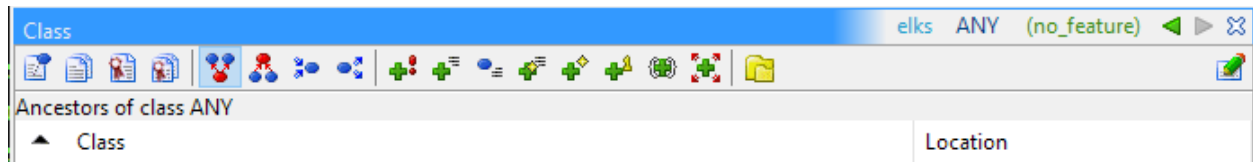


Eiffel Studio - Class Tool

Cliffsnotes on the subject of using the Eiffel Studio “Class” tool



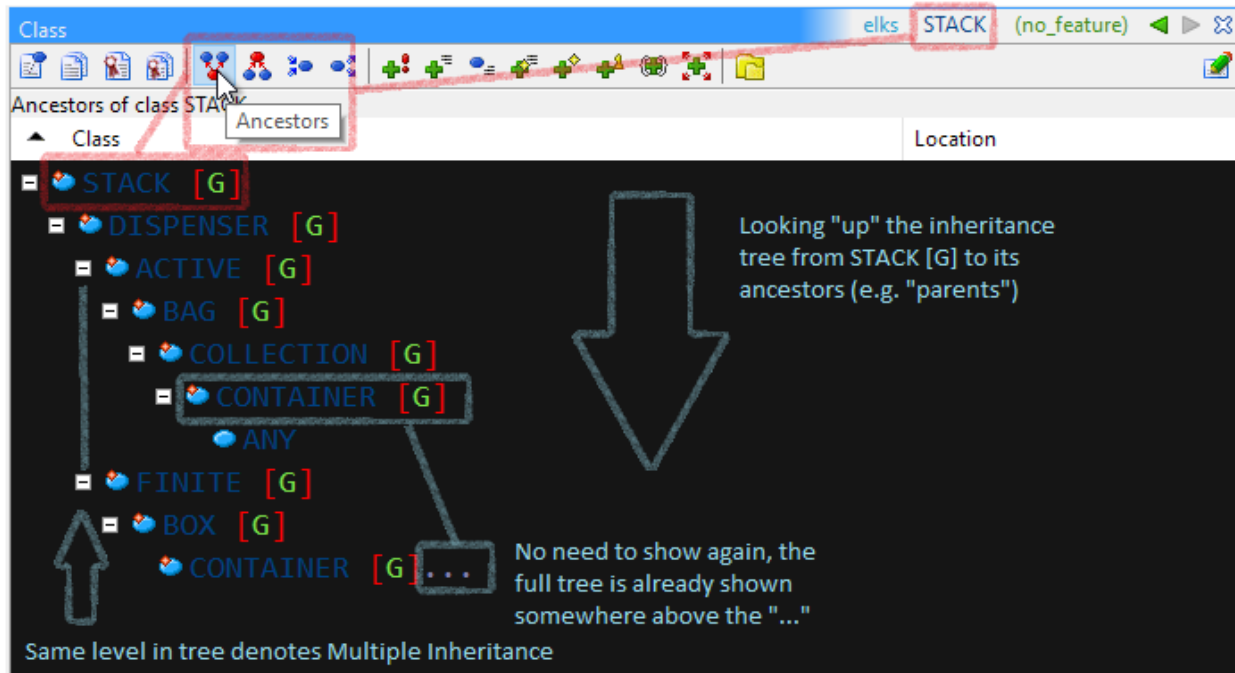
The “Class Tool” is generally (by default) located at the bottom-left of the Eiffel Studio GUI IDE. The tool is used to explore and understand what a class offers to you as you want to use its features and facilities for your own classes. Later, you use the same tool to explore the classes you write in the same way.


