System.Threading.Monitor Class

```
[ILAsm]
.class public sealed Monitor extends System.Object

[C#]
public sealed class Monitor
```

Assembly Info:

- *Name:* mscorlib
- *Public Key:* [00 00 00 00 00 00 00 00 04 00 00 00 00 00 00 00]
- *Version:* 2.0.x.x
- *Attributes:*
  - CLSCompliantAttribute(true)

Summary

Provides a mechanism that synchronizes access to objects.

Inherits From: System.Object

Library: BCL

Thread Safety: All public static members of this type are safe for multithreaded operations. No instance members are guaranteed to be thread safe.

Description

The `System.Threading.Monitor` class controls access to objects by granting a single thread a lock for an object. Object locks provide the ability to restrict access to a block of code, commonly called a critical section. While a thread owns the lock for an object no other thread can acquire the lock for the object. Additionally, the `System.Threading.Monitor` class can be used to ensure that no other thread can access a section of application code being executed by the lock owner, unless the other thread is executing the code using a different locked object.

The following information is maintained for each synchronized object:

- A reference to the thread that currently holds the lock.
- A reference to a "ready queue", which contains the threads that are ready to obtain the lock.
- A reference to a "waiting queue", which contains the threads that are waiting for notification of a change in the state of the locked object.

The following table describes the actions taken by threads that access synchronized objects:

Action	Description
Enter	Acquires a lock for an object. Also marks the beginning of a critical section. No other thread can enter the critical section unless they are executing the instructions in the critical section using a different locked object. [*Note:* See the `System.Threading.Monitor.Enter` and `System.Threading.Monitor.TryEnter` methods.]
Wait	Releases the lock on an object in order to permit other threads to lock and access the object. The calling thread waits while another thread accesses the object. Pulse signals (see below) are used to notify waiting threads about changes to an object's state. [*Note:* See `System.Threading.Monitor.Wait`.]
Pulse (signal)	Sends a signal to one or more waiting threads. The signal notifies a waiting thread that the state of the locked object has changed, and the owner of the lock is ready to release the lock. The waiting thread is placed in the object's ready queue so that it can eventually receive the lock for the object. Once the thread has the lock, it can check the new state of the object to see if the required state has been reached. [*Note:* See `System.Threading.Monitor.Pulse` and `System.Threading.Monitor.PulseAll`.]
Exit	Releases the lock on an object. Also marks the end of a critical section protected by the locked object. [*Note:* See `System.Threading.Monitor.Exit`.]

The `System.Threading.Monitor.Enter` and `System.Threading.Monitor.Exit` methods are used to mark the beginning and end of a critical section. If the critical section is a set of contiguous instructions, then the lock acquired by the `System.Threading.Monitor.Enter` method guarantees that only a single thread can execute the enclosed code with the locked object. This facility is typically used to synchronize access to a static or instance method of a class. If an instance method requires synchronized thread access, the instance method invokes the `System.Threading.Monitor.Enter` and corresponding `System.Threading.Monitor.Exit` methods using itself (the current instance) as the object to lock. Since only one thread can hold the lock on the current instance, the method can only be executed by one thread at a time. Static methods are protected in a similar fashion using the `System.Type` object of the current instance as the locked object.

[*Note:* The functionality provided by the `System.Threading.Monitor.Enter` and `System.Threading.Monitor.Exit` methods is identical to that provided by the C# lock statement.

If a critical section spans an entire method, the locking facility described above can be achieved by placing the `System.Runtime.CompilerServices.MethodImplAttribute` on the method, and specifying the `System.Runtime.CompilerServices.MethodImplOptions.Synchronized` option. Using this attribute, the `System.Threading.Monitor.Enter` and `System.Threading.Monitor.Exit` statements are not needed. Note that the attribute causes the current thread to hold the lock until the method returns; if the lock can be released sooner, use the `System.Threading.Monitor` class (or `C# lock` statement) instead of the attribute.

While it is possible for the `System.Threading.Monitor.Enter` and `System.Threading.Monitor.Exit` statements that lock and release a given object to cross member and/or class boundaries, this practice is strongly discouraged.

