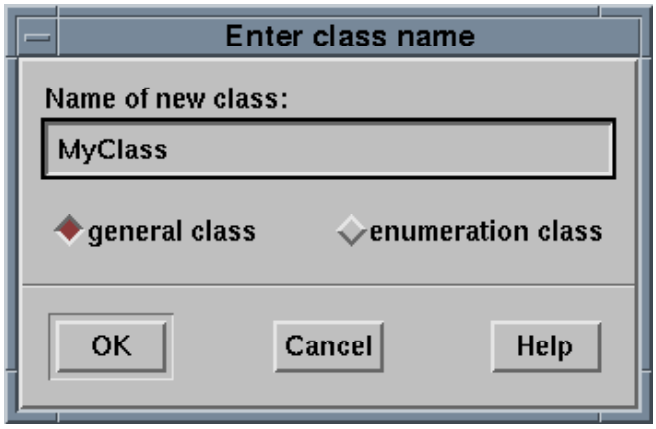
2. Classes

2.1 Create A New Class

menu: Edit – New Class... *shortcut:* Alt-n *toolbar button:* New Class...

To create a new class in a project, select the New Class... menu item or toolbar button. This is only possible while you have a project open.

A dialog will appear to let you enter a name for the new class:



You have to enter a name, and choose one of the options "enumeration class" or "general class".

The name must be a valid Blue identifier. In short, it must be a word that consists only of letters, digits and the underscore (_). No spaces or other characters are allowed. The first character may not be a digit.

The "enumeration class" or "general class" options only determine what kind of skeleton is used for the initial source of the class. It is not a final choice: you could change your mind later, and replace the complete source code of, say, a general class with code for an enumeration class. The environment would recognise that the class is now an enumeration class and treat it accordingly. Of course, this works the other way around as well.

2.2 Remove A Class

menu: Edit – Remove Class *shortcut:* —

To remove a class, select it and then choose Remove Class from the Edit menu. You will be asked to confirm, since removing a class is not reversible. Once a class has been removed, it is deleted in the file system and cannot be restored (unless it is a library class).

2.3 Rename A Class

To rename a class, first **OPEN A CLASS (2.6)**, then replace the name of the class in the class header (the first line of the source) with the new name you want it to have. As soon as you save or close the class, the new name will also be shown in the class icon in the main window.

2.4 Compile A Class

main window:
menu: Tools – Compile Selected *shortcut:* Shift-Alt-c

editor:
menu: Tools – Compile *shortcut:* Alt-c *toolbar button:* Compile

There are two ways to compile a particular class: from the main window or from the editor.

In the main window, select the class and then select Compile Selected from the menu.

If the class is open, select Compile from the Tools menu or the toolbar in the editor.

Both of these functions do the same thing: they will first analyse the dependencies of the selected class and check whether a class that this one depends on is uncompiled. Classes that this one depends on are classes used and the parent class (if any). Those classes are compiled first, if necessary, and then the selected class is compiled. Thus, it is not necessary to explicitly compile used classes first. If an error is found it will be highlighted and an error message is displayed (the class will be opened if it is not open already). If no error is found, the class will be marked as compiled in both the editor and the main window.

It is not necessary to save classes before compilation. They will be implicitly saved before compilation takes place if they were changed since the last save.

2.5 Edit A Class

To edit the source of a class, you first have to **OPEN A CLASS (2.6)**. If the class is in *interface view*, you have to switch to the *implementation view* (see **VIEW THE IMPLEMENTATION OF A CLASS (2.8)**).

Then the class is ready to be edited. If it still cannot be edited (the editor shows a "read-only" mark), then you do not have enough access rights to edit the class. It might be a library class, or the project might belong to someone else who has not given you permission to change it.

2.6 Open A Class

menu: Tools – Show *shortcut:* Alt-e *toolbar button:* Show

"Open a class" is the expression we commonly use to shorten the more precise "Open an editor to show the source of a class". You open a class if you

- want to see the interface of the class
- want to see the implementation
- want to edit the class (change the interface or implementation)

You can open a class by selecting it and then selecting Show Class from the Edit menu or toolbar. A shortcut for opening a class is a double click on the class icon.

Depending on which view you used when you last looked at the class, you might see it in *implementation view* or *interface view*. If you have not opened the class in this session, it will initially appear in interface view if it is compiled, and in implementation view if it is uncompiled.

The leftmost button in the editor toolbar indicates which view is currently shown:

Interface is pushed in: The interface view is shown.

Interface is out: The implementation view is shown.

2.7 View The Interface Of A Class

editor:

menu: Tools – Show Interface *shortcut:* Alt-i *toolbar button:* Interface

To see the interface of a class, you have to **OPEN A CLASS (2.6)**. If the class display is in implementation mode, select the Interface toggle from the Tools menu or the toolbar.

Showing the interface is only possible if the class has been compiled.

2.8 View The Implementation Of A Class

editor:

menu: Tools – Show Interface *shortcut:* Alt-i *toolbar button:* Interface

To see the implementation of a class, you have to **OPEN A CLASS (2.6)**. If the class display is in interface mode, select the Interface toggle from the Tools menu or the toolbar.

3. Project Editing

3.1 Create and Remove Classes

To find out how to create or remove a class from the project, see **CREATE A NEW CLASS (2.1)** and **REMOVE A CLASS (2.2)** in the "Classes" section of this manual.

3.2 Layout A Project

You can change the layout of the project (the icons in the main window) without changing its functionality. You should attempt to create a layout that reflects as closely as possible the logical application structure.

To change the layout, you can **MOVE A CLASS ICON (3.3)** on the screen. Move all the icons until the layout appears the way you want it. You cannot move arrows yourself. All arrows are computed and drawn automatically.

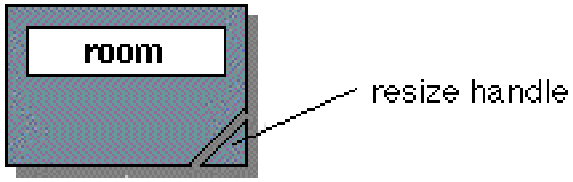
You can also **RESIZE A CLASS ICON (3.4)**.

3.3 Move A Class Icon

To move an icon on the screen, you drag it with the (left) mouse button. Dragging means: you click in the icon and keep holding the mouse button down while you move the mouse around. Release the mouse button to drop the icon.

3.4 Resize A Class Icon

To resize a class icon, you click in the lower right corner of the icon and drag the corner. This corner is separated from the rest of the icon to mark the area in the icon used for resizing.



3.5 Move An Arrow

You cannot move an arrow explicitly. If you do not like the layout of your arrows, all you can do is to move your classes around. All arrows will be redrawn automatically.