The tool is fully Pick-and-Droppable, which means anything you can pick, you may also drop onto an appropriate drop target. Also, the tool itself is a drop-target for many of the other pick-points in the Eiffel Studio IDE GUI. So, feel free to explore and discover.

This Quick Reference is NOT exhaustive by any means. There is much more meat and depth and detail to explore. We invite you to do that exploration more and more as you get comfortable with the tool.

The Class Tool Quick Reference

Ancestors of _____ Tool



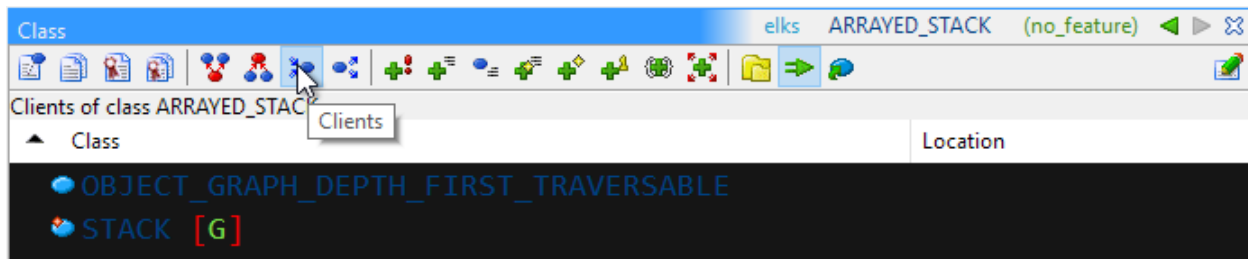
- Easily switch between classes by:
 - Pick-n-Drop a class name directly in the ancestor tool (e.g. PnD {CONTAINER}).
 - At the top of the tool, click into where you see STACK in the example and type a new class name you want to view.
 - Pick-n-Drop a class name from elsewhere in the Eiffel Studio interface.
- Click the  Class column header to resort the list (A-Z or Z-A)



Descendants of _____ Tool

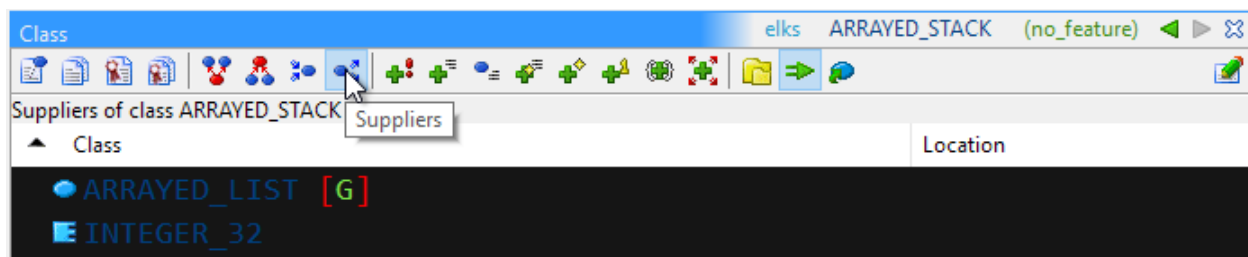
NOTE: Works the same way as the Ancestors tool, but looking down from the selected class instead of up. All the same basic rules for viewing and discovery apply equally to this tool too.

Clients of _____ Tool



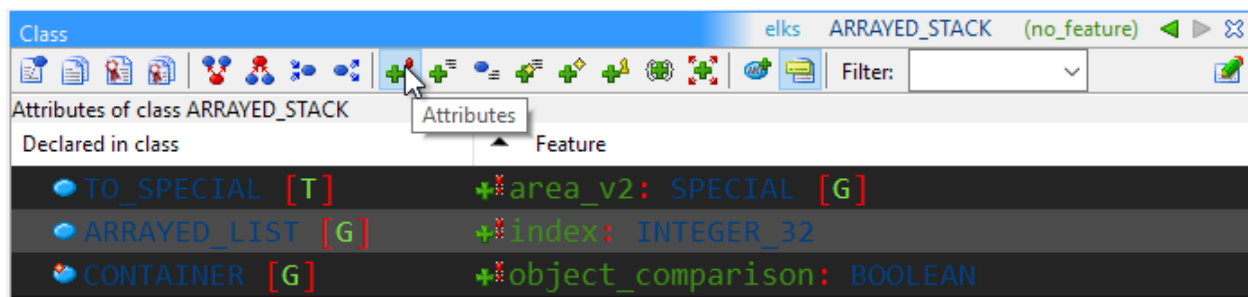
Shows what class in the project universe use the features of the selected class (e.g. ARRAYED_STACK is used by STACK [G] and OBJECT_GRAPH_DEPTH_FIRST_TRAVERSABLE).

