FCUnsortedCollection

Inherits From:	FCOrderedCollection : FCCollection : Object
Declared In:	FCUnsortedCollection.h

Class Description

FCUnsortedCollection is a subclass of FCOrderedCollection that implements the behavior of an ordered collection which doesn't automatically sort itself.

This restriction expands the available methods for manipulating the collection; for instance, it makes no sense to be able to insert an object at a particular position (see - **insertObject:at:** , below) if the collection is just going to move that object into sorted order.

All collection subclasses which require an unsorted collection are subclasses of this class. This includes lists, stacks, queues, and ordered sets.

FCUnsortedCollection is an abstract superclass.   You cannot instantiate it directly; in fact, some of its methods are simply stubs in the superclass and return errors when invoked.   Its basic purpose is to provide common methods and an orthogonal interface to its six instantiable subclasses, all of which fully adhere to the interface described here.   In the documentation below, the term "collection" refers to any non-abstract subclass of FCUnsortedCollection.

FCUnsortedCollection inherits from FCOrderedCollection and FCCollection. The interface documented here only covers the methods that are new or different in FCUnsortedCollection, but all the methods in FCCollection and FCOrderedCollection will work on an FCUnsortedCollection as well.   Refer to the documentation on those classes to complete the description of an FCUnsortedCollection.

Instance Variables

Inherited from Object
None declared in this class.

Inherited from FCCollection
id **_fc_contents** ;
Class **_fc_class** ;
SEL **_fc_sortSelector** ;
BOOL **_fc_archiveByReference** ;

Inherited from FCOrderedCollection
None declared in this class.

Declared in FCUnsortedCollection
None declared in this class.

Method Types

Creating new instances	+alloc +allocFromZone:
Modifying the Contents	-insertObject:at: -replaceObjectAt:with: -replaceFrom:to:with: -replaceFrom:to:withObject: -reverse -switchObjectAt:withObjectAt:
Making Related Collections	-copyReverse -copyReverseFromZone:

Class Methods

alloc

+ **alloc;**

This method cannot be used to create an FCUnsortedCollection object. FCUnsortedCollection is an abstract superclass, you should call **alloc** only on its instantiable subclasses. The method is implemented only to prevent you from using it; if you do use it, it generates an error message.

allocFromZone:

+ **allocFromZone:**(NXZone *)*zone;*

This method cannot be used to create an FCUnsortedCollection object. FCUnsortedCollection is an abstract superclass, you should call **allocFromZone:** only on its instantiable subclasses. The method is implemented only to prevent you from using it; if you do use it, it generates an error message.

Instance Methods

copyReverse

- **copyReverse;**

Returns a new collection object with the same contents as the receiver, only with the order of the objects reversed (e.g., the last object in the receiver would be the first object in new collection, etc.). The objects in the receiving collection aren't copied; therefore, both collections contain pointers to the same set of objects.

See also: **- reverse, - copy** (FCCollection)**, - copyAs:** (FCCollection)**, - deepCopy** (FCCollection)

copyReverseFromZone:

- **copyReverseFromZone:**(NXZone *)*zone;*

Performs the same operation as **copyReverse** except that the new instance is allocated from *zone* .

insertObject:at:

- **insertObject:anObject at:**(unsigned)*index;*

Inserts *anObject* into the collection at *index* , moving objects down one slot to make room.   If *index* equals the value returned by the **count** method, *anObject* is inserted at the end of the collection.   However, the insertion fails and returns **nil** if *index* is greater than the value returned by **count** or *anObject* is **nil** .

If the programmer has set a content class, **insertObject:at:** will fail if *anObject* isn't a kind of that class. If a subclass of FCUnsortedCollection requires unique elements and *anObject* is already in the collection, the insertion will also fail. If *anObject* is successfully inserted into the collection, this method returns self.

See also: **- count**   (FCCollection)**, - addObject:**   (FCCollection)**, - uniqueElements** (FCCollection)**, - setContentClass:**   (FCCollection)

replaceFrom:to:with:

- **replaceFrom:**(unsigned)*first* **to:**(unsigned)*last* **with:otherCollection;**

Removes the objects from positions *first* to *last* and then inserts the objects from *otherCollection* in their place.   The collection *otherCollection* does not have to be the same size as *last* - *first* .

If the programmer has set a content class, **replaceFrom:to:with:** will not insert any objects from *otherCollection* that aren't a kind of that class. If a subclass of FCUnsortedCollection requires unique elements and any objects from *otherCollection* are already in this collection, those objects are skipped as well. If *otherCollection* is successfully inserted into the collection, this method returns self.   If this method fails or the reciever is the same as *otherCollection* , it returns **nil** .

See also: **- replaceObjectAt:with:, - replaceFrom:to:withObject:, - uniqueElements** (FCCollection)**, - setContentClass:**   (FCCollection)

replaceFrom:to:withObject:

- **replaceFrom:**(unsigned)*first* **to:**(unsigned)*last* **withObject:anObject;**

Removes the objects from positions *first* to *last* and then inserts *last* - *first* + 1 copies of *anObject* in their place.

If the programmer has set a content class, **replaceFrom:to:withObject:** will not insert *anObject* if it isn't a kind of that class. If a subclass of FCUnsortedCollection requires unique elements then *anObject* will only be inserted once, no matter how many objects are deleted. If *anObject* is successfully inserted into the collection, this method returns self.   If this method fails or *anObject* is **nil** , it returns **nil** .

See also: **- replaceObjectAt:with:, - replaceFrom:to:with:, - uniqueElements** (FCCollection)**, - setContentClass:**   (FCCollection)

replaceObjectAt:with:

- **replaceObjectAt:**(unsigned)*index* **with:newObject;**

Returns the object at *index* after replacing it with *newObject* . If there's no object at *index* or *newObject* is **nil** , nothing is replaced and **nil** is returned.

If the programmer has set a content class, **replaceObjectAt:with:** will fail and return **nil** if *anObject* isn't a kind of that class. If a subclass of FCUnsortedCollection requires unique elements and *anObject* is already in the collection somewhere else, the replace will also fail.

See also: **- replaceObject:with:**   (FCCollection)**, - count**   (FCCollection)**, - addObject:** (FCCollection)**, - uniqueElements**   (FCCollection)**, - setContentClass:**   (FCCollection)

reverse

- **reverse;**

Reverses the order of the objects in the collection; the old last object will now be the first object, etc.

See also: **- copyReverse**

switchObjectAt:withObjectAt:

- **switchObjectAt:**(unsigned)*index1* **withObjectAt:**(unsigned)*index2;*

Switches the positions of the objects at *index1* and *index2* in the collection. Returns **nil** if *index1* or *index2* is out of bounds.