**Java Bitset类**

一个Bitset类创建一种特殊类型的数组来保存位值。BitSet中数组大小会随需要增加。这和位向量（vector of bits）比较类似。

这是一个传统的类，但它在Java 2中被完全重新设计。

BitSet定义了两个构造方法。

第一个构造方法创建一个默认的对象：

BitSet()

第二个方法允许用户指定初始大小。所有位初始化为0。

BitSet(int size)

BitSet中实现了Cloneable接口中定义的方法如下表所列：

|  |  |
| --- | --- |
| **序号** | **方法描述** |
| 1 | void and(BitSet bitSet) 对此目标位 set 和参数位 set 执行逻辑与操作。 |
| 2 | void andNot(BitSet bitSet) 清除此 BitSet 中所有的位，其相应的位在指定的 BitSet 中已设置。 |
| 3 | int cardinality( ) 返回此 BitSet 中设置为 true 的位数。 |
| 4 | void clear( ) 将此 BitSet 中的所有位设置为 false。 |
| 5 | void clear(int index) 将索引指定处的位设置为 false。 |
| 6 | void clear(int startIndex, int endIndex) 将指定的 fromIndex（包括）到指定的 toIndex（不包括）范围内的位设置为 false。 |
| 7 | Object clone( ) 复制此 BitSet，生成一个与之相等的新 BitSet。 |
| 8 | boolean equals(Object bitSet) 将此对象与指定的对象进行比较。 |
| 9 | void flip(int index) 将指定索引处的位设置为其当前值的补码。 |
| 10 | void flip(int startIndex, int endIndex) 将指定的 fromIndex（包括）到指定的 toIndex（不包括）范围内的每个位设置为其当前值的补码。 |
| 11 | boolean get(int index) 返回指定索引处的位值。 |
| 12 | BitSet get(int startIndex, int endIndex) 返回一个新的 BitSet，它由此 BitSet 中从 fromIndex（包括）到 toIndex（不包括）范围内的位组成。 |
| 13 | int hashCode( ) 返回此位 set 的哈希码值。 |
| 14 | boolean intersects(BitSet bitSet) 如果指定的 BitSet 中有设置为 true 的位，并且在此 BitSet 中也将其设置为 true，则返回 ture。 |
| 15 | boolean isEmpty( ) 如果此 BitSet 中没有包含任何设置为 true 的位，则返回 ture。 |
| 16 | int length( ) 返回此 BitSet 的"逻辑大小"：BitSet 中最高设置位的索引加 1。 |
| 17 | int nextClearBit(int startIndex) 返回第一个设置为 false 的位的索引，这发生在指定的起始索引或之后的索引上。 |
| 18 | int nextSetBit(int startIndex) 返回第一个设置为 true 的位的索引，这发生在指定的起始索引或之后的索引上。 |
| 19 | void or(BitSet bitSet) 对此位 set 和位 set 参数执行逻辑或操作。 |
| 20 | void set(int index) 将指定索引处的位设置为 true。 |
| 21 | void set(int index, boolean v)  将指定索引处的位设置为指定的值。 |
| 22 | void set(int startIndex, int endIndex) 将指定的 fromIndex（包括）到指定的 toIndex（不包括）范围内的位设置为 true。 |
| 23 | void set(int startIndex, int endIndex, boolean v) 将指定的 fromIndex（包括）到指定的 toIndex（不包括）范围内的位设置为指定的值。 |
| 24 | int size( ) 返回此 BitSet 表示位值时实际使用空间的位数。 |
| 25 | String toString( ) 返回此位 set 的字符串表示形式。 |
| 26 | void xor(BitSet bitSet) 对此位 set 和位 set 参数执行逻辑异或操作。 |

**实例**

下面的程序说明这个数据结构支持的几个方法：

import java.util.BitSet;

public class BitSetDemo {

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);

}

}

以上实例编译运行结果如下：

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:

{}