| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Integer.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/lang/InstantiationException.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/InternalError.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Integer.html)    [**NO FRAMES**](http://docs.google.com/Integer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#26in1rg) | [METHOD](#1ksv4uv) |

## **java.lang**

Class Integer

[java.lang.Object](http://docs.google.com/java/lang/Object.html)  
 [java.lang.Number](http://docs.google.com/java/lang/Number.html)  
 **java.lang.Integer**

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

public final class **Integer**extends [Number](http://docs.google.com/java/lang/Number.html)implements [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Integer](http://docs.google.com/java/lang/Integer.html)>

The Integer class wraps a value of the primitive type int in an object. An object of type Integer contains a single field whose type is int.

In addition, this class provides several methods for converting an int to a String and a String to an int, as well as other constants and methods useful when dealing with an int.

Implementation note: The implementations of the "bit twiddling" methods (such as [highestOneBit](http://docs.google.com/java/lang/Integer.html#highestOneBit(int)) and [numberOfTrailingZeros](http://docs.google.com/java/lang/Integer.html#numberOfTrailingZeros(int))) are based on material from Henry S. Warren, Jr.'s *Hacker's Delight*, (Addison Wesley, 2002).

**Since:** JDK1.0 **See Also:**[Serialized Form](http://docs.google.com/serialized-form.html#java.lang.Integer)

| **Field Summary** | |
| --- | --- |
| static int | [**MAX\_VALUE**](http://docs.google.com/java/lang/Integer.html#MAX_VALUE)            A constant holding the maximum value an int can have, 231-1. |
| static int | [**MIN\_VALUE**](http://docs.google.com/java/lang/Integer.html#MIN_VALUE)            A constant holding the minimum value an int can have, -231. |
| static int | [**SIZE**](http://docs.google.com/java/lang/Integer.html#SIZE)            The number of bits used to represent an int value in two's complement binary form. |
| static [Class](http://docs.google.com/java/lang/Class.html)<[Integer](http://docs.google.com/java/lang/Integer.html)> | [**TYPE**](http://docs.google.com/java/lang/Integer.html#TYPE)            The Class instance representing the primitive type int. |

| **Constructor Summary** | |
| --- | --- |
| [**Integer**](http://docs.google.com/java/lang/Integer.html#Integer(int))(int value)            Constructs a newly allocated Integer object that represents the specified int value. |
| [**Integer**](http://docs.google.com/java/lang/Integer.html#Integer(java.lang.String))([String](http://docs.google.com/java/lang/String.html) s)            Constructs a newly allocated Integer object that represents the int value indicated by the String parameter. |

