compact1, compact2, compact3
java.util.concurrent.atomic

Class AtomicInteger

java.lang.Object
  java.lang.Number
    java.util.concurrent.atomic.AtomicInteger

All implemented interfaces:
Serializable

```
public class AtomicInteger
extends Number
implements Serializable
```

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

Constructor Summary

Constructors

Constructor and Description
AtomicInteger() Creates a new AtomicInteger with initial value 0.
AtomicInteger(int initialValue) Creates a new AtomicInteger with the given initial value.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
int	accumulateAndGet(int x, IntBinaryOperator accumulatorFunction) Atomically updates the current value with the results of applying the given function to the current and given values, returning the updated value.
int	addAndGet(int delta) Atomically adds the given value to the current value.
boolean	compareAndSet(int expect, int update) Atomically sets the value to the given updated value if the current value == the expected value.
int	decrementAndGet() Atomically decrements by one the current value.
double	doubleValue() Returns the value of this AtomicInteger as a double after a widening primitive conversion.
float	floatValue() Returns the value of this AtomicInteger as a float after a widening primitive conversion.
int	get() Gets the current value.
int	getAndAccumulate(int x, IntBinaryOperator accumulatorFunction) Atomically updates the current value with the results of applying the given function to the current and given values, returning the previous value.
int	getAndAdd(int delta) Atomically adds the given value to the current value.
int	getAndDecrement() Atomically decrements by one the current value.
int	getAndIncrement() Atomically increments by one the current value.
int	getAndSet(int newValue) Atomically sets to the given value and returns the old value.
int	getAndUpdate(IntUnaryOperator updateFunction) Atomically updates the current value with the results of applying the given function, returning the previous value.
int	incrementAndGet() Atomically increments by one the current value.
int	intValue() Returns the value of this AtomicInteger as an int.
void	lazySet(int newValue) Eventually sets to the given value.
long	longValue() Returns the value of this AtomicInteger as a long after a widening primitive conversion.
void	set(int newValue) Sets to the given value.
String	toString() Returns the String representation of the current value.
int	updateAndGet(IntUnaryOperator updateFunction) Atomically updates the current value with the results of applying the given function, returning the updated value.
boolean	weakCompareAndSet(int expect, int update) Atomically sets the value to the given updated value if the current value == the expected value.

Methods inherited from class java.lang.Number

byteValue, shortValue

Methods inherited from class java.lang.Object

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

Constructor Detail

AtomicInteger

```
public AtomicInteger(int initialValue)
```

Creates a new AtomicInteger with the given initial value.

Parameters:
initialValue - the initial value

AtomicInteger

```
public AtomicInteger()
```

Creates a new AtomicInteger with initial value 0.

Method Detail

get

```
public final int get()
```

Gets the current value.

Returns:
the current value

set

```
public final void set(int newValue)
```

Sets to the given value.

Parameters:
newValue - the new value

lazySet

```
public final void lazySet(int newValue)
```

Eventually sets to the given value.

Parameters:
newValue - the new value

Since:
1.6

getAndSet

```
public final int getAndSet(int newValue)
```

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

Parameters:
newValue - the new value

Returns:
the previous value

compareAndSet

```
public final boolean compareAndSet(int expect,
                                   int update)
```

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

Parameters:
expect - the expected value
update - the new value

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

weakCompareAndSet

```
public final boolean weakCompareAndSet(int expect,
                                       int update)
```

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

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

Parameters:
expect - the expected value
update - the new value

Returns:
true if successful

getAndIncrement

```
public final int getAndIncrement()
```

Atomically increments by one the current value.

Returns:
the previous value

getAndDecrement

```
public final int getAndDecrement()
```

Atomically decrements by one the current value.

Returns:
the previous value

getAndAdd

```
public final int getAndAdd(int delta)
```

Atomically adds the given value to the current value.

Parameters:
delta - the value to add

Returns:
the previous value

incrementAndGet

```
public final int incrementAndGet()
```

Atomically increments by one the current value.

Returns:
the updated value

decrementAndGet

```
public final int decrementAndGet()
```

Atomically decrements by one the current value.

Returns:
the updated value

addAndGet

```
public final int addAndGet(int delta)
```

Atomically adds the given value to the current value.

Parameters:
delta - the value to add

Returns:
the updated value

getAndUpdate

```
public final int getAndUpdate(IntUnaryOperator updateFunction)
```

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.

Parameters:
updateFunction - a side-effect-free function

Returns:
the previous value

Since:
1.8

updateAndGet

```
public final int updateAndGet(IntUnaryOperator updateFunction)
```

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.

Parameters:
updateFunction - a side-effect-free function

Returns:
the updated value

Since:
1.8

getAndAccumulate

```
public final int getAndAccumulate(int x,
                                  IntBinaryOperator accumulatorFunction)
```

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

Parameters:
x - the update value
accumulatorFunction - a side-effect-free function of two arguments

Returns:
the previous value

Since:
1.8

accumulateAndGet

```
public final int accumulateAndGet(int x,
                                  IntBinaryOperator accumulatorFunction)
```

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

Parameters:
x - the update value
accumulatorFunction - a side-effect-free function of two arguments

Returns:
the updated value

Since:
1.8

toString

```
public String toString()
```

Returns the String representation of the current value.

Overrides:
toString in class Object

Returns:
the String representation of the current value

intValue

```
public int intValue()
```

Returns the value of this AtomicInteger as an int.

Specified by:
intValue in class Number

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

longValue

```
public long longValue()
```

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

Specified by:
longValue in class Number

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

See The Java™ Language Specification:
5.1.2 Widening Primitive Conversions

floatValue

```
public float floatValue()
```

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

Specified by:
floatValue in class Number

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

See The Java™ Language Specification:
5.1.2 Widening Primitive Conversions

doubleValue

```
public double doubleValue()
```

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

Specified by:
doubleValue in class Number

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

See The Java™ Language Specification:
5.1.2 Widening Primitive Conversions