3.6 Add A "Uses" Arrow

menu: Edit – Add Uses Arrow *shortcut:* Alt-u *toolbar button:* 

To add a "uses" arrow, select the Add Uses Arrow item from the menu or toolbar and then drag the arrow from the client class to the server class. (That is: click and hold the mouse button in the client class, move the mouse pointer to the server class while still holding the button down, and release the button in the server class.)

Alternatively, you can add the used class to the uses clause in the source of the client class. The project overview will be updated as soon as the class is saved or closed.

3.7 Remove A "Uses" Arrow

menu: Edit – Remove Uses Arrow *shortcut:* Ctrl-Alt-u

To remove a "uses" arrow, select Remove Uses Arrow from the menu and then drag the arrow from the client class to the server class. This will remove the existing arrow.

Alternatively, you can remove the used class from the uses clause in the source of the client class. The project overview will be updated as soon as the class is saved or closed.

3.8 Add An "Inherits" Arrow

menu: Edit – Add Inherit Arrow *shortcut:* Alt-i *toolbar button:* 

To add an "inherits" arrow, select the Add Inherit Arrow item from the menu or toolbar and then drag the arrow from the subclass to the superclass. (That is: click and hold the mouse button in the subclass, move the mouse pointer to the superclass while still holding the button down, and release the button in the superclass.)

Alternatively, you can insert the superclass to the class header in the source of the subclass. The project overview will be updated as soon as the class is saved or closed.

3.9 Remove An "Inherits" Arrow

menu: Edit – Remove Inherit Arrow *shortcut:* Ctrl-Alt-i

To remove an "inherits" arrow, select Remove Inherit Arrow from the menu and then click into the subclass where the arrow originates. This removes the arrow.

Alternatively, you can remove the superclass from the class header in the source of the subclass. The project overview will be updated as soon as the class is saved or closed.

3.10 Write A Project Description

The project description is entered in the *project description note*. The project description note is shown on the main screen as an icon:

You can double-click the note to open it, and then write the description of the project into it. It should at least contain a statement describing what the project does, who wrote it, what it is intended for, and how a user starts it (what classes/routines to use to invoke certain functionality). The project description note provides, by default, a structure to enter some of these details. It is a good idea to use this structure, since it makes it easier to find information.

3.11 Write A Project Comment

A project comment is a short piece of text that appears directly in the main window, together with the project overview (the class icons and arrows). Project comments are usually used to add brief notes to a class or group of classes to make the application structure more explicit.

To enter a project comment, select the toolbar button labelled ***Abc*** or the menu item Add Comment... from the Edit menu.

comment button

NOTE: Not Yet Implemented

4. Editing Source Code

4.1 View The Source Code Of A Class

How to view the source of a class is described in **OPEN A CLASS (2.6)** and **VIEW THE IMPLEMENTATION OF A CLASS (2.8)** in the section about classes.

4.2 Enter Text

Once you see the source of a class on the screen, you can just type to enter text. If this does not work, have a look at the bottom right corner of your editor window. If the class is marked as "read-only" then you cannot enter any text or change any of the existing text.

The reason a class is read-only could be:

- you are looking at the interface (instead of the implementation)
- the class is a library class
- you do not have enough access rights in this project to change classes
- this class is locked (by you or another user)
- the class is in a group project and you have not taken the class for editing

After entering text, you can compile or close the class. A Save function exists, but rarely needs to be called explicitly. Compiling or closing a class automatically saves the class.

4.3 Enter A New Routine

To enter a new routine, type the routine name in the editor at the location where the routine should appear, and then type *Control-R*. *Control-R* calls the Red function *blue-new-routine*. This function will insert the rest of a routine skeleton and place flags into the skeleton at places where text has to be inserted. The function *next-flag* (Ctrl-Tab) can be used to move the cursor to the next flag to fill in the gaps.

4.4 Find Errors In The Source

To check the source of a class for error, you just **COMPILE A CLASS (2.4)**. Compilation will highlight the first error in the source and display the error message in the information area in the editor window.

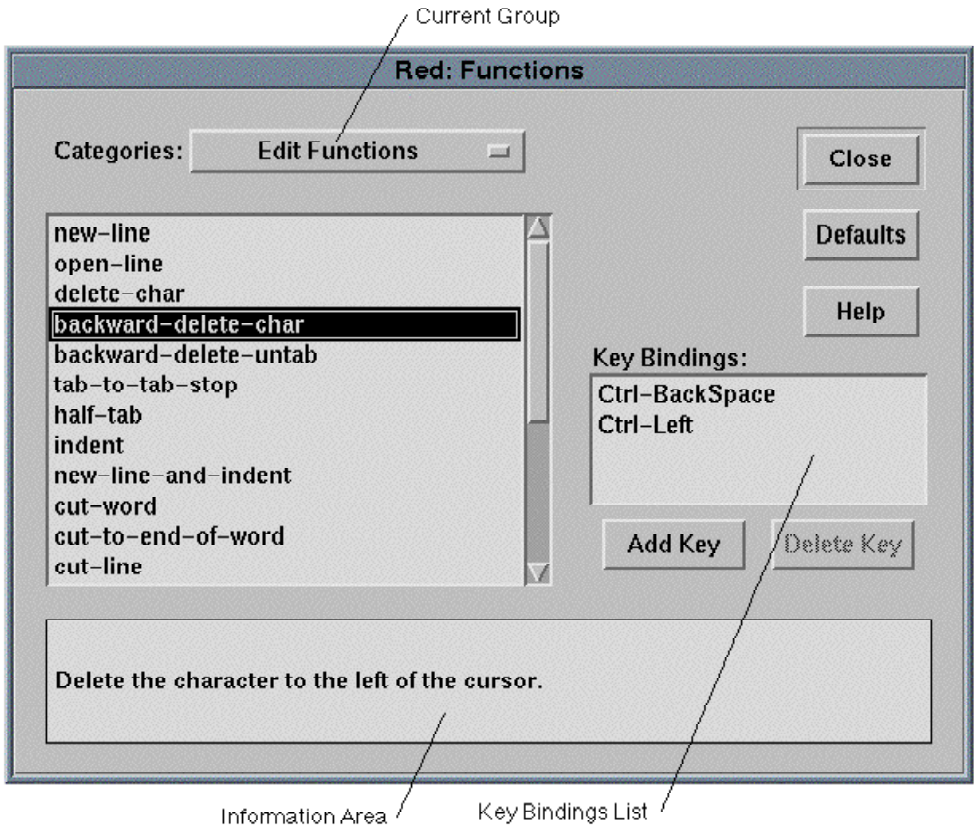
Compilation only shows one error at a time. To see other errors, fix the first one and compile again.