| **Method Summary** | |
| --- | --- |
| static int | [**bitCount**](http://docs.google.com/java/lang/Integer.html#bitCount(int))(int i)            Returns the number of one-bits in the two's complement binary representation of the specified int value. |
| byte | [**byteValue**](http://docs.google.com/java/lang/Integer.html#byteValue())()            Returns the value of this Integer as a byte. |
| int | [**compareTo**](http://docs.google.com/java/lang/Integer.html#compareTo(java.lang.Integer))([Integer](http://docs.google.com/java/lang/Integer.html) anotherInteger)            Compares two Integer objects numerically. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**decode**](http://docs.google.com/java/lang/Integer.html#decode(java.lang.String))([String](http://docs.google.com/java/lang/String.html) nm)            Decodes a String into an Integer. |
| double | [**doubleValue**](http://docs.google.com/java/lang/Integer.html#doubleValue())()            Returns the value of this Integer as a double. |
| boolean | [**equals**](http://docs.google.com/java/lang/Integer.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this object to the specified object. |
| float | [**floatValue**](http://docs.google.com/java/lang/Integer.html#floatValue())()            Returns the value of this Integer as a float. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**getInteger**](http://docs.google.com/java/lang/Integer.html#getInteger(java.lang.String))([String](http://docs.google.com/java/lang/String.html) nm)            Determines the integer value of the system property with the specified name. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**getInteger**](http://docs.google.com/java/lang/Integer.html#getInteger(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) nm, int val)            Determines the integer value of the system property with the specified name. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**getInteger**](http://docs.google.com/java/lang/Integer.html#getInteger(java.lang.String,%20java.lang.Integer))([String](http://docs.google.com/java/lang/String.html) nm, [Integer](http://docs.google.com/java/lang/Integer.html) val)            Returns the integer value of the system property with the specified name. |
| int | [**hashCode**](http://docs.google.com/java/lang/Integer.html#hashCode())()            Returns a hash code for this Integer. |
| static int | [**highestOneBit**](http://docs.google.com/java/lang/Integer.html#highestOneBit(int))(int i)            Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. |
| int | [**intValue**](http://docs.google.com/java/lang/Integer.html#intValue())()            Returns the value of this Integer as an int. |
| long | [**longValue**](http://docs.google.com/java/lang/Integer.html#longValue())()            Returns the value of this Integer as a long. |
| static int | [**lowestOneBit**](http://docs.google.com/java/lang/Integer.html#lowestOneBit(int))(int i)            Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. |
| static int | [**numberOfLeadingZeros**](http://docs.google.com/java/lang/Integer.html#numberOfLeadingZeros(int))(int i)            Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value. |
| static int | [**numberOfTrailingZeros**](http://docs.google.com/java/lang/Integer.html#numberOfTrailingZeros(int))(int i)            Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value. |
| static int | [**parseInt**](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String))([String](http://docs.google.com/java/lang/String.html) s)            Parses the string argument as a signed decimal integer. |
| static int | [**parseInt**](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) s, int radix)            Parses the string argument as a signed integer in the radix specified by the second argument. |
| static int | [**reverse**](http://docs.google.com/java/lang/Integer.html#reverse(int))(int i)            Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value. |
| static int | [**reverseBytes**](http://docs.google.com/java/lang/Integer.html#reverseBytes(int))(int i)            Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value. |
| static int | [**rotateLeft**](http://docs.google.com/java/lang/Integer.html#rotateLeft(int,%20int))(int i, int distance)            Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits. |
| static int | [**rotateRight**](http://docs.google.com/java/lang/Integer.html#rotateRight(int,%20int))(int i, int distance)            Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits. |
| short | [**shortValue**](http://docs.google.com/java/lang/Integer.html#shortValue())()            Returns the value of this Integer as a short. |
| static int | [**signum**](http://docs.google.com/java/lang/Integer.html#signum(int))(int i)            Returns the signum function of the specified int value. |
| static [String](http://docs.google.com/java/lang/String.html) | [**toBinaryString**](http://docs.google.com/java/lang/Integer.html#toBinaryString(int))(int i)            Returns a string representation of the integer argument as an unsigned integer in base 2. |
| static [String](http://docs.google.com/java/lang/String.html) | [**toHexString**](http://docs.google.com/java/lang/Integer.html#toHexString(int))(int i)            Returns a string representation of the integer argument as an unsigned integer in base 16. |
| static [String](http://docs.google.com/java/lang/String.html) | [**toOctalString**](http://docs.google.com/java/lang/Integer.html#toOctalString(int))(int i)            Returns a string representation of the integer argument as an unsigned integer in base 8. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/Integer.html#toString())()            Returns a String object representing this Integer's value. |
| static [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/Integer.html#toString(int))(int i)            Returns a String object representing the specified integer. |
| static [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/lang/Integer.html#toString(int,%20int))(int i, int radix)            Returns a string representation of the first argument in the radix specified by the second argument. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**valueOf**](http://docs.google.com/java/lang/Integer.html#valueOf(int))(int i)            Returns a Integer instance representing the specified int value. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**valueOf**](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String))([String](http://docs.google.com/java/lang/String.html) s)            Returns an Integer object holding the value of the specified String. |
| static [Integer](http://docs.google.com/java/lang/Integer.html) | [**valueOf**](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String,%20int))([String](http://docs.google.com/java/lang/String.html) s, int radix)            Returns an Integer object holding the value extracted from the specified String when parsed with the radix given by the second argument. |

| **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()), [finalize](http://docs.google.com/java/lang/Object.html#finalize()), [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)) |

| **Field Detail** |
| --- |

### MIN\_VALUE

public static final int **MIN\_VALUE**

A constant holding the minimum value an int can have, -231.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Integer.MIN_VALUE)

