Java - The BitSet Class

The BitSet class creates a special type of array that holds bit values. The BitSet array can increase in size as needed. This makes it similar to a vector of bits. This is a legacy class but it has been completely re-engineered in Java 2, version 1.4.

The BitSet defines the following two constructors.

|  |  |
| --- | --- |
| **Sr.No.** | **Constructor & Description** |
| 1 | **BitSet( )**  This constructor creates a default object. |
| 2 | **BitSet(int size)**  This constructor allows you to specify its initial size, i.e., the number of bits that it can hold. All bits are initialized to zero. |

BitSet implements the Cloneable interface and defines the methods listed in the following table −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **void and(BitSet bitSet)**  ANDs the contents of the invoking BitSet object with those specified by bitSet. The result is placed into the invoking object. |
| 2 | **void andNot(BitSet bitSet)**  For each 1 bit in bitSet, the corresponding bit in the invoking BitSet is cleared. |
| 3 | **int cardinality( )**  Returns the number of set bits in the invoking object. |
| 4 | **void clear( )**  Zeros all bits. |
| 5 | **void clear(int index)**  Zeros the bit specified by index. |
| 6 | **void clear(int startIndex, int endIndex)**  Zeros the bits from startIndex to endIndex. |
| 7 | **Object clone( )**  Duplicates the invoking BitSet object. |
| 8 | **boolean equals(Object bitSet)**  Returns true if the invoking bit set is equivalent to the one passed in bitSet. Otherwise, the method returns false. |
| 9 | **void flip(int index)**  Reverses the bit specified by the index. |
| 10 | **void flip(int startIndex, int endIndex)**  Reverses the bits from startIndex to endIndex. |
| 11 | **boolean get(int index)**  Returns the current state of the bit at the specified index. |
| 12 | **BitSet get(int startIndex, int endIndex)**  Returns a BitSet that consists of the bits from startIndex to endIndex. The invoking object is not changed. |
| 13 | **int hashCode( )**  Returns the hash code for the invoking object. |
| 14 | **boolean intersects(BitSet bitSet)**  Returns true if at least one pair of corresponding bits within the invoking object and bitSet are 1. |
| 15 | **boolean isEmpty( )**  Returns true if all bits in the invoking object are zero. |
| 16 | **int length( )**  Returns the number of bits required to hold the contents of the invoking BitSet. This value is determined by the location of the last 1 bit. |
| 17 | **int nextClearBit(int startIndex)**  Returns the index of the next cleared bit, (that is, the next zero bit), starting from the index specified by startIndex. |
| 18 | **int nextSetBit(int startIndex)**  Returns the index of the next set bit (that is, the next 1 bit), starting from the index specified by startIndex. If no bit is set, -1 is returned. |
| 19 | **void or(BitSet bitSet)**  ORs the contents of the invoking BitSet object with that specified by bitSet. The result is placed into the invoking object. |
| 20 | **void set(int index)**  Sets the bit specified by index. |
| 21 | **void set(int index, boolean v)**  Sets the bit specified by index to the value passed in v. True sets the bit, false clears the bit. |
| 22 | **void set(int startIndex, int endIndex)**  Sets the bits from startIndex to endIndex. |
| 23 | **void set(int startIndex, int endIndex, boolean v)**  Sets the bits from startIndex to endIndex, to the value passed in v. true sets the bits, false clears the bits. |
| 24 | **int size( )**  Returns the number of bits in the invoking BitSet object. |
| 25 | **String toString( )**  Returns the string equivalent of the invoking BitSet object. |
| 26 | **void xor(BitSet bitSet)**  XORs the contents of the invoking BitSet object with that specified by bitSet. The result is placed into the invoking object. |

**int cardinality( )**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);  *//cardinality* **int** x = bits2.cardinality();  **int** y = bits1.cardinality();  System.***out***.println(**"number of the set bits in bits2 "** + x);  System.***out***.println(**"number of the set bits in bits1 "** + y);  } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  number of the set bits in bits2 12  number of the set bits in bits1 8 |