Suppliers of _____ Tool



Shows what classes the selected class uses to get its job done (e.g. ARRAYED_STACK uses ARRAYED_LIST and INTEGER_32).

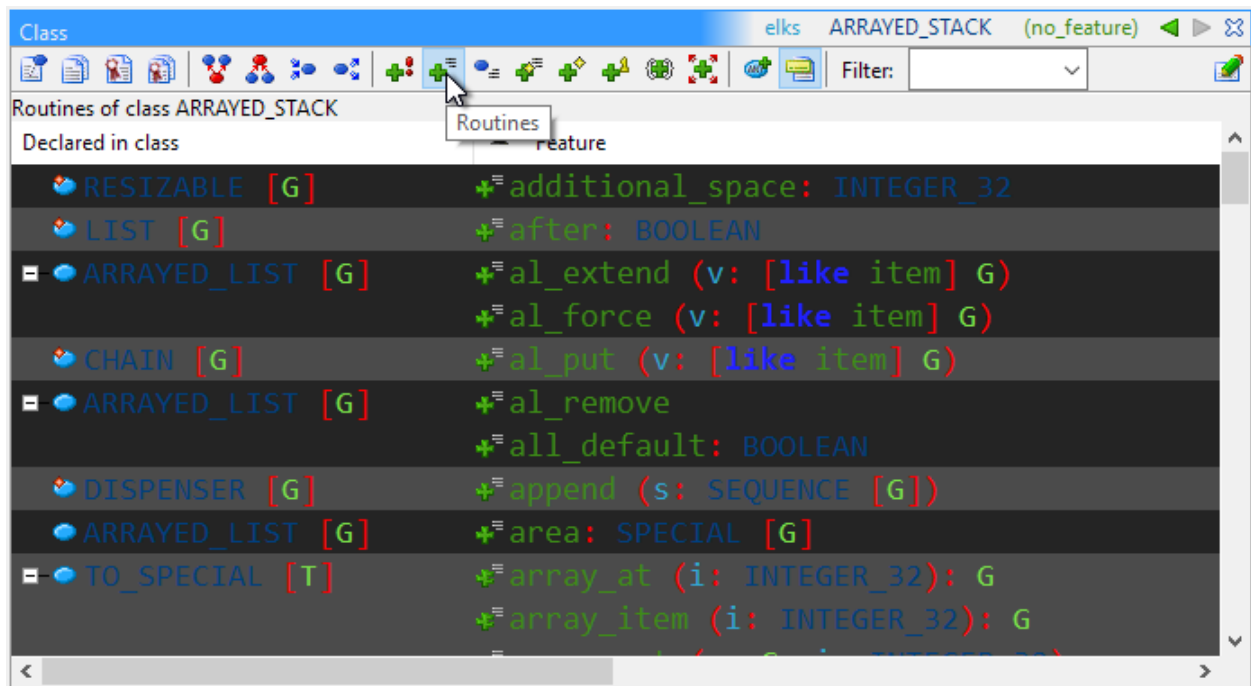
Attributes of _____ Tool



Shows "direct" attributes (e.g. NOT operations, functions, routines, etc.) of the selected class. Note that the list can be sorted by feature name in alpha order, either ascending or descending.