### MAX\_VALUE

public static final int **MAX\_VALUE**

A constant holding the maximum value an int can have, 231-1.

**See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Integer.MAX_VALUE)

### TYPE

public static final [Class](http://docs.google.com/java/lang/Class.html)<[Integer](http://docs.google.com/java/lang/Integer.html)> **TYPE**

The Class instance representing the primitive type int.

**Since:** JDK1.1

### SIZE

public static final int **SIZE**

The number of bits used to represent an int value in two's complement binary form.

**Since:** 1.5 **See Also:**[Constant Field Values](http://docs.google.com/constant-values.html#java.lang.Integer.SIZE)

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

### Integer

public **Integer**(int value)

Constructs a newly allocated Integer object that represents the specified int value.

**Parameters:**value - the value to be represented by the Integer object.

### Integer

public **Integer**([String](http://docs.google.com/java/lang/String.html) s)  
 throws [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html)

Constructs a newly allocated Integer object that represents the int value indicated by the String parameter. The string is converted to an int value in exactly the manner used by the parseInt method for radix 10.

**Parameters:**s - the String to be converted to an Integer. **Throws:** [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) - if the String does not contain a parsable integer.**See Also:**[parseInt(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String,%20int))

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

### toString

public static [String](http://docs.google.com/java/lang/String.html) **toString**(int i,  
 int radix)

Returns a string representation of the first argument in the radix specified by the second argument.

If the radix is smaller than Character.MIN\_RADIX or larger than Character.MAX\_RADIX, then the radix 10 is used instead.

If the first argument is negative, the first element of the result is the ASCII minus character '-' ('\u002D'). If the first argument is not negative, no sign character appears in the result.

The remaining characters of the result represent the magnitude of the first argument. If the magnitude is zero, it is represented by a single zero character '0' ('\u0030'); otherwise, the first character of the representation of the magnitude will not be the zero character. The following ASCII characters are used as digits:

0123456789abcdefghijklmnopqrstuvwxyz

These are '\u0030' through '\u0039' and '\u0061' through '\u007A'. If radix is N, then the first N of these characters are used as radix-N digits in the order shown. Thus, the digits for hexadecimal (radix 16) are 0123456789abcdef. If uppercase letters are desired, the [String.toUpperCase()](http://docs.google.com/java/lang/String.html#toUpperCase()) method may be called on the result:

Integer.toString(n, 16).toUpperCase()

**Parameters:**i - an integer to be converted to a string.radix - the radix to use in the string representation. **Returns:**a string representation of the argument in the specified radix.**See Also:**[Character.MAX\_RADIX](http://docs.google.com/java/lang/Character.html#MAX_RADIX), [Character.MIN\_RADIX](http://docs.google.com/java/lang/Character.html#MIN_RADIX)

### toHexString

public static [String](http://docs.google.com/java/lang/String.html) **toHexString**(int i)

Returns a string representation of the integer argument as an unsigned integer in base 16.

