| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TimeUnit.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/concurrent/TimeoutException.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/TimeUnit.html)    [**NO FRAMES**](http://docs.google.com/TimeUnit.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [ENUM CONSTANTS](#3znysh7) | FIELD | [METHOD](#2et92p0) | DETAIL: [ENUM CONSTANTS](#1t3h5sf) | FIELD | [METHOD](#1ksv4uv) |

## **java.util.concurrent**

Enum TimeUnit

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.lang.Enum](http://docs.google.com/java/lang/Enum.html)<[TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html)>  
 **java.util.concurrent.TimeUnit**

**All Implemented Interfaces:** [Serializable](http://docs.google.com/java/io/Serializable.html), [Comparable](http://docs.google.com/java/lang/Comparable.html)<[TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html)>

public enum **TimeUnit**extends [Enum](http://docs.google.com/java/lang/Enum.html)<[TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html)>

A TimeUnit represents time durations at a given unit of granularity and provides utility methods to convert across units, and to perform timing and delay operations in these units. A TimeUnit does not maintain time information, but only helps organize and use time representations that may be maintained separately across various contexts. A nanosecond is defined as one thousandth of a microsecond, a microsecond as one thousandth of a millisecond, a millisecond as one thousandth of a second, a minute as sixty seconds, an hour as sixty minutes, and a day as twenty four hours.

A TimeUnit is mainly used to inform time-based methods how a given timing parameter should be interpreted. For example, the following code will timeout in 50 milliseconds if the [lock](http://docs.google.com/java/util/concurrent/locks/Lock.html) is not available:

Lock lock = ...;  
 if ( lock.tryLock(50L, TimeUnit.MILLISECONDS) ) ...

while this code will timeout in 50 seconds:

Lock lock = ...;  
 if ( lock.tryLock(50L, TimeUnit.SECONDS) ) ...

Note however, that there is no guarantee that a particular timeout implementation will be able to notice the passage of time at the same granularity as the given TimeUnit.

**Since:** 1.5

| **Enum Constant Summary** | |
| --- | --- |
| [**DAYS**](http://docs.google.com/java/util/concurrent/TimeUnit.html#DAYS) |
| [**HOURS**](http://docs.google.com/java/util/concurrent/TimeUnit.html#HOURS) |
| [**MICROSECONDS**](http://docs.google.com/java/util/concurrent/TimeUnit.html#MICROSECONDS) |
| [**MILLISECONDS**](http://docs.google.com/java/util/concurrent/TimeUnit.html#MILLISECONDS) |
| [**MINUTES**](http://docs.google.com/java/util/concurrent/TimeUnit.html#MINUTES) |
| [**NANOSECONDS**](http://docs.google.com/java/util/concurrent/TimeUnit.html#NANOSECONDS) |
| [**SECONDS**](http://docs.google.com/java/util/concurrent/TimeUnit.html#SECONDS) |

