| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BitSet.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/Arrays.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Calendar.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/BitSet.html)    [**NO FRAMES**](http://docs.google.com/BitSet.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**

Class BitSet

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

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

public class **BitSet**extends [Object](http://docs.google.com/java/lang/Object.html)implements [Cloneable](http://docs.google.com/java/lang/Cloneable.html), [Serializable](http://docs.google.com/java/io/Serializable.html)

This class implements a vector of bits that grows as needed. Each component of the bit set has a boolean value. The bits of a BitSet are indexed by nonnegative integers. Individual indexed bits can be examined, set, or cleared. One BitSet may be used to modify the contents of another BitSet through logical AND, logical inclusive OR, and logical exclusive OR operations.

By default, all bits in the set initially have the value false.

Every bit set has a current size, which is the number of bits of space currently in use by the bit set. Note that the size is related to the implementation of a bit set, so it may change with implementation. The length of a bit set relates to logical length of a bit set and is defined independently of implementation.

Unless otherwise noted, passing a null parameter to any of the methods in a BitSet will result in a NullPointerException.

A BitSet is not safe for multithreaded use without external synchronization.

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

| **Constructor Summary** | |
| --- | --- |
| [**BitSet**](http://docs.google.com/java/util/BitSet.html#BitSet())()            Creates a new bit set. |
| [**BitSet**](http://docs.google.com/java/util/BitSet.html#BitSet(int))(int nbits)            Creates a bit set whose initial size is large enough to explicitly represent bits with indices in the range 0 through nbits-1. |

| **Method Summary** | |
| --- | --- |
| void | [**and**](http://docs.google.com/java/util/BitSet.html#and(java.util.BitSet))([BitSet](http://docs.google.com/java/util/BitSet.html) set)            Performs a logical **AND** of this target bit set with the argument bit set. |
| void | [**andNot**](http://docs.google.com/java/util/BitSet.html#andNot(java.util.BitSet))([BitSet](http://docs.google.com/java/util/BitSet.html) set)            Clears all of the bits in this BitSet whose corresponding bit is set in the specified BitSet. |
| int | [**cardinality**](http://docs.google.com/java/util/BitSet.html#cardinality())()            Returns the number of bits set to true in this BitSet. |
| void | [**clear**](http://docs.google.com/java/util/BitSet.html#clear())()            Sets all of the bits in this BitSet to false. |
| void | [**clear**](http://docs.google.com/java/util/BitSet.html#clear(int))(int bitIndex)            Sets the bit specified by the index to false. |
| void | [**clear**](http://docs.google.com/java/util/BitSet.html#clear(int,%20int))(int fromIndex, int toIndex)            Sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to false. |
| [Object](http://docs.google.com/java/lang/Object.html) | [**clone**](http://docs.google.com/java/util/BitSet.html#clone())()            Cloning this BitSet produces a new BitSet that is equal to it. |
| boolean | [**equals**](http://docs.google.com/java/util/BitSet.html#equals(java.lang.Object))([Object](http://docs.google.com/java/lang/Object.html) obj)            Compares this object against the specified object. |
| void | [**flip**](http://docs.google.com/java/util/BitSet.html#flip(int))(int bitIndex)            Sets the bit at the specified index to the complement of its current value. |
| void | [**flip**](http://docs.google.com/java/util/BitSet.html#flip(int,%20int))(int fromIndex, int toIndex)            Sets each bit from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to the complement of its current value. |
| boolean | [**get**](http://docs.google.com/java/util/BitSet.html#get(int))(int bitIndex)            Returns the value of the bit with the specified index. |
| [BitSet](http://docs.google.com/java/util/BitSet.html) | [**get**](http://docs.google.com/java/util/BitSet.html#get(int,%20int))(int fromIndex, int toIndex)            Returns a new BitSet composed of bits from this BitSet from fromIndex (inclusive) to toIndex (exclusive). |
| int | [**hashCode**](http://docs.google.com/java/util/BitSet.html#hashCode())()            Returns a hash code value for this bit set. |
| boolean | [**intersects**](http://docs.google.com/java/util/BitSet.html#intersects(java.util.BitSet))([BitSet](http://docs.google.com/java/util/BitSet.html) set)            Returns true if the specified BitSet has any bits set to true that are also set to true in this BitSet. |
| boolean | [**isEmpty**](http://docs.google.com/java/util/BitSet.html#isEmpty())()            Returns true if this BitSet contains no bits that are set to true. |
| int | [**length**](http://docs.google.com/java/util/BitSet.html#length())()            Returns the "logical size" of this BitSet: the index of the highest set bit in the BitSet plus one. |
| int | [**nextClearBit**](http://docs.google.com/java/util/BitSet.html#nextClearBit(int))(int fromIndex)            Returns the index of the first bit that is set to false that occurs on or after the specified starting index. |
| int | [**nextSetBit**](http://docs.google.com/java/util/BitSet.html#nextSetBit(int))(int fromIndex)            Returns the index of the first bit that is set to true that occurs on or after the specified starting index. |
| void | [**or**](http://docs.google.com/java/util/BitSet.html#or(java.util.BitSet))([BitSet](http://docs.google.com/java/util/BitSet.html) set)            Performs a logical **OR** of this bit set with the bit set argument. |
| void | [**set**](http://docs.google.com/java/util/BitSet.html#set(int))(int bitIndex)            Sets the bit at the specified index to true. |
| void | [**set**](http://docs.google.com/java/util/BitSet.html#set(int,%20boolean))(int bitIndex, boolean value)            Sets the bit at the specified index to the specified value. |
| void | [**set**](http://docs.google.com/java/util/BitSet.html#set(int,%20int))(int fromIndex, int toIndex)            Sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to true. |
| void | [**set**](http://docs.google.com/java/util/BitSet.html#set(int,%20int,%20boolean))(int fromIndex, int toIndex, boolean value)            Sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to the specified value. |
| int | [**size**](http://docs.google.com/java/util/BitSet.html#size())()            Returns the number of bits of space actually in use by this BitSet to represent bit values. |
| [String](http://docs.google.com/java/lang/String.html) | [**toString**](http://docs.google.com/java/util/BitSet.html#toString())()            Returns a string representation of this bit set. |
| void | [**xor**](http://docs.google.com/java/util/BitSet.html#xor(java.util.BitSet))([BitSet](http://docs.google.com/java/util/BitSet.html) set)            Performs a logical **XOR** of this bit set with the bit set argument. |