]

Monitor.Enter(System.Object) Method

```
[ILAsm]
.method public hidebysig static void Enter(object obj)

[C#]
public static void Enter(object obj)
```

Summary

Acquires an exclusive lock on the specified object.

Parameters

Parameter	Description
obj	The `System.Object` on which to acquire the lock.

Description

This method acquires an exclusive lock on *obj*.

A caller of this method is required to invoke `System.Threading.Monitor.Exit` once for each `System.Threading.Monitor.Enter` invoked.

The caller of this method is blocked if another thread has obtained the lock by calling `System.Threading.Monitor.Enter` and specifying the same object. The caller is not blocked if the current thread holds the lock. The same thread can invoke `System.Threading.Monitor.Enter` more than once (and it will not block); however, an equal number of `System.Threading.Monitor.Exit` calls are required to be invoked before other threads waiting on the object will unblock.

[*Note:* Invoking this member is identical to using the C# `lock` statement.]

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.

1
2
3

Monitor.Exit(System.Object) Method

```
[ILAsm]
.method public hidebysig static void Exit(object obj)

[C#]
public static void Exit(object obj)
```

Summary

Releases an exclusive lock on the specified `System.Object`.

Parameters

Parameter	Description
obj	The `System.Object` on which to release the lock.

Description

This method releases an exclusive lock on *obj*. The caller is required to own the lock on *obj*.

If the caller owns the lock on the specified object, and has made an equal number of `System.Threading.Monitor.Exit` and `System.Threading.Monitor.Enter` calls for the object, then the lock is released. If the caller has not invoked `System.Threading.Monitor.Exit` as many times as `System.Threading.Monitor.Enter`, the lock is not released.

[*Note:* If the lock is released and there are other threads in the ready queue for the object, one of the threads will acquire the lock. If there are other threads in the waiting queue waiting to acquire the lock, they are not automatically moved to the ready queue when the owner of the lock calls `System.Threading.Monitor.Exit`. To move one or more waiting threads into the ready queue, call `System.Threading.Monitor.Pulse` or `System.Threading.Monitor.PulseAll` prior to invoking `System.Threading.Monitor.Exit`.]

Exceptions

Exception	Condition

System.ArgumentNullException	*obj* is `null`.
System.Threading. SynchronizationLockException	The current thread does not own the lock for the specified object.

Monitor.Pulse(System.Object) Method

```
[ILAsm]
.method public hidebysig static void Pulse(object obj)

[C#]
public static void Pulse(object obj)
```

Summary

Notifies the next waiting thread (if any) of a change in the specified locked object's state.

Parameters

Parameter	Description
obj	The `System.Object` a thread might be waiting for.

Description

The thread that currently owns the lock on the specified object invokes this method to signal the next thread in line for the lock (in the queue of threads waiting to acquire the lock on the object). Upon receiving the pulse, the waiting thread is moved to the ready queue. When the thread that invoked `Pulse` releases the lock, the next thread in the ready queue (which is not necessarily the thread that was pulsed) acquires the lock.

[*Note:* To signal a waiting object using `Pulse`, you must be the current owner of the lock.

To signal multiple threads, use the `System.Threading.Monitor.PulseAll` method.

]

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.
System.Threading.	The calling thread does not own the lock for

SynchronizationLockException	the specified object.

1
2
3

Monitor.PulseAll(System.Object) Method

```
[ILAsm]
.method public hidebysig static void PulseAll(object obj)

[C#]
public static void PulseAll(object obj)
```

Summary

Notifies all waiting threads (if any) of a change in the specified locked object's state.

Parameters

Parameter	Description
obj	The `System.Object` that one or more threads might be waiting for.

Description

The thread that currently owns the lock on the specified object invokes this method to signal all threads waiting to acquire the lock on the object. After the signal is sent, the waiting threads are moved to the ready queue. When the thread that invoked `PulseAll` releases the lock, the next thread in the ready queue acquires the lock.

[*Note:* To signal waiting objects using `PulseAll`, you must be the current owner of the lock.

To signal a single thread, use the `System.Threading.Monitor.Pulse` method.

]

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.
System.Threading.SynchronizationLockException	The calling thread does not own the lock for the specified object.