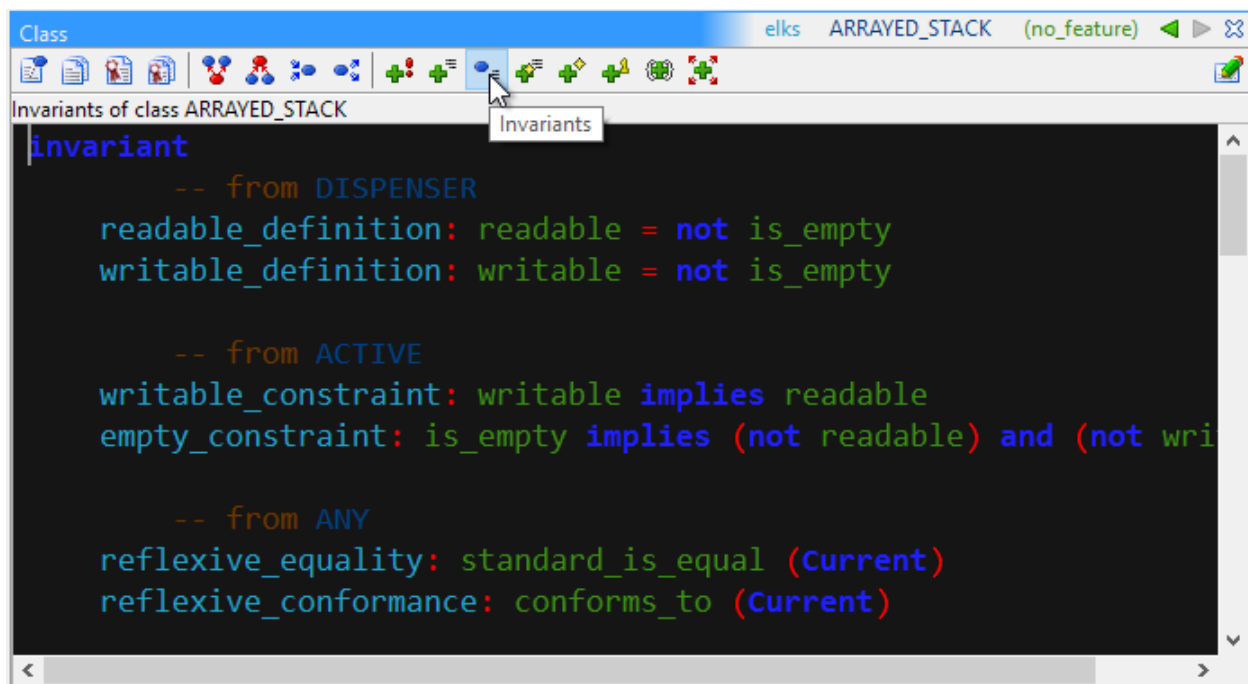
The can also be sorted alpha order by “Declared in class” name, which will group the attributes according to the class in which they are declared (and possibly implemented).

Routines of _____ Tool



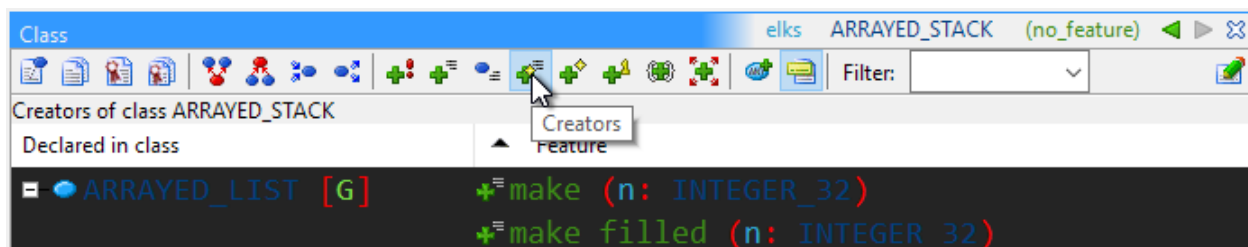
Similar to the “Attributes of _____ Tool” (above), the routines tool shows the routines of the class. Like the attributes tool, you can sort alpha order on feature name or on “Declared in class” name with the same style of discovery (e.g. seeing where a routine came from).