The unsigned integer value is the argument plus 232 if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in hexadecimal (base 16) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0' ('\u0030'); otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as hexadecimal digits:

0123456789abcdef

These are the characters '\u0030' through '\u0039' and '\u0061' through '\u0066'. If uppercase letters are desired, the [String.toUpperCase()](http://docs.google.com/java/lang/String.html#toUpperCase()) method may be called on the result:

Integer.toHexString(n).toUpperCase()

**Parameters:**i - an integer to be converted to a string. **Returns:**the string representation of the unsigned integer value represented by the argument in hexadecimal (base 16).**Since:** JDK1.0.2

### toOctalString

public static [String](http://docs.google.com/java/lang/String.html) **toOctalString**(int i)

Returns a string representation of the integer argument as an unsigned integer in base 8.

The unsigned integer value is the argument plus 232 if the argument is negative; otherwise, it is equal to the argument. This value is converted to a string of ASCII digits in octal (base 8) with no extra leading 0s.

If the unsigned magnitude is zero, it is represented by a single zero character '0' ('\u0030'); otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The following characters are used as octal digits:

01234567

These are the characters '\u0030' through '\u0037'.

**Parameters:**i - an integer to be converted to a string. **Returns:**the string representation of the unsigned integer value represented by the argument in octal (base 8).**Since:** JDK1.0.2

### toBinaryString

public static [String](http://docs.google.com/java/lang/String.html) **toBinaryString**(int i)

Returns a string representation of the integer argument as an unsigned integer in base 2.

The unsigned integer value is the argument plus 232 if the argument is negative; otherwise it is equal to the argument. This value is converted to a string of ASCII digits in binary (base 2) with no extra leading 0s. If the unsigned magnitude is zero, it is represented by a single zero character '0' ('\u0030'); otherwise, the first character of the representation of the unsigned magnitude will not be the zero character. The characters '0' ('\u0030') and '1' ('\u0031') are used as binary digits.

**Parameters:**i - an integer to be converted to a string. **Returns:**the string representation of the unsigned integer value represented by the argument in binary (base 2).**Since:** JDK1.0.2

### toString

public static [String](http://docs.google.com/java/lang/String.html) **toString**(int i)

Returns a String object representing the specified integer. The argument is converted to signed decimal representation and returned as a string, exactly as if the argument and radix 10 were given as arguments to the [toString(int, int)](http://docs.google.com/java/lang/Integer.html#toString(int,%20int)) method.

**Parameters:**i - an integer to be converted. **Returns:**a string representation of the argument in base 10.

### parseInt

public static int **parseInt**([String](http://docs.google.com/java/lang/String.html) s,  
 int radix)  
 throws [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html)

Parses the string argument as a signed integer in the radix specified by the second argument. The characters in the string must all be digits of the specified radix (as determined by whether [Character.digit(char, int)](http://docs.google.com/java/lang/Character.html#digit(char,%20int)) returns a nonnegative value), except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value. The resulting integer value is returned.

An exception of type NumberFormatException is thrown if any of the following situations occurs:

* The first argument is null or is a string of length zero.
* The radix is either smaller than [Character.MIN\_RADIX](http://docs.google.com/java/lang/Character.html#MIN_RADIX) or larger than [Character.MAX\_RADIX](http://docs.google.com/java/lang/Character.html#MAX_RADIX).
* Any character of the string is not a digit of the specified radix, except that the first character may be a minus sign '-' ('\u002D') provided that the string is longer than length 1.
* The value represented by the string is not a value of type int.

Examples:

parseInt("0", 10) returns 0  
 parseInt("473", 10) returns 473  
 parseInt("-0", 10) returns 0  
 parseInt("-FF", 16) returns -255  
 parseInt("1100110", 2) returns 102  
 parseInt("2147483647", 10) returns 2147483647  
 parseInt("-2147483648", 10) returns -2147483648  
 parseInt("2147483648", 10) throws a NumberFormatException  
 parseInt("99", 8) throws a NumberFormatException  
 parseInt("Kona", 10) throws a NumberFormatException  
 parseInt("Kona", 27) returns 411787

**Parameters:**s - the String containing the integer representation to be parsedradix - the radix to be used while parsing s. **Returns:**the integer represented by the string argument in the specified radix. **Throws:** [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) - if the String does not contain a parsable int.

