NSCountedSet

Inherits From: NSMutableSet : NSSet : NSObject

Conforms To: NSCoding
NSCopying
NSMutableCopying (NSSet)
NSObject (NSObject)

Declared In: Foundation/NSSet.h

Class at a Glance

Purpose

An NSCountedSet object stores a modifiable set of objects, where a given object can be included in the set multiple times.

Principal Attributes

- The objects that make up the set.
- For each object in the set, a count of the number of times the object is included in the set.

Commonly Used Methods

– addObject:	Adds an object to the set.
– removeObject:	Removes an object from the set.

Class Description

The NSCountedSet class declares the programmatic interface to an object that manages a mutable set of objects. NSCountedSet provides support for the mathematical concept of a *counted set*. A counted set, both in its mathematical sense and in the implementation of NSCountedSet, is an unordered collection of elements, just as in a regular set, but the elements of the set aren't necessarily distinct. A counted set is also known as a *bag*.

Each distinct object inserted into an NSCountedSet object has a counter associated with it. NSCountedSet keeps track of the number of times objects are inserted and requires that objects be removed the same number of times. Thus, there is only one instance of an object in an NSSet even if the object has been added

to the set multiple times. The NSSet and NSMutableSet classes are provided for static and dynamic sets (respectively) whose elements *are* distinct.

You add or remove objects from a counted set using the **addObject:** and **removeObject:** methods. An NSCountedSet may be queried using the **objectEnumerator** method, which provides for traversing elements of the set one by one. The **countForObject:** method returns the number of times the specified object has been added to this set**.**

Method Types

Initializing an NSCountedSet	– initWithArray: – initWithSet:
Adding and removing entries	– addObject: – removeObject:
Accessing the members	– allObjects – count – countForObject: – member: – objectEnumerator

Instance Methods

addObject:

– (void)**addObject:**(id)*anObject*

Adds *anObject* to the receiver if it isn't already a member. If *anObject* is already a member, **addObject:** increments the count associated with the object. In either case, *anObject* is then sent a *retain* message.

allObjects

– (NSArray *)**allObjects**

Returns an array containing the set's members, or an empty array if the set has no members. Each object is only represented in the array once—that is, if you add an object to the set more than once, it will appear only once in the array that is returned by **allObjects**. The order of the objects in the array isn't defined. This method invokes **objectEnumerator** as part of its implementation.

count

– (unsigned int)**count**

Returns the number of unique members in the set. Objects that are added to the set multiple times are only reflected in the count once.

countForObject:

– (unsigned int)**countForObject:**(id)*anObject*

Returns the count associated with *anObject* in the receiver, which can be thought of as the number of occurrences of *anObject* that are present in the set.

initWithArray:

– (id)**initWithArray:**(NSArray *)*anArray*

Initializes a newly allocated counted set object with the contents of *anArray.* Returns **self**.

See also: – **initWithArray:**(NSSet), – **initWithSet:**(NSSet), +**setWithCapacity:**(NSMutableSet)

initWithSet:

– (id)**initWithSet:**(NSSet *)*aSet*

Initializes a newly allocated counted set object with the contents of *aSet.* Returns **self**.

See also: – **initWithArray:**(NSSet), – **initWithSet:**(NSSet), +**setWithCapacity:**(NSMutableSet)

member:

– (id)**member:**(id)*anObject*

Returns *anObject* if *anObject* is present in the set (as determined by **isEqual:**), otherwise returns **nil**.

See also: **– containsObject:**(NSSet)

objectEnumerator

– (NSEnumerator *)**objectEnumerator**

Returns an enumerator object that lets you access each object in the set, independent of its count. This means that if you add a given object to the counted set multiple times, an enumeration of the set will produce that object only once.

When this method is used with a counted set, your code shouldn't modify the set during enumeration. If you intend to modify the set, use the **allObjects** method to create a "snapshot," then enumerate the snapshot and modify the original set.

See also: – **nextObject** (NSEnumerator)

removeObject:

– (void)**removeObject:**(id)*anObject*

If *anObject* is present in the set, decrements the count associated with it. If the count is decremented to zero, *anObject* is removed from the set and is sent a **release** message. **removeObject:** does nothing if *anObject* is not present in the receiver.

See also: – **countForObject:**