Invariants of _____ Tool



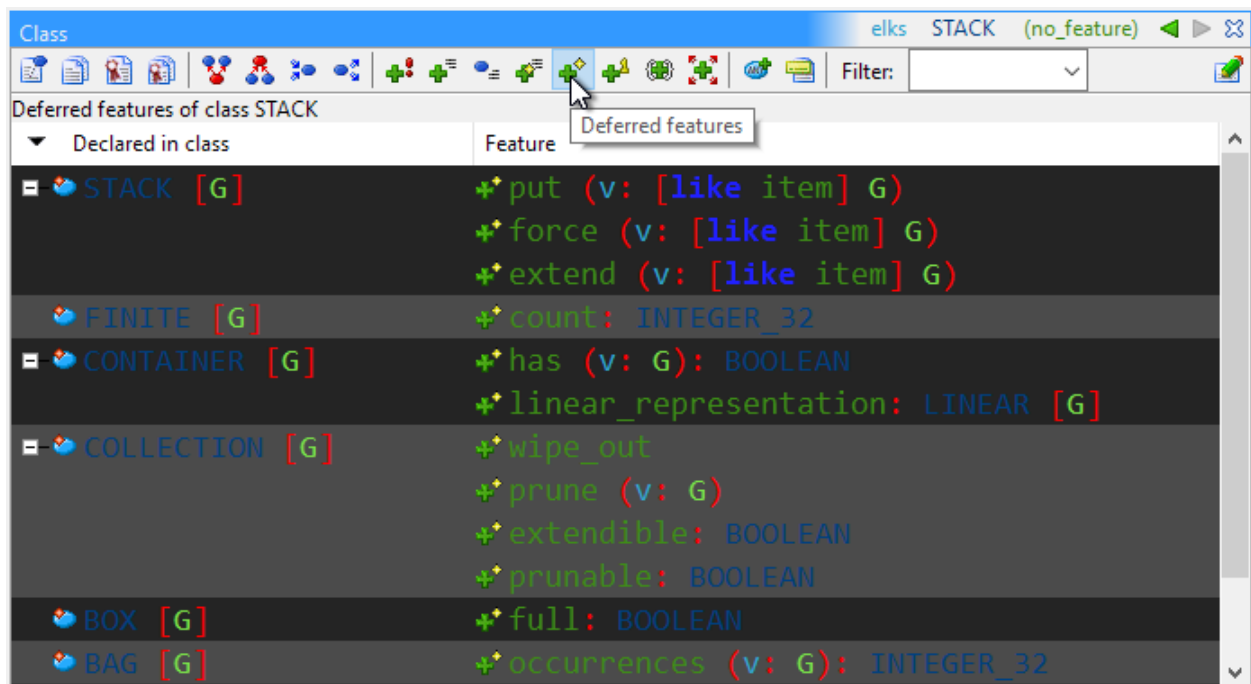
The invariants tool will show all of the invariants of the class and where the invariant was declared (e.g. what ancestor class has the invariant, which is inherited by the selected target class). This tool is handy, especially on your own classes: It will reveal to you what a “stable class state” looks like for your class and you can reason with it to ensure the specification of invariants is in fact correct!