| **Methods inherited from class java.lang.**[**Object**](http://docs.google.com/java/lang/Object.html) |
| --- |
| [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)) |

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

### BitSet

public **BitSet**()

Creates a new bit set. All bits are initially false.

### BitSet

public **BitSet**(int nbits)

Creates a bit set whose initial size is large enough to explicitly represent bits with indices in the range 0 through nbits-1. All bits are initially false.

**Parameters:**nbits - the initial size of the bit set. **Throws:** [NegativeArraySizeException](http://docs.google.com/java/lang/NegativeArraySizeException.html) - if the specified initial size is negative.

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

### flip

public void **flip**(int bitIndex)

Sets the bit at the specified index to the complement of its current value.

**Parameters:**bitIndex - the index of the bit to flip. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.**Since:** 1.4

### flip

public void **flip**(int fromIndex,  
 int toIndex)

Sets each bit from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to the complement of its current value.

**Parameters:**fromIndex - index of the first bit to flip.toIndex - index after the last bit to flip. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if fromIndex is negative, or toIndex is negative, or fromIndex is larger than toIndex.**Since:** 1.4

### set

public void **set**(int bitIndex)

Sets the bit at the specified index to true.

**Parameters:**bitIndex - a bit index. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.**Since:** JDK1.0

### set

public void **set**(int bitIndex,  
 boolean value)

Sets the bit at the specified index to the specified value.

**Parameters:**bitIndex - a bit index.value - a boolean value to set. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.**Since:** 1.4

### set

public void **set**(int fromIndex,  
 int toIndex)

Sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to true.