### parseInt

public static int **parseInt**([String](http://docs.google.com/java/lang/String.html) s)  
 throws [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html)

Parses the string argument as a signed decimal integer. The characters in the string must all be decimal digits, except that the first character may be an ASCII minus sign '-' ('\u002D') to indicate a negative value. The resulting integer value is returned, exactly as if the argument and the radix 10 were given as arguments to the [parseInt(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String,%20int)) method.

**Parameters:**s - a String containing the int representation to be parsed **Returns:**the integer value represented by the argument in decimal. **Throws:** [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) - if the string does not contain a parsable integer.

### valueOf

public static [Integer](http://docs.google.com/java/lang/Integer.html) **valueOf**([String](http://docs.google.com/java/lang/String.html) s,  
 int radix)  
 throws [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html)

Returns an Integer object holding the value extracted from the specified String when parsed with the radix given by the second argument. The first argument is interpreted as representing a signed integer in the radix specified by the second argument, exactly as if the arguments were given to the [parseInt(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String,%20int)) method. The result is an Integer object that represents the integer value specified by the string.

In other words, this method returns an Integer object equal to the value of:

new Integer(Integer.parseInt(s, radix))

**Parameters:**s - the string to be parsed.radix - the radix to be used in interpreting s **Returns:**an Integer object holding the value represented by the string argument in the specified radix. **Throws:** [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) - if the String does not contain a parsable int.

### valueOf

public static [Integer](http://docs.google.com/java/lang/Integer.html) **valueOf**([String](http://docs.google.com/java/lang/String.html) s)  
 throws [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html)

Returns an Integer object holding the value of the specified String. The argument is interpreted as representing a signed decimal integer, exactly as if the argument were given to the [parseInt(java.lang.String)](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String)) method. The result is an Integer object that represents the integer value specified by the string.

In other words, this method returns an Integer object equal to the value of:

new Integer(Integer.parseInt(s))

**Parameters:**s - the string to be parsed. **Returns:**an Integer object holding the value represented by the string argument. **Throws:** [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) - if the string cannot be parsed as an integer.

### valueOf

public static [Integer](http://docs.google.com/java/lang/Integer.html) **valueOf**(int i)

Returns a Integer instance representing the specified int value. If a new Integer instance is not required, this method should generally be used in preference to the constructor [Integer(int)](http://docs.google.com/java/lang/Integer.html#Integer(int)), as this method is likely to yield significantly better space and time performance by caching frequently requested values.

**Parameters:**i - an int value. **Returns:**a Integer instance representing i.**Since:** 1.5

### byteValue

public byte **byteValue**()

Returns the value of this Integer as a byte.

**Overrides:**[byteValue](http://docs.google.com/java/lang/Number.html#byteValue()) in class [Number](http://docs.google.com/java/lang/Number.html) **Returns:**the numeric value represented by this object after conversion to type byte.

### shortValue

public short **shortValue**()

Returns the value of this Integer as a short.

**Overrides:**[shortValue](http://docs.google.com/java/lang/Number.html#shortValue()) in class [Number](http://docs.google.com/java/lang/Number.html) **Returns:**the numeric value represented by this object after conversion to type short.

### intValue

public int **intValue**()

Returns the value of this Integer as an int.

**Specified by:**[intValue](http://docs.google.com/java/lang/Number.html#intValue()) in class [Number](http://docs.google.com/java/lang/Number.html) **Returns:**the numeric value represented by this object after conversion to type int.

### longValue

public long **longValue**()

Returns the value of this Integer as a long.

**Specified by:**[longValue](http://docs.google.com/java/lang/Number.html#longValue()) in class [Number](http://docs.google.com/java/lang/Number.html) **Returns:**the numeric value represented by this object after conversion to type long.

### floatValue

public float **floatValue**()

Returns the value of this Integer as a float.