**BitSet(<int>) & set(<int val>) & void and(BitSet bitSet) & void andNot(BitSet bitSet) & void or(BitSet bitSet) & void xor(BitSet bitSet)**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);   *// AND bits* bits2.and(bits1);  System.***out***.println(**"\nbits2 AND bits1: "**);  System.***out***.println(bits2);   *// OR bits* bits2.or(bits1);  System.***out***.println(**"\nbits2 OR bits1: "**);  System.***out***.println(bits2);   *// XOR bits* bits2.xor(bits1);  System.***out***.println(**"\nbits2 XOR bits1: "**);  System.***out***.println(bits2);   *//void andNot()* bits2.andNot(bits1);  System.***out***.println(**"\nbits2 andNot bits1: "**);  System.***out***.println(bits2);  } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  bits2 AND bits1:  {2, 4, 6, 8, 12, 14}  bits2 OR bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  bits2 XOR bits1:  {}  bits2 andNot bits1:  {} |

**void clear( ) & void clear( <int index> ) void clear(int startIndex, int endIndex)**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);   *//clear* bits2.clear();  System.***out***.println(**"clearing all vals "** + bits2);  bits1.clear(14); *//actually put the value here, to remove* System.***out***.println(**"removed by index "** + bits1);  bits1.clear(2, 5);  System.***out***.println(**"removed by fromindex to toindex "** + bits1);  } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  clearing all vals {}  removed by index {0, 2, 4, 6, 8, 10, 12}  removed by index {0, 6, 8, 10, 12} |

**Object clone( )**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);   *//clone* Object bits3 = bits1.clone();  System.***out***.println(**"bits1 is cloned upon bits3 : "** + bits3);  } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  bits1 is cloned upon bits3 : {0, 2, 4, 6, 8, 10, 12, 14} |

**boolean equals(Object bitSet)**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);   *//clone* Object bits3 = bits1.clone();  System.***out***.println(**"bits1 is cloned upon bits3 : "** + bits3);   *//equals* **if** (bits1.equals(bits3))  System.***out***.println(**" bits1 & bits3 are Same"**);  **else** System.***out***.println(**"bits1 & bits3 are Different"**);  } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  bits1 is cloned upon bits3 : {0, 2, 4, 6, 8, 10, 12, 14}  bits1 & bits3 are Same |

**void flip(int index) & void flip(int startIndex, int endIndex)**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);   System.***out***.println();   bits1.flip(0, 14);  System.***out***.println(**"flipped case for bits1 "** + bits1);  } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  flipped case for bits1 {1, 3, 5, 7, 9, 11, 13, 14} |

**boolean intersects(BitSet bitSet)**

|  |  |
| --- | --- |
| **package** com.company; *//import javax.xml.bind.SchemaOutputResolver;* **import** java.util.\*;  **public class** Main {   **public static void** main(String[] args) {  BitSet bits1 = **new** BitSet(16);  BitSet bits2 = **new** BitSet(16);   *// set some bits* **for**(**int** i = 0; i < 16; i++) {  **if**((i % 2) == 0)  bits1.set(i);  **if**((i % 5) != 0)  bits2.set(i);  }   System.***out***.println(**"Initial pattern in bits1: "**);  System.***out***.println(bits1);  System.***out***.println(**"\nInitial pattern in bits2: "**);  System.***out***.println(bits2);   System.***out***.println();   **if** (bits1.intersects(bits2))  System.***out***.println(**"intersects operation : "** + bits1.intersects(bits2));   } } | Initial pattern in bits1:  {0, 2, 4, 6, 8, 10, 12, 14}  Initial pattern in bits2:  {1, 2, 3, 4, 6, 7, 8, 9, 11, 12, 13, 14}  intersects operation : true |