

# SortedStorage

<b>Inherits From:</b>	Storage : Object
<b>Declared In:</b>	SortedStorage.h

## **Class Description**

The SortedStorage class is a subclass of Storage that inserts in sorted order, according to a method in it's agent. In addition, the SortedStorage class implements Browser delegation, so that it can automatically communicate with a browser in such a way as to display itself properly when requested to. This class is the Storage analog for the SortedList class - what SortedList does for objects, SortedStorage does for arbitrary types.

## **Instance Variables**

id agent;

agent

An object responsible for specific tasks relating to the elements kept in the storage - namely displaying, comparing, and determining leaf and subdirectory status for particular elements. This functionality is described by the SortedStorageAgent protocol.

## Method Types

Sorting

addElement:  
sort:

Element-specific info

displayStringForElementAt:  
compareElementAt:with:

	isLeafAt: subdirectoryForElementAt:
Browser delegation	browser:fillMatrix:inColumn:
Archiving/Unarchiving	read: write:

## Instance Methods

**addElement:**  
- **addElement:***anElement*

Binarily determines where to insert *anElement* into the receiver, using **insertElement:at:** or **addElement:** when the correct location is found. Uses the **compareElementAt:with:** method to determine pivots in the binary

search.

## **agent**

- **agent**

Returns the receiver's *agent*, an Element that should respond to the SortedStorageAgent protocol.

## **browser:fillMatrix:inColumn:**

- **browser:sender** **fillMatrix:matrix** **inColumn:(int)column**

Browser's delegate routine; fills *matrix* in *browser* with information from the receiver, or, the subdirectory of the receiver corresponding to *column*. Subdirectories are obtained by calls to **subdirectoryForElementAt:** in the receiver. Setting **isLeaf:** on the browser cells in *matrix* is determined by calls to **isLeafAt:** in the receiver. Returns the number of cells created and added to *matrix*.

**compareElementAt:with:**

- (int)**compareElementAt:(int)at with:element**

Using the agent method **compare:with:sender:**, returns an integer which, if negative, indicates the *at* element is less than *element*; if zero, indicates the *at* element is equal to *element*; if positive, indicates the *at* element is greater than *element*.

**displayStringForElementAt:**

- (const char \*)**displayStringForElementAt:(int)at**

Returns a read-only string that is typically used to display the *at* element in a browser. The agent **displayStringFor:sender:** method is used to obtain this string.

**isLeafAt:**

- (BOOL)**isLeafAt:(int)at**

Returns essentially [agent isLeaf:[self elementAt:at] sender:self].

**read:**

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

Reads an instance of the timer from *stream*.

**setAgent:**

- **setAgent:***anAgent*

Set's the *agent* instance variable, an object that should respond to the SortedListAgent protocol.

**sort:**

- **sort:***sender*

Sorts the receiver according to the **compareElementAt:with:** method.

Implementation details: copies list, empties self, iteratively calls [self addElement:].

**subdirectoryForElementAt:**

- **subdirectoryForElementAt:**(int)*at*

Returns the SortedStorage object, or nil, which is the subdirectory of the element at *at* in the receiver. Determines this by calling the agent method **subdirectoryFor:sender:.**

**write:**

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

Writes the timer to *stream*.