**Specified by:**[floatValue](http://docs.google.com/java/lang/Number.html#floatValue()) in class [Number](http://docs.google.com/java/lang/Number.html) **Returns:**the numeric value represented by this object after conversion to type float.

### doubleValue

public double **doubleValue**()

Returns the value of this Integer as a double.

**Specified by:**[doubleValue](http://docs.google.com/java/lang/Number.html#doubleValue()) in class [Number](http://docs.google.com/java/lang/Number.html) **Returns:**the numeric value represented by this object after conversion to type double.

### toString

public [String](http://docs.google.com/java/lang/String.html) **toString**()

Returns a String object representing this Integer's value. The value is converted to signed decimal representation and returned as a string, exactly as if the integer value were given as an argument to the [toString(int)](http://docs.google.com/java/lang/Integer.html#toString(int)) method.

**Overrides:**[toString](http://docs.google.com/java/lang/Object.html#toString()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a string representation of the value of this object in base 10.

### hashCode

public int **hashCode**()

Returns a hash code for this Integer.

**Overrides:**[hashCode](http://docs.google.com/java/lang/Object.html#hashCode()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a hash code value for this object, equal to the primitive int value represented by this Integer object.**See Also:**[Object.equals(java.lang.Object)](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### equals

public boolean **equals**([Object](http://docs.google.com/java/lang/Object.html) obj)

Compares this object to the specified object. The result is true if and only if the argument is not null and is an Integer object that contains the same int value as this object.

**Overrides:**[equals](http://docs.google.com/java/lang/Object.html#equals(java.lang.Object)) in class [Object](http://docs.google.com/java/lang/Object.html) **Parameters:**obj - the object to compare with. **Returns:**true if the objects are the same; false otherwise.**See Also:**[Object.hashCode()](http://docs.google.com/java/lang/Object.html#hashCode()), [Hashtable](http://docs.google.com/java/util/Hashtable.html)

### getInteger

public static [Integer](http://docs.google.com/java/lang/Integer.html) **getInteger**([String](http://docs.google.com/java/lang/String.html) nm)

Determines the integer value of the system property with the specified name.

The first argument is treated as the name of a system property. System properties are accessible through the [System.getProperty(java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String)) method. The string value of this property is then interpreted as an integer value and an Integer object representing this value is returned. Details of possible numeric formats can be found with the definition of getProperty.

If there is no property with the specified name, if the specified name is empty or null, or if the property does not have the correct numeric format, then null is returned.

In other words, this method returns an Integer object equal to the value of:

getInteger(nm, null)

**Parameters:**nm - property name. **Returns:**the Integer value of the property.**See Also:**[System.getProperty(java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String)), [System.getProperty(java.lang.String, java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String,%20java.lang.String))

### getInteger

public static [Integer](http://docs.google.com/java/lang/Integer.html) **getInteger**([String](http://docs.google.com/java/lang/String.html) nm,  
 int val)

Determines the integer value of the system property with the specified name.

The first argument is treated as the name of a system property. System properties are accessible through the [System.getProperty(java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String)) method. The string value of this property is then interpreted as an integer value and an Integer object representing this value is returned. Details of possible numeric formats can be found with the definition of getProperty.

The second argument is the default value. An Integer object that represents the value of the second argument is returned if there is no property of the specified name, if the property does not have the correct numeric format, or if the specified name is empty or null.

In other words, this method returns an Integer object equal to the value of:

getInteger(nm, new Integer(val))but in practice it may be implemented in a manner such as:

Integer result = getInteger(nm, null);  
 return (result == null) ? new Integer(val) : result;

to avoid the unnecessary allocation of an Integer object when the default value is not needed.