4.5 Find Out What The Editor Can Do

editor:
menu: Preferences – Key Bindings
menu: Help – Manual

Getting to know your editor is a very valuable thing that can greatly increase your work efficiency (and it's a lot more fun too!). Red (the editor in the Blue system) has two mechanisms to find out what functions are supported and what each function does. The key bindings dialog (select Key Bindings from the Preferences menu) offers a full list of editor functions. It also shows a brief explanation and the key combinations that call the function.

The following is an example of a key bindings dialog:



The functions are organised in groups. Choose a function group first to see a list of functions in that group. The information area then shows a description of the function and the key bindings list shows what key combinations call the function.

The second source of information is the Red Online Manual. To open it, select Manual from the Help menu. Doing so will start a web browser displaying an HTML manual that includes a tutorial and a reference section.

4.6 Find Out What A Function Key Does

If you quickly want to check what function any key combination invokes in Red, type Alt-D and the key in question. If, for instance, you wonder what the function key F5 might do, type [Alt-D] [F5]. Alt-D calls the function describe-key. This function reads the next keypress and displays the name of the function that is called by this key in the information area.

4.7 Change Key Bindings

editor:

menu: Options – Key Bindings *shortcut:* Alt-k

To change key bindings in Red, select Key Bindings from the Preferences menu. This will open the key bindings dialog, that is shown in the **FIND OUT WHAT THE EDITOR CAN DO (4.5)** section. Here, you can add, remove or change the keys and key combinations that call a function.

4.8 Find Out More

editor:

menu: Help – Manual

To find out more about Red, have a look at the Red User Manual. It is available online at

http://www.csse.monash.edu.au/~mik/red-manual

It can also be opened from within Red by selecting Manual from the Help menu.

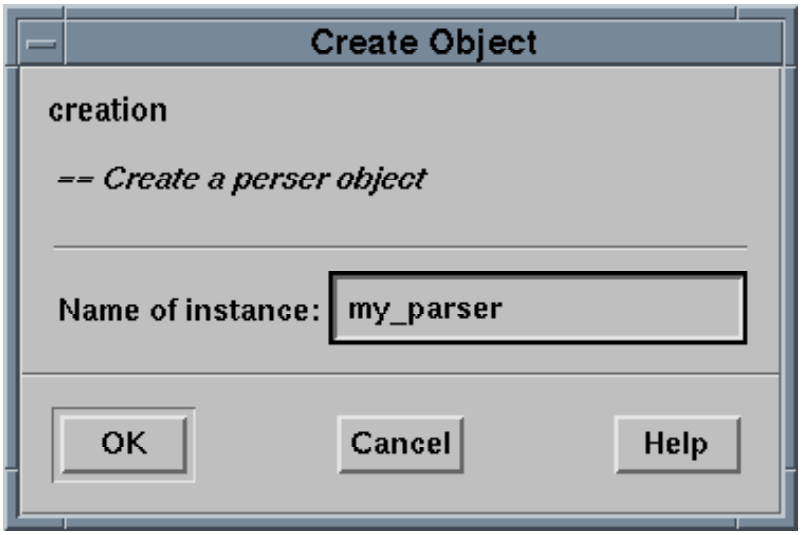
5. Execution

5.1 Create An Object

menu: Tools – Create Object... *shortcut:* Alt-k *toolbar button:* Create...

To create an object of a class, select the class and then select Create from the Tools menu or the toolbar. If your mouse has more than one button, then clicking a class with the right mouse button is a shortcut for Create.

An object creation dialog will appear. This dialog may look like this:

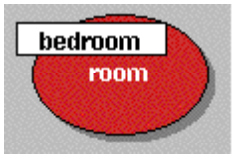


In the name field, you have to enter a name for the object to be created. The name has to be a valid Blue identifier.

If the creation routine has parameters, the dialog will also include fields to enter the parameters. These fields are needed for the call of the creation routine and are identical to the fields in a normal routine call dialog. See **CALL A ROUTINE (5.2)** for details.

Objects can only be created from compiled general classes.

Once the object is created, it will appear on the object bench (the area close to the bottom of the main window) as a red ellipse. The object icon shows its name and class. For example:



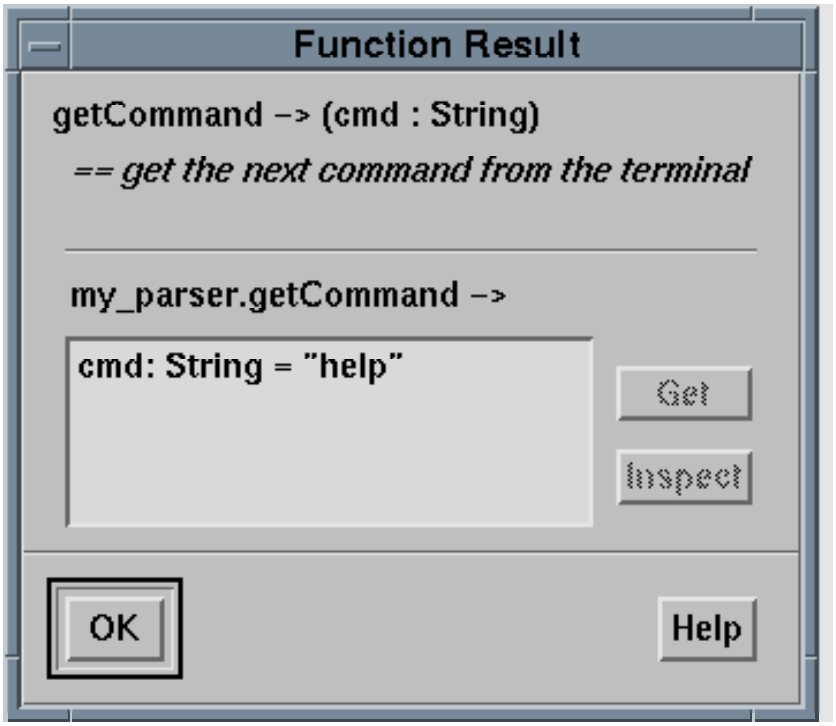
5.2 Call A Routine

To call any routine, you must first **CREATE AN OBJECT (5.1)**. Once you have the object on the object bench, clicking it with the right mouse button will show a pop-up menu which lists all interface routines of the object:

Select the routine you wish to call.

