MiscSortedStorage

Inherits From:	MiscStorage : Storage : Element
Declared In:	MiscSortedStorage.h

Class Description

This class adds sorting capabilities to a normal Storage.   It implements the sorting by using two methods from the MiscCompare protocol: compare: and compare:ignoreCase:.   Whatever elements that are placed in a MiscSortedStorage should conform to the MiscCompare protocol, but the only method they need to implement is the compare: method.   If case comparisons are going to be made (ie: between string objects), then the compare:ignoreCase: method also needs to be implemented.   To improve speed of adding objects, this class just checks to see if an object conforms to the MiscCompare protocol and does not test to see if it actually can respond to whatever method it needs.

Alternatively, if more complicated sorting behavior is wanted, this class can be subclassed and the compare:to:caseCheck: method overridden.   This method is what is called for all methods in the class and is also the only place the MiscCompare routines are called, thus isolating them from the rest of the class.

A number of object inserting methods from the **Storage** class are overridden so that they perform correctly accordingly with the sorting capabilities of this class.   These methods will perform like the original methods when sorting is not enabled. They will also return the same values as the original methods.

Instance Variables

BOOL **ignoreCase**
int **sortOrder**

ignoreCase   Flag to tell whether or not the sorting should ignore case.   This value can be interpreted differently for different types of objects, but is mainly for objects which store strings.

sortOrder   Value to determine whether sorting is ascending or descending.

Method Types

Initializing a new instance   **-** initCount:

Copying a MiscSortedStorage   **-** copyFromZone:

Manipulating objects by index **-** insertElement:at:
- replaceElementAt:with:

Manipulating objects by **id**
- addElement:
- addElementIfAbsent:
± insertElementBySort:
- replaceElement:with:
- indexOf:

Combining MiscSortedStorages
- appendStorage:

Checking the state of an instance
- ignoreCase
- sorted
- sortEnabled
- sortOrder

Setting the state of an instance
- setIgnoreCase:
- setSortEnabled:
- setSortOrder:

Methods for sorting
- compare:to:caseCheck:
- sort

Archiving
- read:
- write:

Class Methods

initialize
    + initialize

Initializes the class, setting the version number of the class.

Instance Methods

addElement:
    - addElement:*anElement*

Performs the same as the Storage version except if sorting is enabled, it will put *anElement* in sorted order.

See also: **- insertElement:at:**, **- appendStorage:**, **- addElement:** (Storage)

compare:to:caseCheck:
    - compare:*objectA* **to:***objectB* **caseCheck:**(BOOL)*flag*

This is the actual comparison routine between two objects. It will return -1, 0, or 1 depending if *objectA* is less than, greater than, or equal to *objectB* respectively. The *flag* parameter tells this routine that it is okay to have a case-sensitive comparison. This routine uses two methods from the MiscCompare protocol. It uses **compare:** if *flag* is NO or if the instance is not to ignore case. It uses **compare:ignoreCase:** if both *flag* is YES and the instance is to ignore case.

See also: **- ignoreCase**, ± **compare:** (MiscCompare), ± **compare:ignoreCase** (MiscCompare)

copyFromZone:
    - copyFromZone:(NXZone *)*zone*

Returns a new MiscSortedStorage object with the same contents as the receiver.   The objects in the MiscSortedStorage aren't copied; therefore, both MiscSortedStorage contain pointers to the same set of objects.   Memory for the new MiscSortedStorage is allocated from *zone*.

See also: **- copy** (Element)

ignoreCase
    - (BOOL)**ignoreCase**

Returns whether or not sorting will ignore case when doing comparisons.

initCount:elementSize:description
    - initCount:(unsigned int)*numSlots* **elementSize:**(unsigned int)*sizeInBytes* **initCount:**(const char *)*string*

Initializes the receiver, a new MiscSortedStorage object, by allocating enough memory for it to hold *numSlots* objects. Returns **self**.

This method is the designated initializer for the class.   It should be used immediately after memory for the MiscSortedStorage has been allocated and before any objects have been assigned to it; it shouldn't be used to reinitialize a MiscSortedStorage that's already in use.

See also: **- capacity** (Storage)

insertElementBySort:
    - insertElementBySort:(void *)*anElement*

This is the only method for inserting objects into a MiscSortedStorage. All other inserting methods call this one to add objects to the Storage. Normally this method need not be used to add objects, but it is faster than the other methods when the Storage is to be in sorted order since this method does not test to see if it should place an object in sorted order or not.

See also: **- count** (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

insertElement:at:
    - insert Element**:***an*Element **at:**(unsigned int)*index*

Performs the same as the Storage version except it places *an*Element in sorted order if sorting is enabled and ignores *index*.

See also: **- count** (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

read:
    - read:(NXTypedStream *)*stream*

Reads the MiscSortedStorage and all the objects it contains from the typed stream *stream.*

See also: **- write:**

replaceElementAt:with:
   - replace Element**At:**(unsigned int)*index* **with:***new*Element

Performs the same as the Storage version except it places *an*Element in sorted order if sorting is enabled and ignores *index*.

See also: - replace Element**:with:**, **- replace**Element**At:with:** (Storage)

setIgnoreCase:
   - setIgnoreCase:(BOOL)*flag*

Sets the case comparison flag to *flag* and sorts the Storage if the value changed.

See also: - count (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

setSortEnabled:
   - setSortEnabled:(BOOL)*flag*

Sets sorting to be either on or off according to *flag* and sorts the Storage if it currently isn't sorted.

See also: - count (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

setSortOrder:
   - setSortOrder:(int)*order*

Sets the sort order according to *order* and reorders the Storage if the value changed.   This value should be Misc_ASCENDING or Misc_DESCENDING.   If the Storage is currently sorted this method does not use the **sort** method to reorder the Storage, but just moves the objects so that they are in reverse order.   Otherwise it will use the **sort** method.

See also: **- count** (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

sort
   - sort

Sorts the Storage if it isn't currently sorted   It uses a QuickSort with an Insertion Sort to handle small partitions.   This method should perform well for all kinds of data order.

See also: **- count** (Storage), **- addElement:**, **- insertElement:at:** (Storage)

sorted
   - (BOOL)sorted

Returns whether or not the Storage instance is sorted.

See also: **- count** (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

sortEnabled
 - (BOOL)**sortEnabled**

Returns whether sorting is enabled.

See also: **- count** (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

sortOrder
 - (int)**sortOrder**

Returns the order of how elements are going to be sorted.

See also: **- count** (Storage), **- add**Element**:**, **- insert**Element**:at:** (Storage)

write:
 - write:(NXTypedStream *)*stream*

Writes the MiscSortedStorage, including all the objects it contains, to the typed stream *stream*.

See also: **- read:**

Constants and Defined Types

```
#define Misc_ASCENDING     1
#define Misc_DESCENDING   -1
```