Creators of _____ Tool



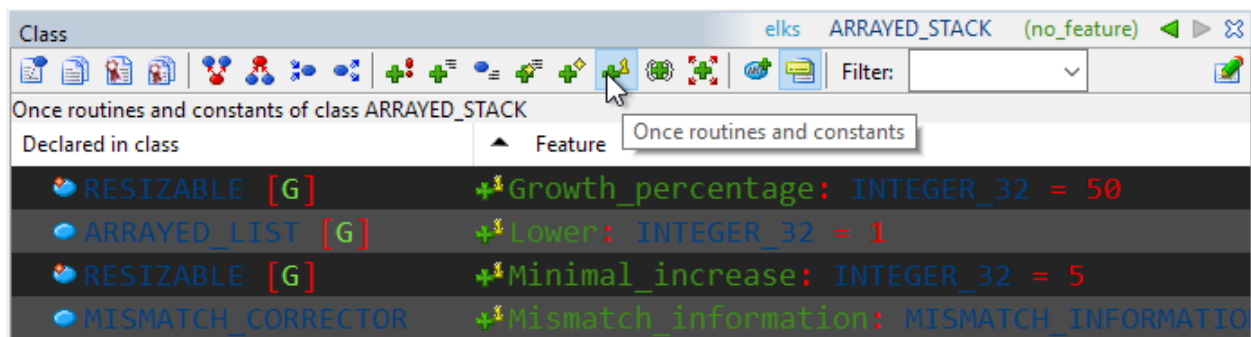
If the selected class is NOT “deferred”, then it will have one or more creation routines. This tool allows you to see what creators have been declared and by what class in the ancestry of it. In the example above, ARRAYED_STACK inherits two creators from ARRAYED_LIST: make and make_filled.

Deferred features of ____ Tool



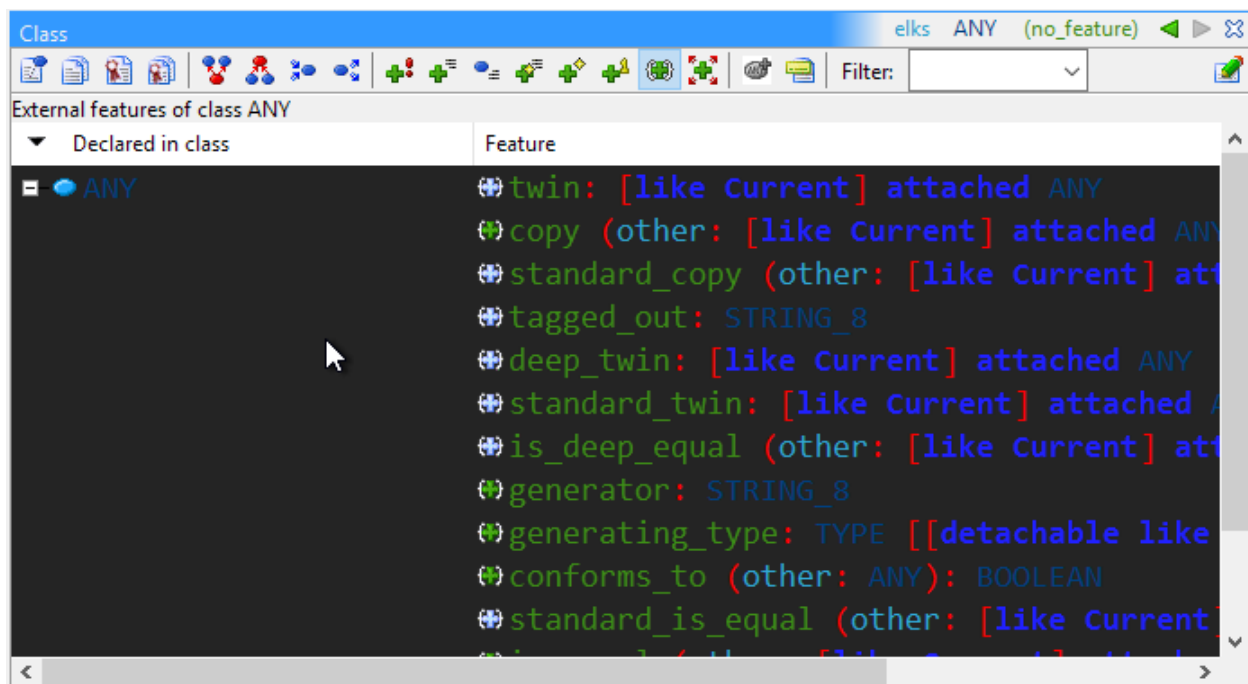
ONLY IF a class is “deferred” will this tool work. Non-deferred classes cannot have deferred features, so this list will always be empty for those classes. Alpha sorting rules apply.