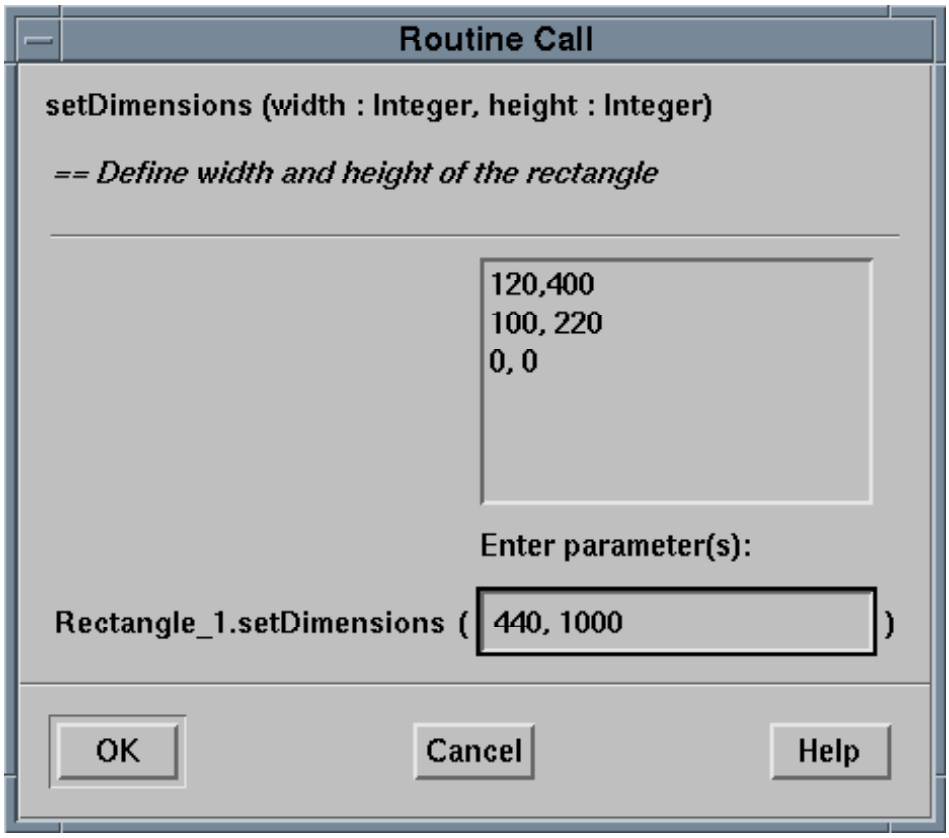
If the routine has parameters, you will have to **ENTER PARAMETERS (5.3)**.

After clicking OK, the routine will execute, and function results (if any) will be displayed in a separate function result dialog:



5.3 Enter Parameters

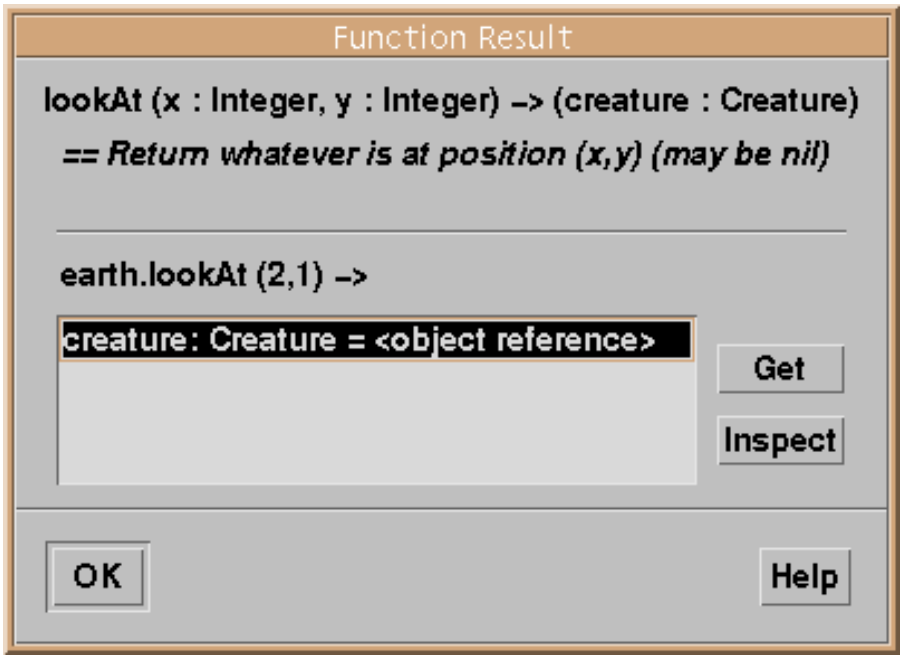
If you **CALL A ROUTINE (5.2)** (including a creation routine which is implicitly called when you **CREATE AN OBJECT (5.1)**) and that routine has parameters, then a parameter dialog will appear to let you enter parameters.



Enter the parameters in the field between the parenthesis or click into the history list to re-use a previously typed parameter list. All parameters are written in exactly the same way they would be written as part of a routine call statement in the source code of a class. Strings, for example, have to be written with quotes.

5.4 Execute An Object That Was Returned By A Routine

If a routine call returns an object that you want to examine or use further, you can select that object in the function result dialog and then select the Get button.



Selecting Get will place that object onto the object bench where you can examine or call it like any of your existing objects.

5.5 Use An Object From The Object Bench As A Parameter

You can use an object that lies on the object bench as a parameter to a routine call. To do this is simple: While you **ENTER PARAMETERS (5.3)** for the routine, simply click on the object you want to pass in. This will insert the object's name into the parameter list. (You could also manually type the object's name.)

5.6 Stop The Execution Of A Blue Program

To interrupt a running Blue program, you can simply type *escape* (a key often labelled ESC) in the main window.

Alternatively, you can **OPEN THE MACHINE CONTROLS WINDOW (6.7)** and click on the stop button.