**Parameters:**fromIndex - index of the first bit to be set.toIndex - index after the last bit to be set. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if fromIndex is negative, or toIndex is negative, or fromIndex is larger than toIndex.**Since:** 1.4

### set

public void **set**(int fromIndex,  
 int toIndex,  
 boolean value)

Sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to the specified value.

**Parameters:**fromIndex - index of the first bit to be set.toIndex - index after the last bit to be setvalue - value to set the selected bits to **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if fromIndex is negative, or toIndex is negative, or fromIndex is larger than toIndex.**Since:** 1.4

### clear

public void **clear**(int bitIndex)

Sets the bit specified by the index to false.

**Parameters:**bitIndex - the index of the bit to be cleared. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.**Since:** JDK1.0

### clear

public void **clear**(int fromIndex,  
 int toIndex)

Sets the bits from the specified fromIndex (inclusive) to the specified toIndex (exclusive) to false.

**Parameters:**fromIndex - index of the first bit to be cleared.toIndex - index after the last bit to be cleared. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if fromIndex is negative, or toIndex is negative, or fromIndex is larger than toIndex.**Since:** 1.4

### clear

public void **clear**()

Sets all of the bits in this BitSet to false.

**Since:** 1.4

### get

public boolean **get**(int bitIndex)

Returns the value of the bit with the specified index. The value is true if the bit with the index bitIndex is currently set in this BitSet; otherwise, the result is false.

**Parameters:**bitIndex - the bit index. **Returns:**the value of the bit with the specified index. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.

### get

public [BitSet](http://docs.google.com/java/util/BitSet.html) **get**(int fromIndex,  
 int toIndex)

Returns a new BitSet composed of bits from this BitSet from fromIndex (inclusive) to toIndex (exclusive).

**Parameters:**fromIndex - index of the first bit to include.toIndex - index after the last bit to include. **Returns:**a new BitSet from a range of this BitSet. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if fromIndex is negative, or toIndex is negative, or fromIndex is larger than toIndex.**Since:** 1.4

### nextSetBit

public int **nextSetBit**(int fromIndex)

Returns the index of the first bit that is set to true that occurs on or after the specified starting index. If no such bit exists then -1 is returned. To iterate over the true bits in a BitSet, use the following loop:

for (int i = bs.nextSetBit(0); i >= 0; i = bs.nextSetBit(i+1)) {  
 // operate on index i here  
 }

**Parameters:**fromIndex - the index to start checking from (inclusive). **Returns:**the index of the next set bit. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.**Since:** 1.4

### nextClearBit

public int **nextClearBit**(int fromIndex)

Returns the index of the first bit that is set to false that occurs on or after the specified starting index.

**Parameters:**fromIndex - the index to start checking from (inclusive). **Returns:**the index of the next clear bit. **Throws:** [IndexOutOfBoundsException](http://docs.google.com/java/lang/IndexOutOfBoundsException.html) - if the specified index is negative.**Since:** 1.4

### length

public int **length**()

Returns the "logical size" of this BitSet: the index of the highest set bit in the BitSet plus one. Returns zero if the BitSet contains no set bits.

**Returns:**the logical size of this BitSet.**Since:** 1.2

### isEmpty

public boolean **isEmpty**()

Returns true if this BitSet contains no bits that are set to true.

**Returns:**boolean indicating whether this BitSet is empty.**Since:** 1.4

### intersects

public boolean **intersects**([BitSet](http://docs.google.com/java/util/BitSet.html) set)

Returns true if the specified BitSet has any bits set to true that are also set to true in this BitSet.

**Parameters:**set - BitSet to intersect with **Returns:**boolean indicating whether this BitSet intersects the specified BitSet.**Since:** 1.4

### cardinality

public int **cardinality**()

Returns the number of bits set to true in this BitSet.

**Returns:**the number of bits set to true in this BitSet.**Since:** 1.4

### and

public void **and**([BitSet](http://docs.google.com/java/util/BitSet.html) set)

Performs a logical **AND** of this target bit set with the argument bit set. This bit set is modified so that each bit in it has the value true if and only if it both initially had the value true and the corresponding bit in the bit set argument also had the value true.