1
2
3

Monitor.TryEnter(System.Object) Method

```
[ILAsm]
.method public hidebysig static bool TryEnter(object obj)

[C#]
public static bool TryEnter(object obj)
```

Summary

Attempts to acquire an exclusive lock on the specified object.

Parameters

Parameter	Description
obj	The `System.Object` on which to acquire the lock.

Return Value

`true` if the current thread acquired the lock; otherwise, `false`.

Description

If successful, this method acquires an exclusive lock on *obj*. This method returns immediately, whether or not the lock is available.

This method is equivalent to `System.Threading.Monitor.TryEnter` (*obj*, `0`).

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.

Monitor.TryEnter(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static bool TryEnter(object obj, int32
millisecondsTimeout)

[C#]
public static bool TryEnter(object obj, int millisecondsTimeout)
```

Summary

Attempts, for the specified number of milliseconds, to acquire an exclusive lock on the specified object.

Parameters

Parameter	Description
obj	The `System.Object` on which to acquire the lock.
millisecondsTimeout	A `System.Int32` containing the maximum number of milliseconds to wait for the lock.

Return Value

`true` if the current thread acquired the lock; otherwise, `false`.

Description

If successful, this method acquires an exclusive lock on *obj*.

If *millisecondsTimeout* equals `System.Threading.Timeout.Infinite`, this method is equivalent to `System.Threading.Monitor.Enter` (*obj*). If *millisecondsTimeout* equals zero, this method is equivalent to `System.Threading.Monitor.TryEnter` (*obj*).

Exceptions

Exception	Condition

System.ArgumentNullException	*obj* is `null`.
System.ArgumentOutOfRangeException	*millisecondsTimeout* is negative, and not equal to `System.Threading.Timeout.Infinite`.

1
2
3

Monitor.TryEnter(System.Object, System.TimeSpan) Method

```
[ILAsm]
.method public hidebysig static bool TryEnter(object obj, valuetype
System.TimeSpan timeout)

[C#]
public static bool TryEnter(object obj, TimeSpan timeout)
```

Summary

Attempts, for the specified amount of time, to acquire an exclusive lock on the specified object.

Parameters

Parameter	Description
obj	The `System.Object` on which to acquire the lock.
timeout	A `System.TimeSpan` set to the maximum amount of time to wait for the lock.

Return Value

`true` if the current thread acquires the lock; otherwise, `false`.

Description

If successful, this method acquires an exclusive lock on *obj*.

If the value of *timeout* converted to milliseconds equals `System.Threading.Timeout.Infinite`, this method is equivalent to `System.Threading.Monitor.Enter` (*obj*). If the value of *timeout* equals zero, this method is equivalent to `System.Threading.Monitor.TryEnter` (*obj*).

Exceptions

Exception	Condition

System.ArgumentNullException	*obj* is `null`.
System.ArgumentOutOfRangeException	The value of *timeout* in milliseconds is negative and is not equal to `System.Threading.Timeout.Infinite`, or is greater than `System.Int32.MaxValue`.

1
2
3

Monitor.Wait(System.Object, System.Int32) Method

```
[ILAsm]
.method public hidebysig static bool Wait(object obj, int32
millisecondsTimeout)

[C#]
public static bool Wait(object obj, int millisecondsTimeout)
```

Summary

Releases the lock on an object and blocks the current thread until it reacquires the lock or until a specified amount of time elapses.

Parameters

Parameter	Description
obj	The `System.Object` on which to wait.
millisecondsTimeout	A `System.Int32` containing the maximum number of milliseconds to wait before this method returns.

Return Value

`true` if the lock was reacquired before the specified time elapsed; otherwise, `false`.

Description

If successful, this method reacquires an exclusive lock on *obj*.

This method behaves identically to `System.Threading.Monitor.Wait` (*obj*), except that it does not block indefinitely unless `System.Threading.Timeout.Infinite` is specified for *millisecondsTimeout*. Once the specified time has elapsed, this method returns a value that indicates whether the lock has been reacquired by the caller. If *millisecondsTimeout* equals 0, this method returns immediately.

[*Note:* This method is called when the caller is waiting for a change in the state of the object, which occurs as a result of another thread's operations on the object. For additional details, see `System.Threading.Monitor.Wait` (*obj*).]

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.
System.Threading.SynchronizationLockException	The calling thread does not own the lock for the specified object.
System.ArgumentOutOfRangeException	The value of *millisecondsTimeout* is negative, and not equal to `System.Threading.Timeout.Infinite`.

Monitor.Wait(System.Object, System.TimeSpan) Method