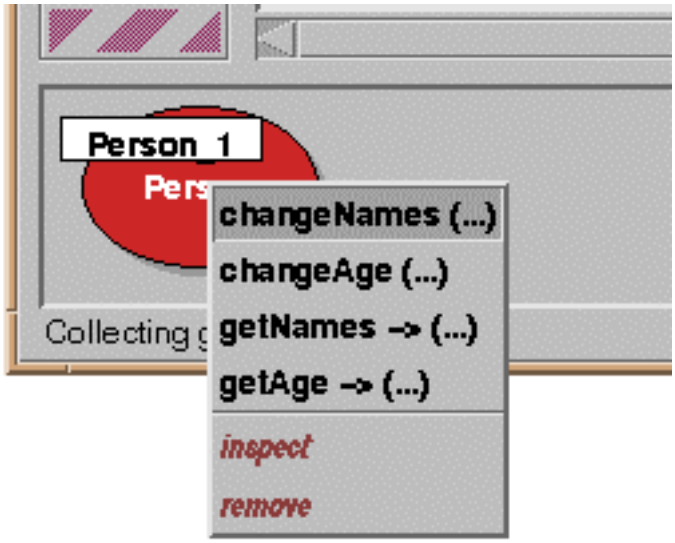
6. Debugging

6.1 Inspect An Object

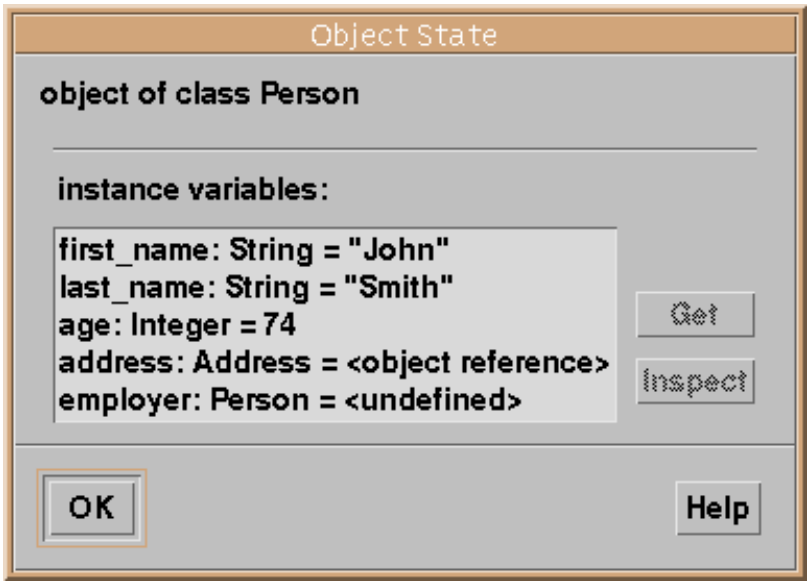
The most essential part of debugging is object inspection. Object inspection allows you to look inside an object and see the current values of its instance variables.

You can inspect an object by selecting the Inspect item from the pop-up object menu or by double clicking the object.

To see the pop-up menu, click the object icon on the object bench with the right mouse button. A menu will pop up that looks similar to this:



Under the routine calls there are two special items. The second from the bottom is Inspect. Select this to open the object. A dialog will be displayed that shows all instance variables, their types and values.



If the value of a variable is itself a complex object, it is only shown as <object reference>. If you are interested in the details of such an object, you can inspect that object in turn by selecting it and then clicking the inspect button.

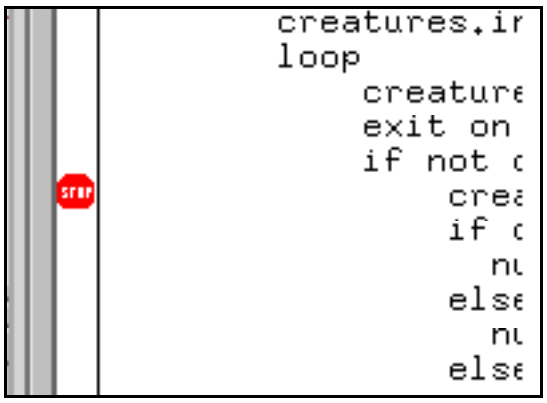
In this way a whole data structure may be examined. If your structure is, for instance, a linked list, you can inspect the objects referred to by the *next* field until this field is *nil*.

6.2 Set A Breakpoint

menu: Debug – Set Breakpoint *shortcut:* Ctrl-b

There are two ways to set a breakpoint: You can select Set Breakpoint from the Debug menu. This will set a breakpoint in the current line (the line the cursor is in).

Or you can click in the tag bar (the area left of the source lines where break points are displayed). Clicking in that area will set a breakpoint at the selected position.



A breakpoint can only be set if the class is compiled. For a compiled class, the tag bar appears white and accepts attempts to set break points. For a

class that has not been compiled the tag bar is grey and does not allow breakpoints to be set.

6.3 Remove A Breakpoint

menu: Debug – Clear Breakpoint *shortcut:* Shift-Ctrl-b

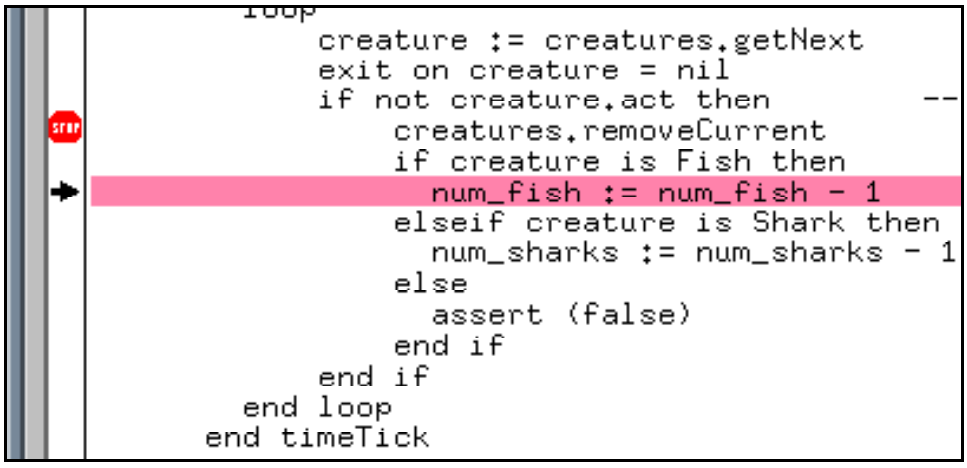
There are two ways to clear a breakpoint: You can select Clear Breakpoint from the Debug menu. This will clear the breakpoint in the current line (the line the cursor is in). If there is no breakpoint in the current line, this function has no effect.

Or you can click directly onto the breakpoint displayed in the tag bar (the area left of the source lines where break points are displayed). This will remove the breakpoint.

6.4 Step Through My Code

To single step through your code, you must first **SET A BREAKPOINT (6.2)**. Once execution reaches the breakpoint, the machine will be interrupted, the source code currently executed will be shown and the execution controls window will be opened automatically. For example:

You can then use the Step or Step Into button to step through your code. The current position in the code will be indicated by an arrow in the tag bar. The arrow points to the instruction that will be executed next – that instruction has not yet been executed.



The Step and Step Into buttons differ only if the next statement is a routine call: Step will execute the whole routine and stop again at the next

instruction after the routine. Step Into will go into the routine and stop at the first instruction within the routine.

