	+ superclass
	superclass
Identifying and comparing instances	
	isEqual:
	hash
	self
	name
	printForDebugger:
Testing inheritance relationships	
	isKindOf:
	isKindOfClassNamed:
	isMemberOf:
	isMemberOfClassNamed:
Testing class functionality	respondsTo:
	+ instancesRespondTo:
Testing for protocol conformance	
	+ conformsTo:
	conformsTo:
Sending messages determined at run time	
	perform:
	perform:with:
	perform:with:with:
Forwarding messages	forward::
	performv::
Obtaining method information	methodFor:
	+ instanceMethodFor:
	descriptionForMethod:
	+ descriptionForInstanceMethod:
Posing	+ poseAs:
Enforcing intentions	notImplemented:
	subclassResponsibility:
Error handling	doesNotRecognize:
	error:
Dynamic loading	+ finishLoading:
	+ startUnloading
Archiving	read:
	write:
	startArchiving:
	awake
	finishUnarchiving
	+ setVersion:
	+ version

zone, init

name, class

conformsTo:

descriptionForMethod:

free

init, class

respondsTo:,   forward::

name, + class

init, + alloc, + allocFromZone:

superclass

awake

Implemented by subclasses to reinitialize the receiving object after it has been unarchived (by read
is automatically sent to every object after it has been unarchived and after all the objects it refers to

The default version of the method defined here merely returns self.

A class can implement an awake method to provide for more initialization than can be done in the
implementation of awake should limit the work it does to the scope of the class definition, and inc
initialization of classes farther up the inheritance hierarchy through a message to super. For examp

read:, finishUnarchiving, awakeFromNib (NXNibNotification protocol in the Application Kit),
(Application class in the Application Kit)

class

Returns the class object for the receiver's class.

copy

Returns a new instance that's an exact copy of the receiver. This method creates only one new obj
instance variables that point to other objects, the instance variables in the copy will point to the sar
of the instance variables are copied, but the objects they point to are not.

This method does its work by invoking the copyFromZone: method and specifying that the copy sl
the same memory zone as the receiver. If a subclass implements its own copyFromZone: method,
use it to copy instances of the subclass. Therefore, a class can support copying from both methods
class-specific version of copyFromZone:.

copyFromZone:

copyFromZone:(NXZone *)zone

Returns a new instance that's an exact copy of the receiver. Memory for the new instance is alloca

This method creates only one new object. If the receiver has instance variables that point to other
variables in the copy will point to the same objects. The values of the instance variables are copiec
point to are not.

Subclasses should implement their own versions of copyFromZone:, not copy, to define class-spec

copy, zone

(struct objc_method_description *)descriptionForMethod:(SEL)aSelector

Returns a pointer to a structure that describes the aSelector method, or NULL if the aSelector meth
When the receiver is an instance, aSelector should be an instance method when the receiver is a cla
method.

The objc_method_description structure is declared in objc/Protocol.h, and is mostly used in the im
protocols. It includes two fieldsÐthe selector for the method (which will be the same as aSelector
encoding the method's return and argument types:

descriptionForClassMethod: (Protocol class in the Run-Time System), descriptionForInstanceMe
the Run-Time System)

doesNotRecognize:(SEL)aSelector

Handles aSelector messages that the receiver doesn't recognize. The run-time system invokes this
object receives an aSelector message that it can't respond to or forward. This method, in turn, invo
to generate an error message and abort the current process.

doesNotRecognize: messages should be sent only by the run-time system. Although they're somet
code to prevent a method from being inherited, it's better to use the error: method directly. For exa
subclass might renounce the copy method by reimplementing it to include an error: message as fol

error:, subclassResponsibility:, + name

error:(const char *)aString, ...

Generates a formatted error message, in the manner of printf(), from aString followed by a variable
For example:

subclassResponsibility:, notImplemented:, doesNotRecognize:

There's no default implementation of the finishUnarchiving method. The Object class declares thi
define it.

read:, awake, startArchiving:

forward:(SEL)aSelector :(marg_list)argFrame

Implemented by subclasses to forward messages to other objects. When an object is sent an aSelec
run-time system can't find an implementation of the method for the receiving object, it sends the o
message to give it an opportunity to delegate the message to another receiver. (If the delegated rec
the message either, it too will be given a chance to forward it.)

The forward:: message thus allows an object to establish relationships with other objects that will,
act on its behalf. The forwarding object is, in a sense, able to ªinheritº some of the characteristics
forwards the message to.

A forward:: message is generated only if the aSelector method isn't implemented by the receiving
of the classes it inherits from.

An implementation of the forward:: method has two tasks:

·To locate an object that can respond to the aSelector message. This need not be the same object f

·To send the message to that object, using the performv:: method.

In the simple case, in which an object forwards messages to just one destination (such as the hypot
variable in the example below), a forward:: method could be as simple as this:

performv::, doesNotRecognize:

free

Frees the memory occupied by the receiver and returns nil. Subsequent messages to the object wil
indicating that a message was sent to a freed object (provided that the freed memory hasn't been re

Subclasses must implement their own versions of free to deallocate any additional memory consun
as dynamically allocated storage for data, or other objects that are tightly coupled to the freed obje
without it. After performing the class-specific deallocation, the subclass method should incorporat
of free through a message to super:

the same id.

isEqual:

init

Implemented by subclasses to initialize a new object (the receiver) immediately after memory for i
An init message is generally coupled with an alloc or allocFromZone: message in the same line of

(BOOL)isEqual:anObject

Returns YES if the receiver is the same as anObject, and NO if it isn't. This is determined by com
receiver to the id of anObject.

