PriorityQueue

Inherits From: Object

Declared In: "PriorityQueue.h"

Class Description

A PriorityQueue is a collection of objects, ordered by unsigned integer priority. The class provides an interface that permits queueing, dequeueing, and counting of objects in a PriorityQueue.

PriorityQueues grow dynamically when new objects are added. The default mechanism automatically doubles the capacity of the PriorityQueue when it becomes full.

Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

Instance Variables

NODE ***heap**;
unsigned int **top**;
unsigned int **size**;

heap	The heap managed by the PriorityQueue object.
top	The actual number of objects in the heap.
size	The total number of objects that can fit in currently allocated memory.

Method Types

Initializing a new PriorityQueue object	± init ± initCount:
Copying and freeing a PriorityQueue	± copyFromZone: ± free
Queueing and dequeueing objects by priority	± addObject:withPriority: ± removeObject
Returning highest priority in a PriorityQueue	± highestPriority
Counting objects	± count

Comparing and combining PriorityQueues	± isEqual:
	± appendQueue:
Emptying a PriorityQueue	± empty
	± freeObjects
Getting and setting the capacity of a PriorityQueue	± capacity
	± setAvailableCapacity:
Sending messages to the objects	± makeObjectsPerform:
	± makeObjectsPerform:with:
Archiving	± read:
	± write:

Instance Methods

addObject:withPriority:

± **addObject:***anObject* **withPriority:**(unsigned)*aPriority*

Inserts *anObject* into the priority Queue with priority *aPriority*, and returns **self**. However, if *anObject* is **nil**, nothing is inserterted and **nil** is returned.

See also: ± **insertObject:at:**, ± **appendList:**

appendQueue:

± **appendQueue:**(PriorityQueue *)*otherQueue*

Inserts all the objects in *otherQueue* into the receiving Queue, and returns **self**.

See also: ± **addObject:withPriority:**

capacity
± (unsigned int)**capacity**

Returns the maximum number of objects that can be stored in the Queue without allocating more memory for it. When new memory is allocated, it's taken from the same zone that was specified when the Queue was created.

See also: ± **count**, ± **setAvailableCapacity:**

copyFromZone:
± **copyFromZone:(NXZone *)***zone*

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

See also: ± **copy** (Object)

count
± (unsigned int)**count**

Returns the number of objects currently in the PriorityQueue.

See also: ± **capacity**

empty
± **empty**

Empties the Queue of all its objects without freeing them, and returns **self**. The current capacity of the Queue isn't changed.

See also: ± **freeObjects**

free

± **free**

Deallocates the Queue object and the memory it allocated for the array of object **id**s. However, the objects themselves aren't freed.

See also: ± **freeObjects**

freeObjects

± **freeObjects**

Removes every object from the PriorityQueue, sends each one of them a **free** message, and returns **self**. The Queue object itself isn't freed and its current capacity isn't altered.

The methods that free the objects shouldn't have the side effect of modifying the Queue.

See also: ± **empty**

init

± **init**

Initializes the receiver, a new PriorityQueue object, and allocates memory for its array of object **id**s. It's initial capacity will be 1. Minimal amounts of memory will be allocated when objects are added to the Queue. Or an initial capacity can be set, before objects are added, using the **setAvailableCapacity:** method. Returns **self**.

See also: ± **initCount:**, ± **setAvailableCapacity:**

initCount:

± **initCount:**(unsigned int)*numSlots*

Initializes the receiver, a new PriorityQueue 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 PriorityQueue has been allocated and before any objects have been assigned to it; it shouldn't be used to reinitialize a Queue that's already in use.

See also: ± **capacity**

isEqual:

± (BOOL)**isEqual:***anObject*

Compares the receiving PriorityQueue to *anObject*. If *anObject* is a PriorityQueue with exactly the same contents as the receiver, this method returns YES. If not, it returns NO.

Two Queues have the same contents if they each hold the same number of objects and the **id**s in each Queue are identical and occur in the same order. Whether objects in both queues have identical priorities has no bearing on the test for equality.

makeObjectsPerform:

± **makeObjectsPerform:**(SEL)*aSelector*

Sends an *aSelector* message to each object in the PriorityQueue in reverse order (starting with the last object and continuing backwards through the Queue to the first object), and returns **self**. The *aSelector* method must be one that takes no arguments. It shouldn't have the side effect of modifying the Queue.

makeObjectsPerform:with:

± **makeObjectsPerform:**(SEL)*aSelector* **with:***anObject*

Sends an *aSelector* message to each object in the PriorityQueue in reverse order (starting with the last object and continuing backwards through the Queue to the first object), and returns **self**. The message is sent each time with *anObject* as an argument, so the *aSelector* method must be one that takes a single argument of type **id**. The *aSelector* method shouldn't, as a side effect, modify the Queue.

read:

± **read:**(NXTypedStream *)*stream*

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

See also: ± **write:**

setAvailableCapacity:

± **setAvailableCapacity:**(unsigned int)*numSlots*

Sets the storage capacity of the PriorityQueue to at least *numSlots* objects and returns **self**. However, if the Queue already contains more than *numSlots* objects (if the **count** method returns a number greater than *numSlots*), its capacity is left unchanged and **nil** is returned.

See also: ± **capacity**, ± **count**

write:

± **write:**(NXTypedStream *)*stream*

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

See also: ± **read:**