Once routines and constants of ____ Tool



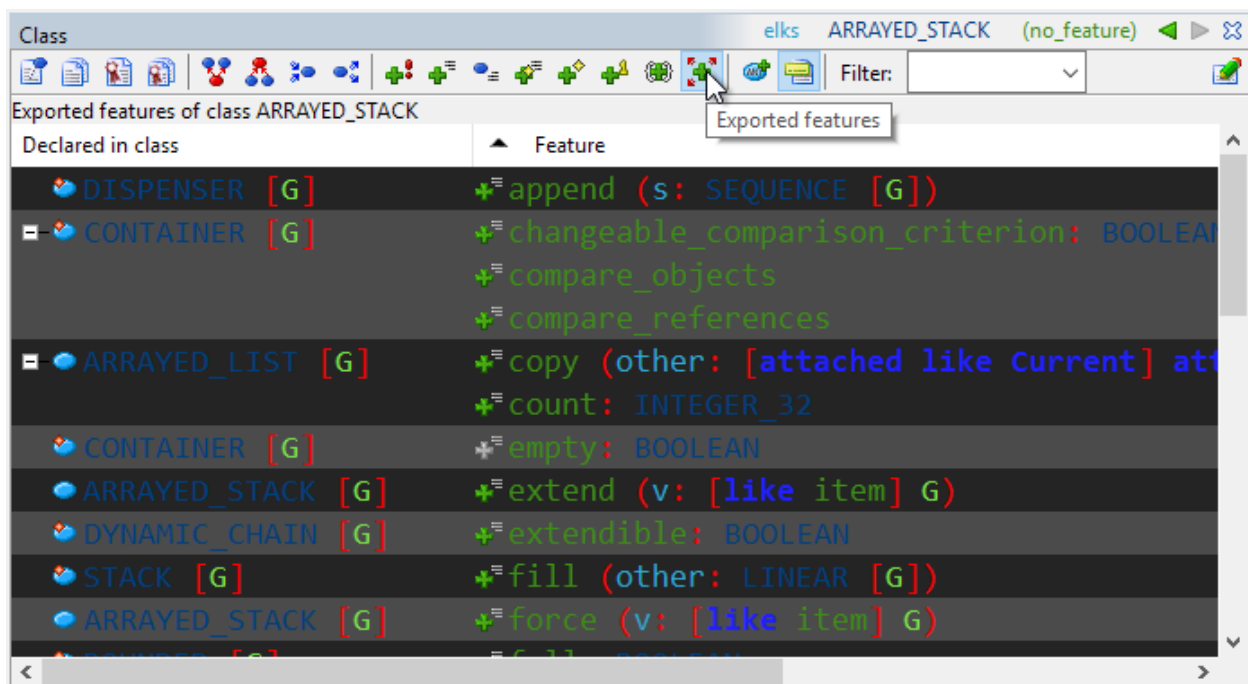
Shows all routines declared as “once’s” or as “constants” and in what ancestor class the declaration happens. Like other lists, this can be sorted alpha ascending or descending by feature name or “Declared in class” name.

External features of _____ Tool



Shows features whose implementation is a call to either some external C/C++ library or has C/C++ coded “in-line” in the feature itself. The tool can also show “built in” features—that is—features whose code is a part of the Eiffel “Run-time”, which gets “compiled into” any EXE or DLL that you “finalize”. Features with a  symbol are available to be redefined (modified) in descendant classes, Whereas, features with a  symbol are “frozen” in the ancestor and cannot be redefined or otherwise modified in a descendant.

Exported features of ____ Tool



Shows all features that are exported to {ANY}. Feature groups exported to {NONE} or a {CLIENT, LIST} will not be shown in this list. As above, these columns may be alpha sorted as needed.