init
initCount:
Copying and freeing a List copyFromZone:
free
Manipulating objects by index insertObject:at:
addObject:
removeObjectAt:
removeLastObject
replaceObjectAt:with:
objectAt:
lastObject
count
Manipulating objects by id addObject:
addObjectIfAbsent:
removeObject:
replaceObject:with:
indexOf:
Comparing and combining Lists
isEqual:
appendList:
Emptying a List empty
freeObjects
Sending messages to the objects
makeObjectsPerform:
makeObjectsPerform:with:
Managing the storage capacity capacity
setAvailableCapacity:
Archiving read:
write:

addObjectIfAbsent:anObject

Inserts anObject at the end of the List and returns self, provided that anObject isn't already in the L the List, it won't be inserted, but self is still returned.

If anObject is nil, nothing is inserted and nil is returned.

insertObject:at:

appendList:(List *)otherList

Inserts all the objects in otherList at the end of the receiving List, and returns self. The ordering of maintained.

addObject:

(unsigned int)capacity

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

count, setAvailableCapacity:

copyFromZone:(NXZone *)zone

Returns a new List object with the same contents as the receiver. The objects in the List aren't cop Lists contain pointers to the same set of objects. Memory for the new List is allocated from zone.

copy (Object)

(unsigned int)count

Returns the number of objects currently in the List.

capacity

empty

Empties the List of all its objects without freeing them, and returns self. The current capacity of th

freeObjects

free

Deallocates the List object and the memory it allocated for the array of object ids. However, the ob freed.

empty

(unsigned int)indexOf:anObject

Returns the index of the first occurrence of anObject in the List, or NX_NOT_IN_LIST if anObjec

init

Initializes the receiver, a new List object, but doesn't allocate any memory for its array of object id
will be 0. Minimal amounts of memory will be allocated when objects are added to the List. Or a
set, before objects are added, using the setAvailableCapacity: method. Returns self.

initCount:, setAvailableCapacity:

initCount:(unsigned int)numSlots

Initializes the receiver, a new List object, by allocating enough memory for it to hold numSlots obj

This method is the designated initializer for the class. It should be used immediately after memory
allocated and before any objects have been assigned to it it shouldn't be used to reinitialize a List t

capacity

insertObject:anObject at:(unsigned int)index

Inserts anObject into the List at index, moving objects down one slot to make room. If index equa
the count method, anObject is inserted at the end of the List. However, the insertion fails if index
returned by count or anObject is nil.

If anObject is successfully inserted into the List, this method returns self. If not, it returns nil.

count, addObject:

(BOOL)isEqual:anObject

Compares the receiving List to anObject. If anObject is a List with exactly the same contents as th
returns YES. If not, it returns NO.

Two Lists have the same contents if they each hold the same number of objects and the ids in each
occur in the same order.

lastObject

Returns the last object in the List, or nil if there are no objects in the List. This method doesn't rer
returned.

makeObjectsPerform:(SEL)aSelector with:anObject

Sends an aSelector message to each object in the List in reverse order (starting with the last object backwards through the List to the first object), and returns self. The message is sent each time wit argument, so the aSelector method must be one that takes a single argument of type id. The aSelec as a side effect, modify the List.

objectAt:(unsigned int)index

Returns the id of the object located at slot index, or nil if index is beyond the end of the List.

count

read:(NXTypedStream *)stream

Reads the List and all the objects it contains from the typed stream stream.

write:

removeLastObject

Removes the object occupying the last position in the List and returns it. If there are no objects in returns nil.

lastObject, removeObjectAt:

removeObject:anObject

Removes the first occurrence of anObject from the List, and returns it. If anObject isn't in the List nil.

The positions of the remaining objects in the List are adjusted so there's no gap.

removeLastObject, removeObjectAt:

removeObjectAt:(unsigned int)index

Removes the object located at index and returns it. If there's no object at index, this method return

The positions of the remaining objects in the List are adjusted so there's no gap.

removeLastObject, removeObject:

replaceObjectAt:(unsigned int)index with:newObject

Returns the object at index after replacing it with newObject. If there's no object at index or newO
replaced and nil is returned.

replaceObject:with:

setAvailableCapacity:(unsigned int)numSlots

Sets the storage capacity of the List to at least numSlots objects and returns self. However, if the L
more than numSlots objects (if the count method returns a number greater than numSlots), its capa
and nil is returned.

capacity, count

write:(NXTypedStream *)stream

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

read: