NSCopying

Adopted By: <<*various*>>

Declared In: foundation/NSObject.h

Protocol Description

A class whose instances provide functional copies of themselves must adopt the NSCopying protocol. The exact meaning of "copy" can vary from class to class, but a copy must be a functionally independent object, identical to the original at the time the copy was made. Where the concept "immutable vs. mutable" applies to an object, this protocol produces immutable copies; see the NSMutableCopying protocol for details on making mutable copies.

In most cases, to produce a copy that's independent of the original, a *deep copy* must be made. A deep copy is one in which every instance variable of the receiver is duplicated, not simply referenced in the copy. If the receiver's instance variables themselves have instance variables, those too must be duplicated, and so on. A deep copy is thus a completely separate object from the original; changes to it don't affect the original, and changes to the original don't affect it. Further, for an immutable copy no part at any level may be changed, making a copy a "snapshot" of the original object.

Making a complete deep copy isn't always needed. Some objects can reasonably share instance variables among themselves—for example a static string object that gets replaced but not modified. In such cases your class can implement NSCopying more cheaply than it might otherwise need to.

The copied instance returned by the methods in this protocol are *not* autoreleased.

Instance Methods

copy

– **copy**

Invokes **copyWithZone:** with the same memory zone as the receiver. Subclasses should implement their own versions of **copyWithZone:**, not **copy**, to define class-specific copying.

See also: – **mutableCopy** (NSMutableCopying protocol)

copyWithZone:

– (id)**copyWithZone:**(NSZone *)*zone*

Returns a new instance that's a copy of the receiver. Memory for the new instance is allocated from *zone*. The copy returned is immutable if the consideration "immutable vs. mutable" applies to the receiving object; otherwise the exact nature of the copy is determined by the class. The returned copy is not autoreleased.

Making an immutable copy usually involves making immutable copies of all of the receiver's instance variables. Thus, many new objects may be made in the process of copying one object.

See also: – **mutableCopyWithZone:** (NSMutableCopying protocol)