**Class AtomicInteger** java.lang.Object java.lang.Number java.util.concurrent.atomic.AtomicInteger **All Implemented Interfaces:** Serializable public class AtomicInteger extends Number implements Serializable

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

ALL CLASSES

PREV CLASS NEXT CLASS

compact1, compact2, compact3

**Constructor Summary** 

**Constructor and Description** 

AtomicInteger(int initialValue)

Creates a new AtomicInteger with initial value 0.

Creates a new AtomicInteger with the given initial value.

All Methods Instance Methods Concrete Methods

**Method and Description** 

addAndGet(int delta)

decrementAndGet()

doubleValue()

floatValue()

Gets the current value.

getAndAdd(int delta)

getAndDecrement()

getAndIncrement()

incrementAndGet()

lazySet(int newValue)

Eventually sets to the given value.

intValue()

longValue()

toString()

set(int newValue)

Sets to the given value.

getAndSet(int newValue)

get()

accumulateAndGet(int x, IntBinaryOperator accumulatorFunction)

getAndAccumulate(int x, IntBinaryOperator accumulatorFunction)

Atomically sets the value to the given updated value if the current value == the expected value.

Returns the value of this AtomicInteger as a double after a widening primitive conversion.

Returns the value of this AtomicInteger as a float after a widening primitive conversion.

Returns the value of this AtomicInteger as a long after a widening primitive conversion.

Atomically sets the value to the given updated value if the current value == the expected value.

Atomically updates the current value with the results of applying the given function, returning the previous value.

Atomically updates the current value with the results of applying the given function, returning the updated value.

Atomically adds the given value to the current value.

compareAndSet(int expect, int update)

Atomically decrements by one the current value.

Atomically adds the given value to the current value.

Atomically decrements by one the current value.

Atomically increments by one the current value.

Atomically sets to the given value and returns the old value.

getAndUpdate(IntUnaryOperator updateFunction)

Atomically increments by one the current value.

Returns the value of this AtomicInteger as an int.

Returns the String representation of the current value.

updateAndGet(IntUnaryOperator updateFunction)

weakCompareAndSet(int expect, int update)

Constructors

AtomicInteger()

**Method Summary** 

**Modifier and Type** 

int

int

int

double

float

int

int

int

int

int

int

int

int

int

void

long

void

String

boolean

byteValue, shortValue

**Constructor Detail** 

AtomicInteger

**Parameters:** 

**AtomicInteger** 

**Method Detail** 

get

**Returns:** 

set

public AtomicInteger()

public final int get()

public final void set(int newValue)

public final void lazySet(int newValue)

public final int getAndSet(int newValue)

Atomically sets to the given value and returns the old value.

public final boolean compareAndSet(int expect,

public final boolean weakCompareAndSet(int expect,

int update)

Atomically sets the value to the given updated value if the current value == the expected value.

true if successful. False return indicates that the actual value was not equal to the expected value.

May fail spuriously and does not provide ordering guarantees, so is only rarely an appropriate alternative to compareAndSet.

int update)

Atomically sets the value to the given updated value if the current value == the expected value.

Gets the current value.

the current value

Sets to the given value.

newValue - the new value

Eventually sets to the given value.

newValue - the new value

newValue - the new value

expect - the expected value

expect - the expected value

public final int getAndIncrement()

public final int getAndDecrement()

Atomically decrements by one the current value.

public final int getAndAdd(int delta)

public final int incrementAndGet()

public final int decrementAndGet()

Atomically decrements by one the current value.

public final int addAndGet(int delta)

Atomically adds the given value to the current value.

updateFunction - a side-effect-free function

updateFunction - a side-effect-free function

public final int getAndAccumulate(int x,

public final int accumulateAndGet(int x,

public final int getAndUpdate(IntUnaryOperator updateFunction)

public final int updateAndGet(IntUnaryOperator updateFunction)

accumulatorFunction - a side-effect-free function of two arguments

accumulatorFunction - a side-effect-free function of two arguments

the numeric value represented by this object after conversion to type int.

Returns the value of this AtomicInteger as a long after a widening primitive conversion.

the numeric value represented by this object after conversion to type long.

Returns the value of this AtomicInteger as a float after a widening primitive conversion.

the numeric value represented by this object after conversion to type float.