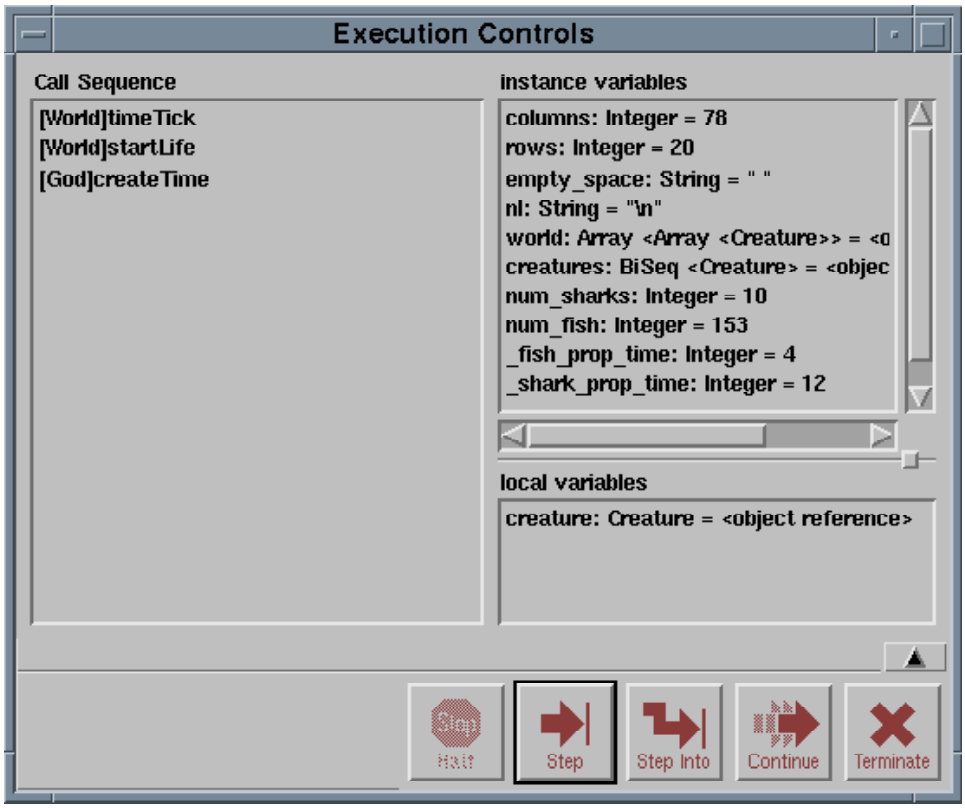
Continue may be used to continue execution normally (until execution finishes or the next breakpoint is reached). Terminate may be used to terminate the execution.

6.5 Inspect Variable Values In My Program

If you want to inspect the value of instance variables between interactive routine calls, see **INSPECT AN OBJECT (6.1)**.

If you want to see the value of local variables or instance variables at a particular moment of the execution, **SET A BREAKPOINT (6.2)** at the place in the code where you want to inspect the variable. Once execution reaches the breakpoint and the execution controls are shown, you can expand the execution controls window with the expand button at the top right corner.

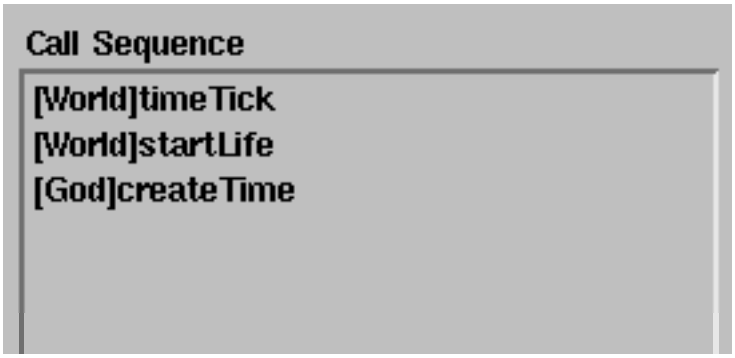
The window will be expanded to a full debugger window that displays current values of interface and local variables.



You can inspect any object listed in the variable list by double clicking it, just as for interactive objects on the object bench, as described in **INSPECT AN OBJECT (6.1)**.

6.6 Find Out About The Call Sequence At A Breakpoint

Once you have **SET A BREAKPOINT (6.2)**, and the machine execution stops, you can find out about the sequence of routine calls that brought you to this particular piece of code. To do this, expand the machine controls window (as described in **INSPECT VARIABLE VALUES IN MY PROGRAM (6.5)**). The left half of the window shows the routine call sequence (currently open routine calls). It presents a kind of stack display:



The routine at the bottom of the list is the one that was interactively called, with the call sequence progressing upwards in the list towards the current routine at the top.

Local and instance variables in intermediate routines can be inspected by selecting the routine in this call sequence list.

6.7 Open The Machine Controls Window

menu: View – Show Machine Controls *shortcut:* Alt-x

The machine controls window opens automatically when the machine hits a breakpoint or is interrupted by the user (e.g. by pressing ESC).

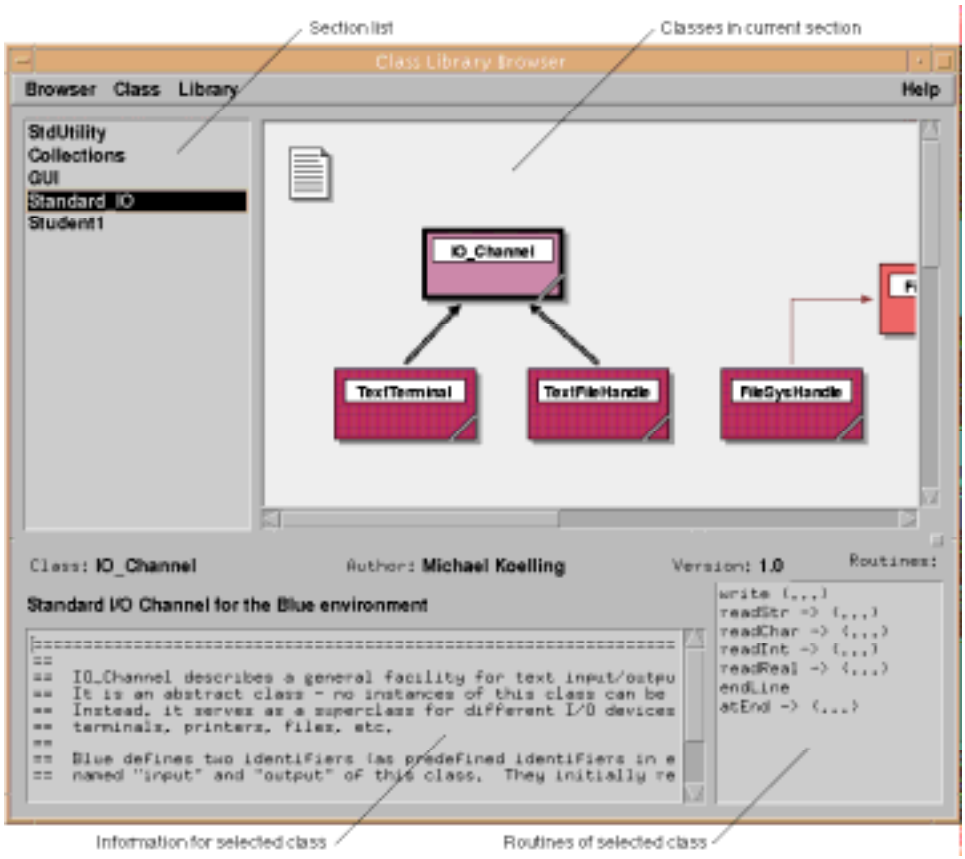
If you want to open or close the machine controls window manually, you can do this by selecting the Show Machine Controls toggle in the view menu.

