

# Chuyên đề mở rộng

## Java Data Structures

### Mục tiêu

- ✓ Sử dụng được một số lớp cấu trúc dữ liệu.

### Tóm tắt

- **Enumration interface:** bản chất không phải là một cấu trúc dữ liệu nhưng nó quan trọng trong bối cảnh các cấu trúc dữ liệu khác. Interface này cung cấp phương tiện để lấy phần tử kế tiếp từ cấu trúc dữ liệu.
- **Bitset class:** nó tạo ra một mảng các bit biểu diễn bằng giá trị Boolean các phần tử mảng có thể cài đặt hoặc xóa riêng lẻ, nó cung cấp các phương thức thao tác với bit thông thường bằng các toán tử bitwise (AND, OR, XOR). Ví dụ có 2 BitSet, lớp sẽ cung cấp phương thức để thực hiện thao tác bitwise logic trên 2 đối tượng này.
- **Stack class:** lưu trữ ngăn xếp và lấy phần tử theo nguyên tắc LIFO
- **Dictionary class:** lớp gốc này đã lỗi thời và thay vào đó Java giới thiệu HashMap, Hashtable như giải pháp thay thế. Mục đích để lưu trữ dữ liệu dưới dạng cặp: KHÓA / GIÁ TRỊ

### Bài thực hành số 1:

**Yêu cầu:** viết hàm demo lớp cấu trúc Enumeration với Vector.

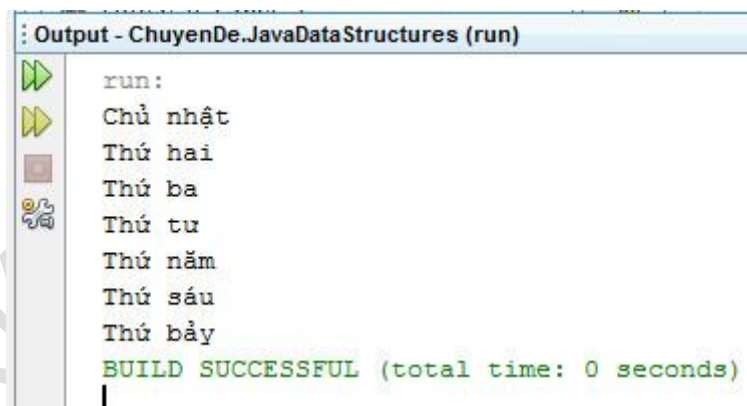
### Code tham khảo:

```
import java.util.Enumeration;
import java.util.Vector;

/**
 *
 * @author minhvufo
 */
public class EnumerationTester {

    /**
     * @param args the command line arguments
     */
}
```

```
*/  
public static void main(String[] args) {  
    Enumeration days;  
    Vector dayNames = new Vector(); // Collection lưu ngày trong tuần  
  
    // Thêm dữ liệu  
    dayNames.add("Chủ nhật");  
    dayNames.add("Thứ hai");  
    dayNames.add("Thứ ba");  
    dayNames.add("Thứ tư");  
    dayNames.add("Thứ năm");  
    dayNames.add("Thứ sáu");  
    dayNames.add("Thứ bảy");  
  
    // Lấy đối tượng Enumeration từ Vector trên  
    days = dayNames.elements();  
  
    // Truy cập tuần tự bằng gọi hàm nextElement  
    while (days.hasMoreElements()) {  
        System.out.println(days.nextElement());  
    }  
}
```



## Bài thực hành số 2:

**Yêu cầu:** tạo lớp BitSetDemo trong đó khởi tạo và gán dữ liệu cho 2 BitSet, gọi thực hiện các toán tử trên 2 đối tượng này.

