| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AtomicMarkableReference.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/atomic/AtomicLongFieldUpdater.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/atomic/AtomicReference.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/atomic/AtomicMarkableReference.html)    [**NO FRAMES**](http://docs.google.com/AtomicMarkableReference.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

## **java.util.concurrent.atomic**

Class AtomicMarkableReference<V>

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

**Type Parameters:**V - The type of object referred to by this reference

public class **AtomicMarkableReference<V>**extends [Object](http://docs.google.com/java/lang/Object.html)

An AtomicMarkableReference maintains an object reference along with a mark bit, that can be updated atomically.

Implementation note. This implementation maintains markable references by creating internal objects representing "boxed" [reference, boolean] pairs.

**Since:** 1.5

| **Constructor Summary** | |
| --- | --- |
| [**AtomicMarkableReference**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#AtomicMarkableReference(V,%20boolean))([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) initialRef, boolean initialMark)            Creates a new AtomicMarkableReference with the given initial values. |

| **Method Summary** | |
| --- | --- |
| boolean | [**attemptMark**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#attemptMark(V,%20boolean))([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) expectedReference, boolean newMark)            Atomically sets the value of the mark to the given update value if the current reference is == to the expected reference. |
| boolean | [**compareAndSet**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#compareAndSet(V,%20V,%20boolean,%20boolean))([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) expectedReference, [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) newReference, boolean expectedMark, boolean newMark)            Atomically sets the value of both the reference and mark to the given update values if the current reference is == to the expected reference and the current mark is equal to the expected mark. |
| [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) | [**get**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#get(boolean%5B%5D))(boolean[] markHolder)            Returns the current values of both the reference and the mark. |
| [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) | [**getReference**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#getReference())()            Returns the current value of the reference. |
| boolean | [**isMarked**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#isMarked())()            Returns the current value of the mark. |
| void | [**set**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#set(V,%20boolean))([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) newReference, boolean newMark)            Unconditionally sets the value of both the reference and mark. |
| boolean | [**weakCompareAndSet**](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html#weakCompareAndSet(V,%20V,%20boolean,%20boolean))([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) expectedReference, [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) newReference, boolean expectedMark, boolean newMark)            Atomically sets the value of both the reference and mark to the given update values if the current reference is == to the expected reference and the current mark is equal to the expected mark. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [clone](http://docs.google.com/java/lang/Object.html#clone()), [equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [getClass](http://docs.google.com/java/lang/Object.html#getClass()), [hashCode](http://docs.google.com/java/lang/Object.html#hashCode()), [notify](http://docs.google.com/java/lang/Object.html#notify()), [notifyAll](http://docs.google.com/java/lang/Object.html#notifyAll()), [toString](http://docs.google.com/java/lang/Object.html#toString()), [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)) |

| **Constructor Detail** |
| --- |

### AtomicMarkableReference

public **AtomicMarkableReference**([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) initialRef,  
 boolean initialMark)

Creates a new AtomicMarkableReference with the given initial values.

**Parameters:**initialRef - the initial referenceinitialMark - the initial mark

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

### getReference

public [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) **getReference**()

Returns the current value of the reference.

**Returns:**the current value of the reference

### isMarked

public boolean **isMarked**()

Returns the current value of the mark.

**Returns:**the current value of the mark

### get

public [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) **get**(boolean[] markHolder)

Returns the current values of both the reference and the mark. Typical usage is boolean[1] holder; ref = v.get(holder); .

**Parameters:**markHolder - an array of size of at least one. On return, markholder[0] will hold the value of the mark. **Returns:**the current value of the reference

### weakCompareAndSet

public boolean **weakCompareAndSet**([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) expectedReference,  
 [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) newReference,  
 boolean expectedMark,  
 boolean newMark)

Atomically sets the value of both the reference and mark to the given update values if the current reference is == to the expected reference and the current mark is equal to the expected mark.

May [fail spuriously](http://docs.google.com/package-summary.html#Spurious) and does not provide ordering guarantees, so is only rarely an appropriate alternative to compareAndSet.

**Parameters:**expectedReference - the expected value of the referencenewReference - the new value for the referenceexpectedMark - the expected value of the marknewMark - the new value for the mark **Returns:**true if successful

### compareAndSet

public boolean **compareAndSet**([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) expectedReference,  
 [V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) newReference,  
 boolean expectedMark,  
 boolean newMark)

Atomically sets the value of both the reference and mark to the given update values if the current reference is == to the expected reference and the current mark is equal to the expected mark.

**Parameters:**expectedReference - the expected value of the referencenewReference - the new value for the referenceexpectedMark - the expected value of the marknewMark - the new value for the mark **Returns:**true if successful

### set

public void **set**([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) newReference,  
 boolean newMark)

Unconditionally sets the value of both the reference and mark.

**Parameters:**newReference - the new value for the referencenewMark - the new value for the mark

### attemptMark

public boolean **attemptMark**([V](http://docs.google.com/java/util/concurrent/atomic/AtomicMarkableReference.html) expectedReference,  
 boolean newMark)

Atomically sets the value of the mark to the given update value if the current reference is == to the expected reference. Any given invocation of this operation may fail (return false) spuriously, but repeated invocation when the current value holds the expected value and no other thread is also attempting to set the value will eventually succeed.

**Parameters:**expectedReference - the expected value of the referencenewMark - the new value for the mark **Returns:**true if successful

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/AtomicMarkableReference.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/atomic/AtomicLongFieldUpdater.html)   [**NEXT CLASS**](http://docs.google.com/java/util/concurrent/atomic/AtomicReference.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/concurrent/atomic/AtomicMarkableReference.html)    [**NO FRAMES**](http://docs.google.com/AtomicMarkableReference.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | [CONSTR](#3znysh7) | [METHOD](#2et92p0) | DETAIL: FIELD | [CONSTR](#3dy6vkm) | [METHOD](#2s8eyo1) |

[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).