**Parameters:**nm - property name.val - default value. **Returns:**the Integer value of the property.**See Also:**[System.getProperty(java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String)), [System.getProperty(java.lang.String, java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String,%20java.lang.String))

### getInteger

public static [Integer](http://docs.google.com/java/lang/Integer.html) **getInteger**([String](http://docs.google.com/java/lang/String.html) nm,  
 [Integer](http://docs.google.com/java/lang/Integer.html) val)

Returns the integer value of the system property with the specified name. The first argument is treated as the name of a system property. System properties are accessible through the [System.getProperty(java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String)) method. The string value of this property is then interpreted as an integer value, as per the Integer.decode method, and an Integer object representing this value is returned.

* If the property value begins with the two ASCII characters 0x or the ASCII character #, not followed by a minus sign, then the rest of it is parsed as a hexadecimal integer exactly as by the method [valueOf(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String,%20int)) with radix 16.
* If the property value begins with the ASCII character 0 followed by another character, it is parsed as an octal integer exactly as by the method [valueOf(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String,%20int)) with radix 8.
* Otherwise, the property value is parsed as a decimal integer exactly as by the method [valueOf(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#valueOf(java.lang.String,%20int)) with radix 10.

The second argument is the default value. The default value is returned if there is no property of the specified name, if the property does not have the correct numeric format, or if the specified name is empty or null.

**Parameters:**nm - property name.val - default value. **Returns:**the Integer value of the property.**See Also:**[System.getProperty(java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String)), [System.getProperty(java.lang.String, java.lang.String)](http://docs.google.com/java/lang/System.html#getProperty(java.lang.String,%20java.lang.String)), [decode(java.lang.String)](http://docs.google.com/java/lang/Integer.html#decode(java.lang.String))

### decode

public static [Integer](http://docs.google.com/java/lang/Integer.html) **decode**([String](http://docs.google.com/java/lang/String.html) nm)  
 throws [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html)

Decodes a String into an Integer. Accepts decimal, hexadecimal, and octal numbers given by the following grammar:*DecodableString:* *Signopt DecimalNumeral* *Signopt* 0x *HexDigits* *Signopt* 0X *HexDigits* *Signopt* # *HexDigits* *Signopt* 0 *OctalDigits*

*Sign:* -*DecimalNumeral*, *HexDigits*, and *OctalDigits* are defined in [§3.10.1](http://java.sun.com/docs/books/jls/second_edition/html/lexical.doc.html#48282) of the [Java Language Specification](http://java.sun.com/docs/books/jls/html/).

The sequence of characters following an (optional) negative sign and/or radix specifier ("0x", "0X", "#", or leading zero) is parsed as by the Integer.parseInt method with the indicated radix (10, 16, or 8). This sequence of characters must represent a positive value or a [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) will be thrown. The result is negated if first character of the specified String is the minus sign. No whitespace characters are permitted in the String.

**Parameters:**nm - the String to decode. **Returns:**a Integer object holding the int value represented by nm **Throws:** [NumberFormatException](http://docs.google.com/java/lang/NumberFormatException.html) - if the String does not contain a parsable integer.**See Also:**[parseInt(java.lang.String, int)](http://docs.google.com/java/lang/Integer.html#parseInt(java.lang.String,%20int))

### compareTo

public int **compareTo**([Integer](http://docs.google.com/java/lang/Integer.html) anotherInteger)

Compares two Integer objects numerically.

**Specified by:**[compareTo](http://docs.google.com/java/lang/Comparable.html#compareTo(T)) in interface [Comparable](http://docs.google.com/java/lang/Comparable.html)<[Integer](http://docs.google.com/java/lang/Integer.html)> **Parameters:**anotherInteger - the Integer to be compared. **Returns:**the value 0 if this Integer is equal to the argument Integer; a value less than 0 if this Integer is numerically less than the argument Integer; and a value greater than 0 if this Integer is numerically greater than the argument Integer (signed comparison).**Since:** 1.2

### highestOneBit

public static int **highestOneBit**(int i)

Returns an int value with at most a single one-bit, in the position of the highest-order ("leftmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Returns:**an int value with a single one-bit, in the position of the highest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.**Since:** 1.5

### lowestOneBit

public static int **lowestOneBit**(int i)

Returns an int value with at most a single one-bit, in the position of the lowest-order ("rightmost") one-bit in the specified int value. Returns zero if the specified value has no one-bits in its two's complement binary representation, that is, if it is equal to zero.