7. Library Classes

7.1 Find Out What Library Classes Exist

menu: Tools – Library Browser *shortcut:* Alt-b

To find out about available library classes, you can browse a library online from within the Blue environment. To browse a library, you open the library browser (item Library Browser in the Tools menu). The following picture shows the browser main window.



The browser can be used to view different libraries. Initially it will show the Blue Standard Library, but other libraries may be built and used by individual users or groups of users. See **BUILD MY OWN CLASS LIBRARY (7.6)** for details about additional libraries.

Each library is divided into *sections*, which are listed on the left of the main window. Each section contains classes that are related – either perform related functionality or are usually used together. The *Collections* section in the standard library, for instance, contains different sorts of classes to manage collections of objects – lists, stacks, sets, etc.

To browse though the library, first select one of the sections from the *section list*. The classes in that section will be shown in the *class area.*

Then a class may be selected to **SEE THE DETAILS OF A LIBRARY CLASS (7.2)**.

7.2 See The Details Of A Library Class

browser:

menu: Class – Show Class Interface *shortcut:* Alt-e

Details of a class may be obtained in two levels: selecting a class displays details in the *class details* and *routine list* areas (see picture above). Alternatively, a double click on the class icon opens the class and displays the full class interface. Users may or may not be permitted to view the implementation of a library class – they are, however, never permitted to change the source of a library class.

7.3 Use A Library Class In My Project

browser:

menu: Class – Use Class In Project *shortcut:* Alt-u

To use a library class in a project, select the class in the browser window. Then choose Use Class In Project from the Class menu. The class will be inserted into the current project. Should the class depend on other classes (a superclass or used classes) those other classes do not have to be declared as used in the project, unless they are explicitly called by user code.

7.4 Take A Copy From A Library Class

browser:

menu: Class – Copy Class To Project *shortcut:* Alt-???

To copy a library class to a project, select the class in the browser window. Then choose Copy Class To Project from the Class menu. A copy of the class will be inserted into the project. The copy is from then on independent from the original and may be altered and recompiled.

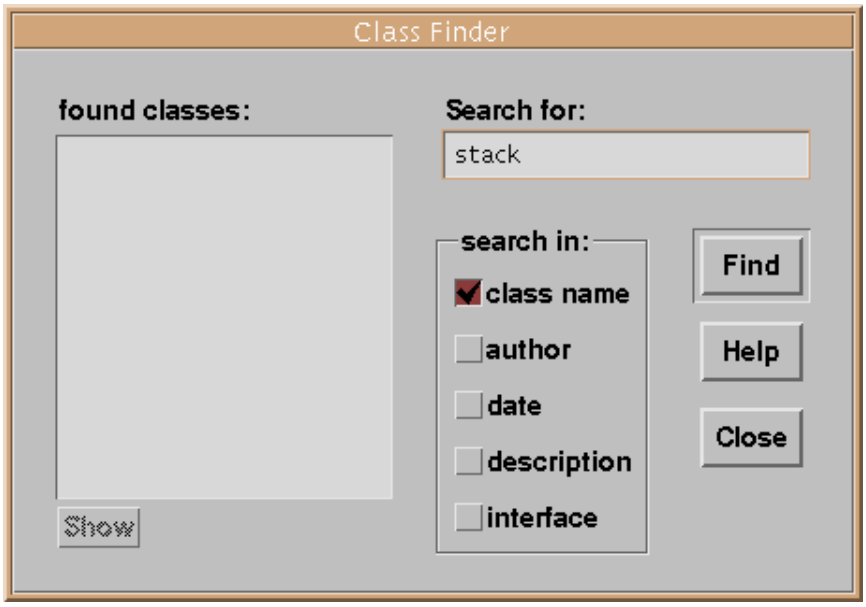
Not all library classes permit read access for the source. Classes can only be copied if read access is granted for the source of the class.

7.5 Search The Class Library

browser:

menu: Class – Find... *shortcut:* Alt-f

If you have some sort of an idea what you are looking for, but you do not know where to find it, you can search the whole class library. Choose Find from the Class menu.



You can type in a search string and specify where the string should appear. Options are:

- in the class name
- in the author string
- in the class comment
- in a routine name
- in a routine comment

You can choose more than one option. After clicking OK, all classes that contain the search string in one or more of the indicated areas are listed in the *found classes list*. Selecting a class name from this list will display its group and select the class in the main window.

7.6 Build My Own Class Library

NOTE: Not Yet Implemented

7.7 Insert Classes Into The Library

NOTE: Not Yet Implemented

8. Miscellaneous

8.1 Use A Project That Is Locked After A Crash

menu: Project – Recover After Crash *shortcut:* —

When a user opens a project, that project is automatically locked (if it is not a group project). In a group project, classes are locked separately and explicitly. Both of these mechanisms are necessary to avoid data corruption.

When the system is terminated unexpectedly (by an outside signal or a system crash) it is possible that locks have not been removed. This prevents further use of the project.

To deal with this situation, **OPEN THE PROJECT (1.2)** and then select Recover After Crash from the Project menu. This will remove all the locks that currently exist. In a group project, all classes will be free. In a single-user project, you will have a lock on the project.

Be careful to use this function *only if you are sure that you are the only user currently in the project.* If other users have the project open at the same time, you will remove their locks and classes may become corrupted or changes may be lost.