```
[ILAsm]
.method public hidebysig static bool Wait(object obj, valuetype
System.TimeSpan timeout)

[C#]
public static bool Wait(object obj, TimeSpan timeout)
```

Summary

Releases the lock on an object and blocks the current thread until it reacquires the lock or until a specified amount of time elapses.

Parameters

Parameter	Description
obj	The `System.Object` on which to wait.
timeout	A `System.TimeSpan` set to the maximum amount of time to wait before this method returns.

Return Value

`true` if the lock was reacquired before the specified time elapsed; otherwise, `false`.

Description

If successful, this method reacquires an exclusive lock on *obj*.

This method behaves identically to `System.Threading.Monitor.Wait` (*obj*), except that it does not block indefinitely unless `System.Threading.Timeout.Infinite` milliseconds is specified for *timeout*. Once the specified time has elapsed, this method returns a value that indicates whether the lock has been reacquired by the caller. If *timeout* equals 0, this method returns immediately.

[*Note:* This method is called when the caller is waiting for a change in the state of the object, which occurs as a result of another thread's operations on the object. For additional details, see `System.Threading.Monitor.Wait` (*obj*).]

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.
System.Threading.SynchronizationLockException	The calling thread does not own the lock for the specified object.
System.ArgumentOutOfRangeException	If *timeout* is negative, and is not equal to `System.Threading.Timeout.Infinite`, or is greater than `System.Int32.MaxValue`.

Monitor.Wait(System.Object) Method

```
[ILAsm]
.method public hidebysig static bool Wait(object obj)

[C#]
public static bool Wait(object obj)
```

Summary

Releases the lock on an object and blocks the current thread until it reacquires the lock.

Parameters

Parameter	Description
obj	The `System.Object` on which to wait.

Return Value

`true` if the call returned because the caller reacquired the lock for the specified object. This method does not return if the lock is not reacquired.

Description

This method reacquires an exclusive lock on *obj*.

The thread that currently owns the lock on the specified object invokes this method in order to release the object so that another thread can access it. The caller is blocked while waiting to reacquire the lock. This method is called when the caller is waiting for a change in the state of the object, which occurs as a result of another thread's operations on the object.

When a thread calls `Wait`, it releases the lock on the object and enters the object's waiting queue. The next thread in the object's ready queue (if there is one) acquires the lock and has exclusive use of the object. All threads that call `Wait` remain in the waiting queue until they receive a signal via `System.Threading.Monitor.Pulse` or `System.Threading.Monitor.PulseAll` sent by the owner of the lock. If `Pulse` is sent, only the thread at the head of the waiting queue is affected. If `PulseAll` is sent, all threads that are waiting for the object are affected. When the signal is received, one or more threads leave the waiting queue and enter the ready queue. A thread in the ready queue is permitted to reacquire the lock.

This method returns when the calling thread reacquires the lock on the object. Note that

this method blocks indefinitely if the holder of the lock does not call `System.Threading.Monitor.Pulse` or `System.Threading.Monitor.PulseAll`.

The caller executes `System.Threading.Monitor.Wait` once, regardless of the number of times `System.Threading.Monitor.Enter` has been invoked for the specified object. Conceptually, the `System.Threading.Monitor.Wait` method stores the number of times the caller invoked `System.Threading.Monitor.Enter` on the object and invokes `System.Threading.Monitor.Exit` as many times as necessary to fully release the locked object. The caller then blocks while waiting to reacquire the object. When the caller reacquires the lock, the system calls `System.Threading.Monitor.Enter` as many times as necessary to restore the saved `Enter` count for the caller.

Calling `System.Threading.Monitor.Wait` releases the lock for the specified object only; if the caller is the owner of locks on other objects, these locks are not released.

Exceptions

Exception	Condition
System.ArgumentNullException	*obj* is `null`.
System.Threading.SynchronizationLockException	The calling thread does not own the lock for the specified object.