Lớp BitSet trong Java

Lớp BitSet trong Java tạo một kiểu mảng đặc biệt mà giữ các giá trị bit. Mảng BitSet này có thể tăng giảm kích cỡ nếu cần. Điều này làm nó tương tự như một vector của các bit.

Đây là một lớp legacy nhưng nó đã hoàn toàn được thiết kế lại trong Java 2, phiên bản 1.4.

Để hiểu sâu hơn các khái niệm được trình bày trong chương này, mời bạn tham khảo loạt bài: **Ví dụ về Cấu trúc dữ liệu (Data Structure) trong Java**.

BitSet định nghĩa hai constructor. Phiên bản đầu tiên tạo một đối tượng mặc định:

```
BitSet( )
```

Phiên bản thứ hai cho phép bạn xác định kích cỡ ban đầu của nó, ví dụ như số bit mà nó có thể giữ. Tất cả các bit được khởi tạo về 0.

```
BitSet(int size)
```

BitSet triển khai Cloneable Interface và định nghĩa các phương thức được liệt kê sau:

STT	Phương thức và Miêu tả
1	**void and(BitSet bitSet)** AND nội dung của đối tượng BitSet đang triệu hồi với nội dung đã được xác định bởi bitSet. Kết quả được đặt trong đối tượng đang triệu hồi
2	**void andNot(BitSet bitSet)** Với mỗi bit trong bitSet, bit tương ứng trong BitSet đang triệu hồi bị xóa
3	**int cardinality( )** Trả về số bit được thiết lập trong đối tượng đang triệu hồi
4	**void clear( )**

	Thiết lập tất cả bit về 0
5	**void clear(int index)** Thiết lập tất cả bit được xác định bởi index về 0
6	**void clear(int startIndex, int endIndex)** Thiết lập tất cả bit được xác định từ startIndex tới endIndex -1 về 0
7	**Object clone( )** Sao chép đối tượng BitSet đang triệu hồi
8	**boolean equals(Object bitSet)** Trả về true nếu BitSet đang triệu hồi là tương đương với thiết lập bit trong bitSet. Nếu không là false
9	**void flip(int index)** Đảo ngược các bit được xác định bởi index (
10	**void flip(int startIndex, int endIndex)** Đảo ngược các bit được xác định từ startIndex tới endIndex - 1
11	**boolean get(int index)** Trả về trạng thái hiện tại của bit tại index đã cho
12	**BitSet get(int startIndex, int endIndex)** Trả về một BitSet mà chứa các bit từ startIndex tới endIndex-1. Đối tượng đang triệu hồi không bị thay đổi
13	**int hashCode( )**

	Trả về hash code cho đối tượng đang triệu hồi
14	**boolean intersects(BitSet bitSet)** Trả về true nếu ít nhất một cặp bit tương ứng trong đối tượng đang gọi và bitSet là 1
15	**boolean isEmpty( )** Trả về true nếu tất cả bit trong đối tượng đang gọi là 0
16	**int length( )** Trả về số bit cần thiết để giữ nội dung của BitSet đang gọi. Giá trị này được xác định bởi vị trí của bit cuối cùng
17	**int nextClearBit(int startIndex)** Trả về chỉ mục của bit bị xóa kế tiếp (mà là zero bit kế tiếp), bắt đầu từ chỉ mục được xác định bởi startIndex
18	**int nextSetBit(int startIndex)** Trả về chỉ mục của set bit kế tiếp (mà là 1 bit kế tiếp), bắt đầu từ chỉ mục được xác định bởi startIndex. Nếu không bit nào được thiết lập, thì trả về - 1
19	**void or(BitSet bitSet)** OR nội dung của đối tượng BitSet đang gọi với nội dung được thiết lập bởi bitSet. Kết quả được đặt trong đối tượng đang triệu hồi
20	**void set(int index)** Thiết lập bit được xác định bởi index
21	**void set(int index, boolean v)** Thiết lập bit được xác định bởi index tới giá trị được truyền trong v: nếu là true thì thiết

	lập bit, là false thì xóa bit đó
22	**void set(int startIndex, int endIndex)** Thiết lập bit từ startIndex tới endIndex - 1
23	**void set(int startIndex, int endIndex, boolean v)** Thiết lập bit, từ startIndex tới endIndex-1, tới giá trị được truyền trong v: nếu là true thì thiết lập bit, là false thì xóa bit đó
24	**int size( )** Trả về số bit trong đối tượng BitSet đang gọi
25	**String toString( )** Trả về chuỗi tương đương của đối tượng BitSet đang gọi
26	**void xor(BitSet bitSet)** XOR nội dung của đối tượng BitSet đang gọi với nội dung được xác định bởi bitSet. Kết quả được đặt trong đối tượng đang gọi

Ví dụ

Chương trình sau là ví dụ minh họa các phương thức được hỗ trợ bởi cấu trúc dữ liệu này:

```
import java.util.BitSet;

public class BitSetDemo {

  public static void main(String args[]) {
     BitSet bits1 = new BitSet(16);
     BitSet bits2 = new BitSet(16);

     // thiet la mot so bit
```

```
     for(int i=0; i<16; i++) {
        if((i%2) == 0) bits1.set(i);
        if((i%5) != 0) bits2.set(i);
     }
     System.out.println("Pattern ban dau trong bits1: ");
     System.out.println(bits1);
     System.out.println("\nPattern ban dau trong 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);
  }
}
```

Nó sẽ cho kết quả sau:

```
Pattern ban dau trong bits1:
{0, 2, 4, 6, 8, 10, 12, 14}

Pattern ban dau trong 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:
{}
```