Subclasses may need to override this method to provide a different test of equivalence. For examp
two objects might be said to be the same if they're both the same kind of object and they both cont

(BOOL)isKindOf:aClassObject

Returns YES if the receiver is an instance of aClassObject or an instance of any class that inherits
Otherwise, it returns NO. For example, in this code isKindOf: would return YES because, in the A
Menu class inherits from Window:

isMemberOf:

(BOOL)isKindOfClassNamed:(const char *)aClassName

Returns YES if the receiver is an instance of aClassName or an instance of any class that inherits f
method is the same as isKindOf:, except it takes the class name, rather than the class id, as its argu

isMemberOfClassNamed:

(BOOL)isMemberOf:aClassObject

Returns YES if the receiver is an instance of aClassObject. Otherwise, it returns NO. For exampl
isMemberOf: would return NO:

isKindOf:

(IMP)methodFor:(SEL)aSelector

Locates and returns the address of the receiver's implementation of the aSelector method, so that it
function. If the receiver is an instance, aSelector should refer to an instance method if the receiver
refer to a class method.

aSelector must be a valid, nonNULL selector. If in doubt, use the respondsTo: method to check be
selector to methodFor:.

IMP is defined (in the objc/objc.h header file) as a pointer to a function that returns an id and takes
arguments (in addition to the two ªhiddenº argumentsÐ self and _cmdÐthat are passed to every m
:

(const char *)name

Implemented by subclasses to return a name associated with the receiver.

By default, the string returned contains the name of the receiver's class. However, this method is c
to return a more object-specific name. You should therefore not rely on it to return the name of th
name of the class, use the class name method instead:

notImplemented:(SEL)aSelector

Used in the body of a method definition to indicate that the programmer intended to implement the
stub for the time being. aSelector is the selector for the unimplemented method notImplemented: r
self. For example:

Sends an aSelector message to the receiver and returns the result of the message. This is equivalen
aSelector message directly to the receiver. For example, all three of the following messages do the

perform:with:, perform:with:with:, methodFor:

perform:(SEL)aSelector with:anObject

Sends an aSelector message to the receiver with anObject as an argument. This method is the sam
that you can supply an argument for the aSelector message. aSelector should identify a method tha
argument of type id.

perform:, perform:with:afterDelay:cancelPrevious: (Application Kit Object Additions)

perform:(SEL)aSelector
with:anObject
with:anotherObject

Sends the receiver an aSelector message with anObject and anotherObject as arguments. This met
perform:, except that you can supply two arguments for the aSelector message. aSelector should id
can take two arguments of type id.

perform:

performv:(SEL)aSelector :(marg_list)argFrame

Sends the receiver an aSelector message with the arguments in argFrame. performv:: messages are
implementations of the forward:: method. Both arguments, aSelector and argFrame, are identical t
run-time system passes to forward::. They can be taken directly from that method and passed throu
performv::.

performv:: should be restricted to implementations of the forward:: method. Because it doesn't res
arguments in the aSelector message or their type, it may seem like a more flexible way of sending
, perform:with:, or perform:with:with:. However, it's not an appropriate substitute for those metho

provides the class name and the hexadecimal address of the receiver, formatted as follows:

<classname: 0xaddress>

Debuggers can use this method to ask objects to identify themselves.

read:(NXTypedStream *)stream

Implemented by subclasses to read the receiver's instance variables from the typed stream stream.
implement a read: method for any class you create, if you want its instances (or instance of classes
be archivable.

The method you implement should unarchive the instance variables defined in the class in a manne
way they were archived by write:. In each class, the read: method should begin with a message to

awake, finishUnarchiving, write:

(BOOL)respondsTo:(SEL)aSelector

Returns YES if the receiver implements or inherits a method that can respond to aSelector message
The application is responsible for determining whether a NO response should be considered an err

Note that if the receiver is able to forward aSelector messages to another object, it will be able to r
albeit indirectly, even though this method returns NO.

forward::, + instancesRespondTo:

self

Returns the receiver.

startArchiving:(NXTypedStream *)stream

Implemented by subclasses to prepare an object for being archivedÐthat is, for being written to th
A startArchiving: message is sent to an object just before it's archivedÐbut only if it implements a
respond. The message gives the object an opportunity to do anything necessary to get itself, or the
write: message begins the archiving process.

implement the method, it will inherit it from the abstract superclass. That version of the method ge
it's invoked. To avoid the error, subclasses must override the superclass method.

For example, if subclasses are expected to implement doSomething methods, the superclass would
way:

doesNotRecognize:, notImplemented:, error:

superclass

Returns the class object for the receiver's superclass.

write:(NXTypedStream *)stream

Implemented by subclasses to write the receiver's instance variables to the typed stream stream. Y
write: method for any class you create, if you want to be able to archive its instances (or instances
from it).

The method you implement should archive only the instance variables defined in the class, but sho
message to super so that all inherited instance variables will also be archived:

read:, startArchiving:

(NXZone *)zone

Returns a pointer to the zone from which the receiver was allocated. Objects created without speci
allocated from the default zone, which is returned by NXDefaultMallocZone().