| **Method Summary** | |
| --- | --- |
| long | [**convert**](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))(long sourceDuration, [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) sourceUnit)            Convert the given time duration in the given unit to this unit. |
| void | [**sleep**](http://docs.google.com/java/util/concurrent/TimeUnit.html#sleep(long))(long timeout)            Performs a Thread.sleep using this unit. |
| void | [**timedJoin**](http://docs.google.com/java/util/concurrent/TimeUnit.html#timedJoin(java.lang.Thread,%20long))([Thread](http://docs.google.com/java/lang/Thread.html) thread, long timeout)            Performs a timed Thread.join using this time unit. |
| void | [**timedWait**](http://docs.google.com/java/util/concurrent/TimeUnit.html#timedWait(java.lang.Object,%20long))([Object](http://docs.google.com/java/lang/Object.html) obj, long timeout)            Performs a timed Object.wait using this time unit. |
| long | [**toDays**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toDays(long))(long duration)            Equivalent to DAYS.convert(duration, this). |
| long | [**toHours**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toHours(long))(long duration)            Equivalent to HOURS.convert(duration, this). |
| long | [**toMicros**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toMicros(long))(long duration)            Equivalent to MICROSECONDS.convert(duration, this). |
| long | [**toMillis**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toMillis(long))(long duration)            Equivalent to MILLISECONDS.convert(duration, this). |
| long | [**toMinutes**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toMinutes(long))(long duration)            Equivalent to MINUTES.convert(duration, this). |
| long | [**toNanos**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toNanos(long))(long duration)            Equivalent to NANOSECONDS.convert(duration, this). |
| long | [**toSeconds**](http://docs.google.com/java/util/concurrent/TimeUnit.html#toSeconds(long))(long duration)            Equivalent to SECONDS.convert(duration, this). |
| static [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) | [**valueOf**](http://docs.google.com/java/util/concurrent/TimeUnit.html#valueOf(java.lang.String))([String](http://docs.google.com/java/lang/String.html) name)            Returns the enum constant of this type with the specified name. |
| static [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html)[] | [**values**](http://docs.google.com/java/util/concurrent/TimeUnit.html#values())()            Returns an array containing the constants of this enum type, in the order they are declared. |

| **Methods inherited from class java.lang.**[**Enum**](http://docs.google.com/java/lang/Enum.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Enum.html#clone()), [compareTo](http://docs.google.com/java/lang/Enum.html#compareTo(E)), [equals](http://docs.google.com/java/lang/Enum.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Enum.html#finalize()), [getDeclaringClass](http://docs.google.com/java/lang/Enum.html#getDeclaringClass()), [hashCode](http://docs.google.com/java/lang/Enum.html#hashCode()), [name](http://docs.google.com/java/lang/Enum.html#name()), [ordinal](http://docs.google.com/java/lang/Enum.html#ordinal()), [toString](http://docs.google.com/java/lang/Enum.html#toString()), [valueOf](http://docs.google.com/java/lang/Enum.html#valueOf(java.lang.Class,%20java.lang.String)) |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [wait](http://docs.google.com/java/lang/Object.html#wait()), [wait](http://docs.google.com/java/lang/Object.html#wait(long)), [wait](http://docs.google.com/java/lang/Object.html#wait(long,%20int)) |

| **Enum Constant Detail** |
| --- |

### NANOSECONDS

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **NANOSECONDS**

### MICROSECONDS

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **MICROSECONDS**

### MILLISECONDS

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **MILLISECONDS**

### SECONDS

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **SECONDS**

### MINUTES

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **MINUTES**

### HOURS

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **HOURS**

### DAYS

public static final [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **DAYS**

| **Method Detail** |
| --- |

### values

public static [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html)[] **values**()

Returns an array containing the constants of this enum type, in the order they are declared. This method may be used to iterate over the constants as follows:

for (TimeUnit c : TimeUnit.values())  
  System.out.println(c);

**Returns:**an array containing the constants of this enum type, in the order they are declared

### valueOf

public static [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) **valueOf**([String](http://docs.google.com/java/lang/String.html) name)

Returns the enum constant of this type with the specified name. The string must match *exactly* an identifier used to declare an enum constant in this type. (Extraneous whitespace characters are not permitted.)

**Parameters:**name - the name of the enum constant to be returned. **Returns:**the enum constant with the specified name **Throws:** [IllegalArgumentException](http://docs.google.com/java/lang/IllegalArgumentException.html) - if this enum type has no constant with the specified name [NullPointerException](http://docs.google.com/java/lang/NullPointerException.html) - if the argument is null

### convert

public long **convert**(long sourceDuration,  
 [TimeUnit](http://docs.google.com/java/util/concurrent/TimeUnit.html) sourceUnit)

Convert the given time duration in the given unit to this unit. Conversions from finer to coarser granularities truncate, so lose precision. For example converting 999 milliseconds to seconds results in 0. Conversions from coarser to finer granularities with arguments that would numerically overflow saturate to Long.MIN\_VALUE if negative or Long.MAX\_VALUE if positive.