**Parameters:**set - a bit set.

### or

public void **or**([BitSet](http://docs.google.com/java/util/BitSet.html) set)

Performs a logical **OR** of this bit set with the bit set argument. This bit set is modified so that a bit in it has the value true if and only if it either already had the value true or the corresponding bit in the bit set argument has the value true.

**Parameters:**set - a bit set.

### xor

public void **xor**([BitSet](http://docs.google.com/java/util/BitSet.html) set)

Performs a logical **XOR** of this bit set with the bit set argument. This bit set is modified so that a bit in it has the value true if and only if one of the following statements holds:

* The bit initially has the value true, and the corresponding bit in the argument has the value false.
* The bit initially has the value false, and the corresponding bit in the argument has the value true.

**Parameters:**set - a bit set.

### andNot

public void **andNot**([BitSet](http://docs.google.com/java/util/BitSet.html) set)

Clears all of the bits in this BitSet whose corresponding bit is set in the specified BitSet.

**Parameters:**set - the BitSet with which to mask this BitSet.**Since:** 1.2

### hashCode

public int **hashCode**()

Returns a hash code value for this bit set. The hash code depends only on which bits have been set within this BitSet. The algorithm used to compute it may be described as follows.

Suppose the bits in the BitSet were to be stored in an array of long integers called, say, words, in such a manner that bit k is set in the BitSet (for nonnegative values of k) if and only if the expression

((k>>6) < words.length) && ((words[k>>6] & (1L << (bit & 0x3F))) != 0)

is true. Then the following definition of the hashCode method would be a correct implementation of the actual algorithm:

public int hashCode() {  
 long h = 1234;  
 for (int i = words.length; --i >= 0; ) {  
 h ^= words[i] \* (i + 1);  
 }  
 return (int)((h >> 32) ^ h);  
 }

Note that the hash code values change if the set of bits is altered.

Overrides the hashCode method of Object.

**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 bit set.**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)

### size

public int **size**()

Returns the number of bits of space actually in use by this BitSet to represent bit values. The maximum element in the set is the size - 1st element.

**Returns:**the number of bits currently in this bit set.

### equals

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

Compares this object against the specified object. The result is true if and only if the argument is not null and is a Bitset object that has exactly the same set of bits set to true as this bit set. That is, for every nonnegative int index k,

((BitSet)obj).get(k) == this.get(k)

must be true. The current sizes of the two bit sets are not compared.

Overrides the equals method of 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:**[size()](http://docs.google.com/java/util/BitSet.html#size())

### clone

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

Cloning this BitSet produces a new BitSet that is equal to it. The clone of the bit set is another bit set that has exactly the same bits set to true as this bit set.

Overrides the clone method of Object.

**Overrides:**[clone](http://docs.google.com/java/lang/Object.html#clone()) in class [Object](http://docs.google.com/java/lang/Object.html) **Returns:**a clone of this bit set.**See Also:**[size()](http://docs.google.com/java/util/BitSet.html#size())

### toString

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

Returns a string representation of this bit set. For every index for which this BitSet contains a bit in the set state, the decimal representation of that index is included in the result. Such indices are listed in order from lowest to highest, separated by ", " (a comma and a space) and surrounded by braces, resulting in the usual mathematical notation for a set of integers.

Overrides the toString method of Object.

Example:

BitSet drPepper = new BitSet();

Now drPepper.toString() returns "{}".

drPepper.set(2);

Now drPepper.toString() returns "{2}".

drPepper.set(4);  
 drPepper.set(10);

Now drPepper.toString() returns "{2, 4, 10}".

**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 this bit set.

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/BitSet.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/Arrays.html)   [**NEXT CLASS**](http://docs.google.com/java/util/Calendar.html) | [**FRAMES**](http://docs.google.com/index.html?java/util/BitSet.html)    [**NO FRAMES**](http://docs.google.com/BitSet.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).