Returns the value of this AtomicInteger as a double after a widening primitive conversion.

the numeric value represented by this object after conversion to type double.

ALL CLASSES

For further API reference and developer documentation, see Java SE Documentation. That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright © 1993, 2024, Oracle and/or its affiliates. All rights reserved. Use is subject to license terms. Also see the documentation redistribution policy. Modify Preferências de Cookies. Modify Ad Choices.

IntBinaryOperator accumulatorFunction)

IntBinaryOperator accumulatorFunction)

contention among threads. The function is applied with the current value as its first argument, and the given update as the second argument.

contention among threads. The function is applied with the current value as its first argument, and the given update as the second argument.

Atomically updates the current value with the results of applying the given function, returning the previous value. The function should be side-effect-free, since it may be re-applied when attempted updates fail due to contention among threads.

Atomically updates the current value with the results of applying the given function, returning the updated value. The function should be side-effect-free, since it may be re-applied when attempted updates fail due to contention among threads.

Atomically updates the current value with the results of applying the given function to the current and given values, returning the previous value. The function should be side-effect-free, since it may be re-applied when attempted updates fail due to

Atomically updates the current value with the results of applying the given function to the current and given values, returning the updated value. The function should be side-effect-free, since it may be re-applied when attempted updates fail due to

**Java™ Platform** 

**Standard Ed. 8** 

Atomically increments by one the current value.

Atomically adds the given value to the current value.

Atomically increments by one the current value.

update - the new value

true if successful

the previous value

getAndDecrement

the previous value

delta - the value to add

the previous value

incrementAndGet

the updated value

decrementAndGet

the updated value

getAndIncrement

update - the new value

weakCompareAndSet

the previous value

compareAndSet

**Parameters:** 

**Returns:** 

**Parameters:** 

**Returns:** 

**Returns:** 

**Returns:** 

getAndAdd

**Parameters:** 

**Returns:** 

**Returns:** 

**Returns:** 

addAndGet

**Parameters:** 

**Returns:** 

delta - the value to add

the updated value

getAndUpdate

**Parameters:** 

the previous value

updateAndGet

**Parameters:** 

the updated value

getAndAccumulate

**Parameters:** 

**Returns:** 

Since:

1.8

x - the update value

the previous value

accumulateAndGet

**Parameters:** 

**Returns:** 

Since:

toString

**Overrides:** 

**Returns:** 

intValue

Specified by:

**Returns:** 

longValue

Specified by:

**Returns:** 

floatValue

Specified by:

doubleValue

Specified by:

PREV CLASS NEXT CLASS

Submit a bug or feature

**Returns:** 

**Returns:** 

1.8

x - the update value

the updated value

public String toString()

toString in class Object

public int intValue()

intValue in class Number

public long longValue()

longValue in class Number

public float floatValue()

floatValue in class Number

public double doubleValue()

doubleValue in class Number

See The Java™ Language Specification:

5.1.2 Widening Primitive Conversions

OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP

FRAMES NO FRAMES

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

See The Java™ Language Specification:

5.1.2 Widening Primitive Conversions

See The Java™ Language Specification:

5.1.2 Widening Primitive Conversions

Returns the String representation of the current value.

the String representation of the current value

Returns the value of this AtomicInteger as an int.

**Returns:** 

Since:

1.8

**Returns:** 

Since:

1.8

**Parameters:** 

lazySet

**Parameters:** 

getAndSet

**Parameters:** 

**Returns:** 

Since:

1.6

Methods inherited from class java.lang.Number

**Methods inherited from class java.lang.Object** 

public AtomicInteger(int initialValue)

Creates a new AtomicInteger with initial value 0.

initialValue - the initial value

Creates a new AtomicInteger with the given initial value.

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

int

boolean

java.util.concurrent.atomic

**Java™ Platform** 

**Standard Ed. 8** 

An int value that may be updated atomically. See the java.util.concurrent.atomic package specification for description of the properties of atomic variables. An AtomicInteger is used in applications such as atomically incremented counters, and cannot be used as a replacement for an Integer. However, this class does extend Number to allow uniform access by tools and utilities that deal with numerically-based classes. Since: 1.5 See Also: Serialized Form

Atomically updates the current value with the results of applying the given function to the current and given values, returning the updated value.

Atomically updates the current value with the results of applying the given function to the current and given values, returning the previous value.