For example, to convert 10 minutes to milliseconds, use: TimeUnit.MILLISECONDS.convert(10L, TimeUnit.MINUTES)

**Parameters:**sourceDuration - the time duration in the given sourceUnitsourceUnit - the unit of the sourceDuration argument **Returns:**the converted duration in this unit, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.

### toNanos

public long **toNanos**(long duration)

Equivalent to NANOSECONDS.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.**See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### toMicros

public long **toMicros**(long duration)

Equivalent to MICROSECONDS.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.**See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### toMillis

public long **toMillis**(long duration)

Equivalent to MILLISECONDS.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.**See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### toSeconds

public long **toSeconds**(long duration)

Equivalent to SECONDS.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.**See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### toMinutes

public long **toMinutes**(long duration)

Equivalent to MINUTES.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.**Since:** 1.6 **See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### toHours

public long **toHours**(long duration)

Equivalent to HOURS.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration, or Long.MIN\_VALUE if conversion would negatively overflow, or Long.MAX\_VALUE if it would positively overflow.**Since:** 1.6 **See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### toDays

public long **toDays**(long duration)

Equivalent to DAYS.convert(duration, this).

**Parameters:**duration - the duration **Returns:**the converted duration**Since:** 1.6 **See Also:**[convert(long, java.util.concurrent.TimeUnit)](http://docs.google.com/java/util/concurrent/TimeUnit.html#convert(long,%20java.util.concurrent.TimeUnit))

### timedWait

public void **timedWait**([Object](http://docs.google.com/java/lang/Object.html) obj,  
 long timeout)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Performs a timed Object.wait using this time unit. This is a convenience method that converts timeout arguments into the form required by the Object.wait method.

For example, you could implement a blocking poll method (see [BlockingQueue.poll](http://docs.google.com/java/util/concurrent/BlockingQueue.html#poll(long,%20java.util.concurrent.TimeUnit))) using:

public synchronized Object poll(long timeout, TimeUnit unit) throws InterruptedException {  
 while (empty) {  
 unit.timedWait(this, timeout);  
 ...  
 }  
 }

**Parameters:**obj - the object to wait ontimeout - the maximum time to wait. If less than or equal to zero, do not wait at all. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting.**See Also:**[Object.wait(long, int)](http://docs.google.com/java/lang/Object.html#wait(long,%20int))

### timedJoin

public void **timedJoin**([Thread](http://docs.google.com/java/lang/Thread.html) thread,  
 long timeout)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Performs a timed Thread.join using this time unit. This is a convenience method that converts time arguments into the form required by the Thread.join method.

**Parameters:**thread - the thread to wait fortimeout - the maximum time to wait. If less than or equal to zero, do not wait at all. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while waiting.**See Also:**[Thread.join(long, int)](http://docs.google.com/java/lang/Thread.html#join(long,%20int))

### sleep

public void **sleep**(long timeout)  
 throws [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html)

Performs a Thread.sleep using this unit. This is a convenience method that converts time arguments into the form required by the Thread.sleep method.

**Parameters:**timeout - the minimum time to sleep. If less than or equal to zero, do not sleep at all. **Throws:** [InterruptedException](http://docs.google.com/java/lang/InterruptedException.html) - if interrupted while sleeping.**See Also:**[Thread.sleep(long)](http://docs.google.com/java/lang/Thread.html#sleep(long))

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/TimeUnit.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/java/util/concurrent/TimeoutException.html)   NEXT CLASS | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/TimeUnit.html)    [**NO FRAMES**](http://docs.google.com/TimeUnit.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [ENUM CONSTANTS](#3znysh7) | FIELD | [METHOD](#2et92p0) | DETAIL: [ENUM CONSTANTS](#1t3h5sf) | FIELD | [METHOD](#1ksv4uv) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).