NSLock

Inherits From:	NSObject
Conforms To:	NSLocking NSObject (NSObject)
Declared In:	Foundation/NSLock.h

Class Description

An NSLock object is used to coordinate the operation of multiple threads of execution within the same application. An NSLock object can be used to mediate access to an application's global data or to protect a critical section of code, allowing it to run atomically.

An NSLock object represents a lock that can be acquired by only a single thread at a time. While one thread holds the lock, any other thread is prevented from doing so until the owner relinquishes the lock. An application can have multiple NSLock objects, each protecting different sections of code. However, each NSLock must be created before the application becomes multithreaded.

The basic interface to NSLock is declared by the NSLocking protocol, which declares the **lock** and **unlock** methods. To this base, NSLock adds the **tryLock** and **lockBeforeDate:** methods. Whereas the **lock** method declared in the protocol doesn't return until it is successful, the methods declared in this class add more flexible means of acquiring a lock.

An NSLock could be used to coordinate the updating of a visual display shared by a number of threads involved in a single calculation:

```
BOOL moreToDo = YES;
NSLock *theLock = [[NSLock alloc] init];
...
while (moreToDo) {
    /* Do another increment of calculation */
    /* until there's no more to do. */
    if ([theLock tryLock]) {
        /* Update display used by all threads. */
        [theLock unlock];
    }
}
```

The NSLock, NSConditionLock, and NSRecursiveLock classes all adopt the NSLocking protocol and offer various additional features and performance characteristics. See the NSConditionLock and NSRecursiveLock class descriptions for more information.

Adopted Protocols

NSLocking	– lock – unlock

Method Types

Acquiring a lock	– lockBeforeDate: – tryLock

Instance Methods

lockBeforeDate

– (BOOL)**lockBeforeDate:**(NSDate *)*limit*

Attempts to acquire a lock before the date represented by *limit*. The thread is blocked until the receiver acquires the lock or *limit* is reached. Returns YES if the lock is acquired within this time limit. Returns NO if the time limit expires before a lock can be acquired.

tryLock

– (BOOL)**tryLock**

Attempts to acquire a lock. Returns immediately, with a value of YES if successful and NO otherwise.