**Returns:**an int value with a single one-bit, in the position of the lowest-order one-bit in the specified value, or zero if the specified value is itself equal to zero.**Since:** 1.5

### numberOfLeadingZeros

public static int **numberOfLeadingZeros**(int i)

Returns the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

Note that this method is closely related to the logarithm base 2. For all positive int values x:

* floor(log2(x)) = 31 - numberOfLeadingZeros(x)
* ceil(log2(x)) = 32 - numberOfLeadingZeros(x - 1)

**Returns:**the number of zero bits preceding the highest-order ("leftmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.**Since:** 1.5

### numberOfTrailingZeros

public static int **numberOfTrailingZeros**(int i)

Returns the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value. Returns 32 if the specified value has no one-bits in its two's complement representation, in other words if it is equal to zero.

**Returns:**the number of zero bits following the lowest-order ("rightmost") one-bit in the two's complement binary representation of the specified int value, or 32 if the value is equal to zero.**Since:** 1.5

### bitCount

public static int **bitCount**(int i)

Returns the number of one-bits in the two's complement binary representation of the specified int value. This function is sometimes referred to as the *population count*.

**Returns:**the number of one-bits in the two's complement binary representation of the specified int value.**Since:** 1.5

### rotateLeft

public static int **rotateLeft**(int i,  
 int distance)

Returns the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits. (Bits shifted out of the left hand, or high-order, side reenter on the right, or low-order.)

Note that left rotation with a negative distance is equivalent to right rotation: rotateLeft(val, -distance) == rotateRight(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateLeft(val, distance) == rotateLeft(val, distance & 0x1F).

**Returns:**the value obtained by rotating the two's complement binary representation of the specified int value left by the specified number of bits.**Since:** 1.5

### rotateRight

public static int **rotateRight**(int i,  
 int distance)

Returns the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits. (Bits shifted out of the right hand, or low-order, side reenter on the left, or high-order.)

Note that right rotation with a negative distance is equivalent to left rotation: rotateRight(val, -distance) == rotateLeft(val, distance). Note also that rotation by any multiple of 32 is a no-op, so all but the last five bits of the rotation distance can be ignored, even if the distance is negative: rotateRight(val, distance) == rotateRight(val, distance & 0x1F).

**Returns:**the value obtained by rotating the two's complement binary representation of the specified int value right by the specified number of bits.**Since:** 1.5

### reverse

public static int **reverse**(int i)

Returns the value obtained by reversing the order of the bits in the two's complement binary representation of the specified int value.

**Returns:**the value obtained by reversing order of the bits in the specified int value.**Since:** 1.5

### signum

public static int **signum**(int i)

Returns the signum function of the specified int value. (The return value is -1 if the specified value is negative; 0 if the specified value is zero; and 1 if the specified value is positive.)

**Returns:**the signum function of the specified int value.**Since:** 1.5

### reverseBytes

public static int **reverseBytes**(int i)

Returns the value obtained by reversing the order of the bytes in the two's complement representation of the specified int value.

**Returns:**the value obtained by reversing the bytes in the specified int value.**Since:** 1.5

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/Integer.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/lang/InstantiationException.html)   [**NEXT CLASS**](http://docs.google.com/java/lang/InternalError.html) | [**FRAMES**](http://docs.google.com/index.html?java/lang/Integer.html)    [**NO FRAMES**](http://docs.google.com/Integer.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | [FIELD](#3znysh7) | [CONSTR](#2et92p0) | [METHOD](#tyjcwt) | DETAIL: [FIELD](#1t3h5sf) | [CONSTR](#26in1rg) | [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).