SortedListAgent

Adopted By:	ClassAgent
	StringAgent
	Others, used to implement various inspectors in TTools.
Declared In:	SortedListAgent.h

Protocol Description

SortedListAgent is a protocol describing the methods that a SortedList's *agent* is expected to respond to.   These methods encompass functionality that is specific to a particular set of objects that will be kept in a SortedList.   This is why, for instance, the agent for a list of String objects is an instance of the StringAgent class.

The functionality that the protocol requires relates to two things: sorting and hierarchical display.   Only one method in the protocol is necessary to implement sorting; that is the **compare:with:sender:** method.   The other methods are all concerned with hierarchical display in browsers.   Most of the methods carry a sender with them so that the agent may have some context in which to consider it's task; for instance, an object might be the agent for two different SortedLists, and might want to respond differently depending on what the sender was.

Method Types

Sorting   - compare:with:sender:

Hierarchical display   - displayStringFor:sender:
                                   - titleOfColumn:
                                   - isLeaf:sender:
                                   - subdirectoryFor:sender:

Instance Methods

compare:with:sender:
    - (int)**compare:**_first_ **with:**_second_ **sender:**_sender_

Returns a negative number to indicate that _first < second_; returns 0 to indicate that _first==second_.   Returns a positive number to indicate that _first>second_.   For instance, StringAgent's implementation just returns strcmp([first stringValue], [second stringValue]).   If you can bend your mind around it, consider that ClassAgent's implementation of this method returns strcmp([first name], [second name]).   Note the use of class methods - _first_ and _second_ can be any object, even a class object.

displayStringFor:sender:
    - (const char *)**displayStringFor:**_anObject_ **sender:**_sender_

Returns the string to use while displaying _anObject_ in a browser.   For instance, StringAgent just returns [anObject stringValue].   ClassAgent returns [anObject name].

isLeaf:sender:
- **-** (BOOL)**isLeaf:***anObject* **sender:***sender*

Returns YES to indicate that *anObject* has no sub-nodes in the file browser.   Returns NO to indicate that it does.
StringAgent returns YES always.   ClassAgent uses the ClassAdditions category of object to determine if the class *anObject*
has no subclasses.   (The ClassAdditions category uses run-time functions and structures to glean this information).

subdirectoryFor:sender:
- **-** (id)**subdirectoryFor:***anObject*   **sender:***sender*

Returns a SortedList (or subclass thereof) containing the sub-nodes of *anObject*.   In StringAgent, this is never called, so it
just returns nil.   In ClassAgent, the ClassAdditions category of Object is used to populate a privately-held list with the
subclasses of the class *anObject.*

titleOfColumn:
- **-** (const char *)**titleOfColumn:**(int)*col*

Returns a read-only string indicating the title of the next browser column.   Note that this works only if the browser is set up
to ask for titles.