8.2 View Interfaces Of Standard Classes

menu: Help – Standard Class Interfaces *shortcut:* —

The Standard Class Interfaces item in the Help menu is itself a menu that contains entries for all predefined classes (Integer, Boolean, Real, String, Enumeration, Array and TextTerminal). By choosing one of these items, you can open the interface of a standard class. This can provide useful information about routines and their signatures.

8.3 Get An Estimate Of The Efficiency Of My Algorithm

menu: Options – Preferences *shortcut:* —

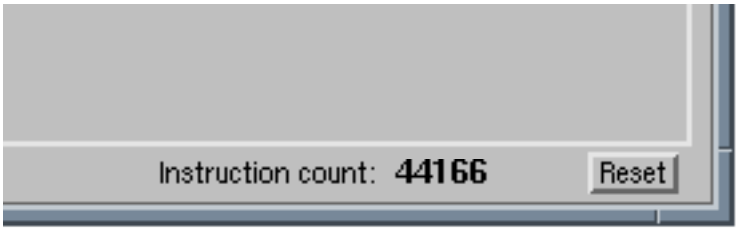
Blue offers a simple mechanism to estimate relative efficiency of algorithms. "Relative" efficiency means that you cannot obtain absolute results, such as "this algorithms takes 12.4 seconds to execute on my machine". Absolute results are, in general, uninteresting anyway.

But you can obtain results such as "this algorithm takes 4 times as long as this other algorithm for the same data set", or "this algorithm is exponential!".

The way this is done in Blue is by counting the number of machine instructions executed. This is only a very rough measure, since not all machine instructions execute in the same time, but in our count they appear as equal. Nonetheless, it is a good enough measure most of the time, since algorithms that are typically compared often use similar instructions.

To use the machine instruction counter, select Preferences from the Options menu. In the preferences dialog you will see an option labelled Show Machine Instruction Counter. Switch this option on and close the preferences dialog.

After doing this, you will see the counter near the bottom right corner of the main window.



The counter is updated every time the machine stops (after every interactive routine execution). It continues to accumulate over several routine calls, but can be reset at any tine with the Reset button. By resetting the counter to zero before the start of an algorithm, the number of machine instructions needed for the execution of the algorithm can be determined.

This can be repeated and compared to the results for other algorithms or the same algorithm with different data sets.

8.4 Show/Hide The Terminal Window

menu: View – Text Terminal *shortcut:* —

To show or hide the Text Terminal toggle the Show Text Terminal item in the View menu. The text terminal is automatically popped up (shown if it was hidden or de-iconified if it was iconified) if output is written to it or input is expected.

8.5 Clear The Screen Of The Text Terminal

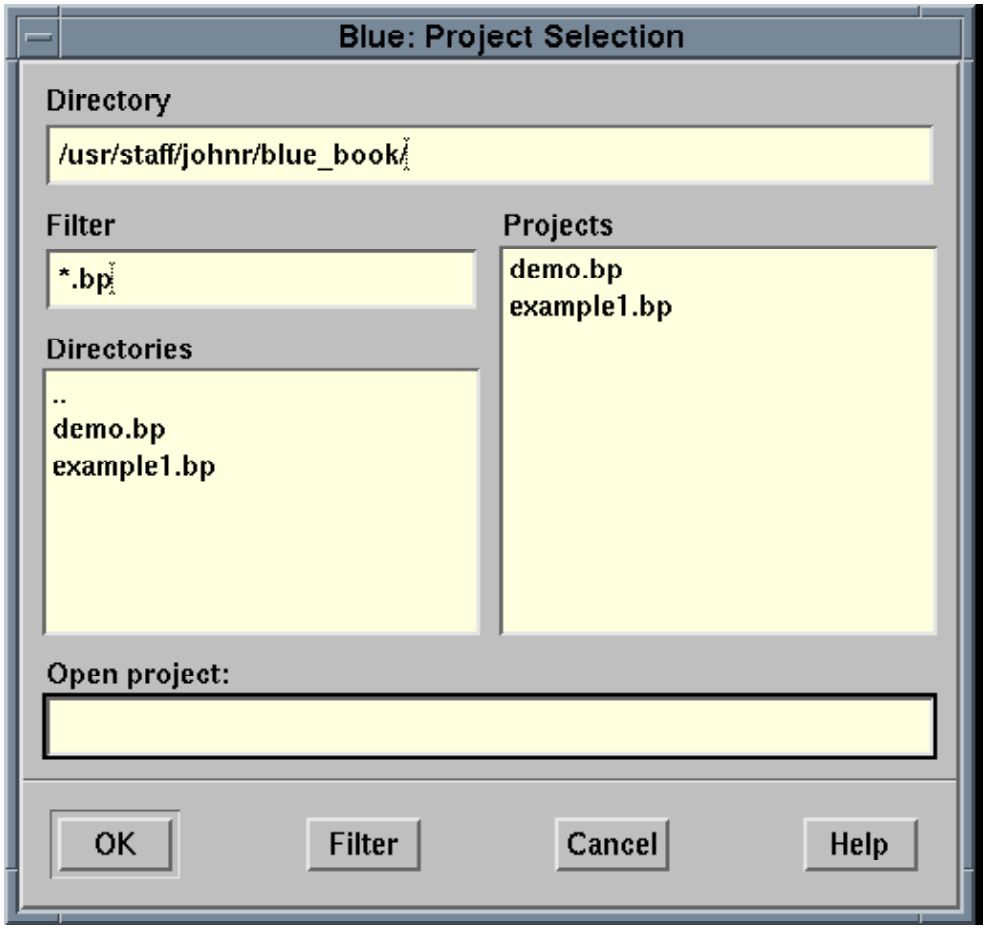
menu: Testing – Clear Terminal *shortcut:* —

To clear the text terminal, select Clear Terminal from the Testing menu.

The terminal can also be cleared by a Blue application. See the interface of the TextTerminal class (see **VIEW INTERFACES OF STANDARD CLASSES (8.2)**) to get the details about the routine interface.

8.6 Use the File Selection Dialog

This dialog looks and behaves slightly differently on different systems – in Windows it looks different than on a Macintosh, which is again different from Unix. On Blue systems for Unix it looks like:



Use the directory list or the directory field to select a directory where you want to create the new Blue project. Type a name for the project in the project name field.

For details about how to use the file selection dialog, see the description of your operating system (the Motif description for Unix systems).

8.7 Start Another Program (Mail Reader, Web Browser, etc.)

menu: Programs – <...> *shortcut:* —

The Programs menu contains entries of programs unrelated to Blue. You can start those programs by selecting them from the menu.

The menu entries can be modified by the system administrator.