

MaskedException

Inherits From: NSEException
Declared In: MathArray/MaskedException.h

Class Description

This is a simple class that allows one to control the effects of raised exceptions. Perhaps one of the best uses of this class is for debugging. If an exception occurs during debugging, the program aborts (unless the exception is caught), but in any case, the original context of the error is completely lost, making it difficult to trace the cause of the error. With MaskedException, you can cause the exception to abort the program in place, without jumping out of the context, so you can go back and look for at the frame that caused the error. You can set a mask for one exception or a whole class of excetions using MaskedExcetion. The set of choices (masks) includes:

- RaiseException* Normal operation. Raise the exception as usual.
- AbortException* Abort, rather than raise the exception. Prints a message to stderr.
- HandleException* Sends a message to a specific exception handler object.
- LogException* Log the exception (NSLog) and continue excecution. Dangerous!

IgnoreException Completely ignore the exception and continue execution. Dangerous!

Note the last two choices, which allow one to completely ignore the exception and continue as if nothing happened. Normally, this is not very useful and would probably cause some kind of program crash, but it could be used in cases where an exception is meant to be a warning rather than an error. The third choice allows one to set a specific handler object to handle one or more exceptions. When the exception is raised, the message **exceptionWithName:reason:userInfo** is sent to the object (like the `NSException` class method of the same name).

Instance Variables

Method Types

- + `bodySnatchNSException`
- + `maskForException:`
- + `setHandler:forException:`
- + `setHandler:forExceptionsWithPrefix:`
- + `setHandlerForAllExceptions:`
- + `setMask:forException:`
- + `setMask:forExceptionsWithPrefix:`
- + `setMaskForAllExceptions:`

Class Methods

bodysnatchNSException

+ (void) **bodysnatchNSException**

Easy way to get MaskedException to work with all exceptions (same as [self poseAsClass:[NSException class]]).

See also:

maskForException:

+ (ExceptionMask) **maskForException:**(NSString *)*exceptionName*

Returns the mask for the exception with name *exceptionName* if it was explicitly set, or the mask for the set of exception names that have the same prefix as *exceptionName*. Otherwise returns the global exception mask (*RaiseException* by default) set by **setMaskForAllExceptions**.

See also: + setMask:forException:, + setMask:forExceptionsWithPrefix:, + setMaskForAllExceptions:

setHandler:forException:

+(void) **setHandler:**(id <MaskedExceptionHandling>)*handler*
forException:(NSString *)*exceptionName*

Set the handler object for exceptions with the name *exceptionName*.

See also:

setHandler:forExceptionsWithPrefix:

+ (void) **setHandler:**(id <MaskedExceptionHandling>)*handler*
forExceptionsWithPrefix:(NSString *)*prefix*

Set the handler object for any exception with a name that has the prefix *prefix*. This handler is used for all exceptions with matching prefix unless a handler is set explicitly for an exception using **setHandler:forException:**.

See also:

setHandlerForAllExceptions:

+ (void) **setHandlerForAllExceptions:**(id <MaskedExceptionHandling>)*handler*

Set a handler for all exceptions. This handler is used for all exceptions unless a handler is set explicitly for an exception using **setHandler:forException:** or **setHandler:forExceptionsWithPrefix:**.

See also:

setMask:forException:

+ (void) **setMask:**(ExceptionMask)*mask*
 forException:(NSString *)*exceptionName*

Set the mask for exceptions with the name *exceptionName*.

See also:

setMask:forExceptionsWithPrefix:

+ (void) **setMask:**(ExceptionMask)*mask*
 forExceptionsWithPrefix:(NSString *)*prefix*

Set the mask for any exception with a name that has the prefix *prefix*. This mask is used for all exceptions with a matching prefix, unless a mask is set explicitly for an exception using **setMask:forException:**.

See also:

setMaskForAllExceptions:

+ (void) **setMaskForAllExceptions:**(ExceptionMask)*mask*

Set a mask for all exceptions. This mask is used for all exceptions unless a mask is set explicitly for an exception using **setMask:forException:** or **setMask:forExceptionsWithPrefix:**.

See also:

Instance Methods