### Code tham khảo:

```
package chuyende.javadatastructures;
```

```
import java.util.BitSet;
```

```
/**
```

```

*
* @author minhvuvc
*/
public class BitSetDemo {

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        BitSet bits1 = new BitSet(16);
        BitSet bits2 = new BitSet(16);

        // Cài đặt dữ liệu từng phần tử cho BitSet
        for (int i = 0; i < 16; i++) {
            if ((i % 2) == 0) {
                bits1.set(i);
            }
            if ((i % 5) != 0) {
                bits2.set(i);
            }
        }

        System.out.println("Giá trị trong bits1: ");
        System.out.println(bits1);
        System.out.println("\nGiá trị trong bits2: ");
        System.out.println(bits2);

        // AND bits: thực hiện toán tử AND
        bits2.and(bits1);
        System.out.println("\nbits2 AND bits1: ");
        System.out.println(bits2);

        // OR bits: thực hiện toán tử OR
        bits2.or(bits1);
        System.out.println("\nbits2 OR bits1: ");
        System.out.println(bits2);

        // XOR bits: : thực hiện toán tử XOR
        bits2.xor(bits1);
        System.out.println("\nbits2 XOR bits1: ");
        System.out.println(bits2);
    }
}

```

```
Output - ChuyenDe.JavaDataStructures (run)

run:
Giá trị trong bits1:
{0, 2, 4, 6, 8, 10, 12, 14}

Giá trị 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:
{}

BUILD SUCCESSFUL (total time: 0 seconds)
```

### Bài thực hành số 3:

**Yêu cầu:** tạo lớp StackDemo trong đó khởi tạo và gán dữ liệu cho đối tượng Stack. Sử dụng phương thức cung cấp sẵn của lớp để lấy dữ liệu.

#### Code tham khảo:

`package` chuyende.javadatastructures;

`import` java.util.Stack;

```
/**
 *
 * @author minhvufc
 */
public class StackDemo {

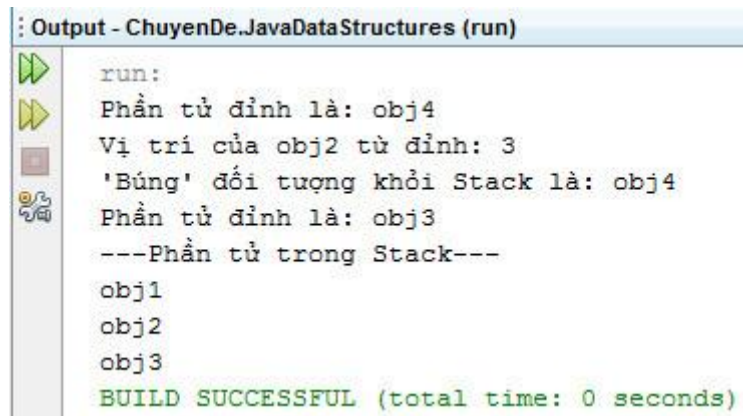
    /**
     * Khởi tạo đối tượng Stack có chèn phần tử
     *
     * @return
     */
    private static Stack getInitializedStack() {
        Stack stack = new Stack();
        stack.push("obj1");
        stack.push("obj2");
        stack.push("obj3");
        stack.push("obj4");
        return stack;
    }
}
```

```

    }

    /**
     * @param args the command line arguments
     */
    public static void main(String[] args) {
        Stack initializedStack = StackDemo.getInitializedStack();
        System.out.println("Phần tử đỉnh là: " + initializedStack.peek());
        System.out.println("Vị trí của obj2 từ đỉnh: " +
        initializedStack.search("obj2"));
        System.out.println("'Búng' đối tượng khỏi Stack là: " + initializedStack.pop());
        System.out.println("Phần tử đỉnh là: " + initializedStack.peek());
        System.out.println("---Phần tử trong Stack--- ");
        for (Object obj : initializedStack) {
            System.out.println(obj);
        }
    }
}

```



```

: Output - ChuyenDe.JavaDataStructures (run)

run:
Phần tử đỉnh là: obj4
Vị trí của obj2 từ đỉnh: 3
'Búng' đối tượng khỏi Stack là: obj4
Phần tử đỉnh là: obj3
---Phần tử trong Stack---
obj1
obj2
obj3
BUILD SUCCESSFUL (total time: 0 seconds)

```