JAVA - THE STACK CLASS

http://www.tutorialspoint.com/java/java_stack_class.htm

Stack is a subclass of Vector that implements a standard last-in, first-out stack.

Stack only defines the default constructor, which creates an empty stack. Stack includes all the methods defined by Vector, and adds several of its own.

```
Stack( )
```

Apart from the methods inherited from its parent class Vector, Stack defines following methods:

SN	Methods with Description
1	**boolean empty()** Tests if this stack is empty. Returns true if the stack is empty, and returns false if the stack contains elements.
2	**Object peek( )** Returns the element on the top of the stack, but does not remove it.
3	**Object pop( )** Returns the element on the top of the stack, removing it in the process.
4	**Object push(Object element)** Pushes element onto the stack. element is also returned.
5	**int search(Object element)** Searches for element in the stack. If found, its offset from the top of the stack is returned. Otherwise, .1 is returned.

Example:

The following program illustrates several of the methods supported by this collection:

```
import java.util.*;

public class StackDemo {

   static void showpush(Stack st, int a) {
      st.push(new Integer(a));
      System.out.println("push(" + a + ")");
      System.out.println("stack: " + st);
   }

   static void showpop(Stack st) {
      System.out.print("pop -> ");
      Integer a = (Integer) st.pop();
      System.out.println(a);
      System.out.println("stack: " + st);
   }

   public static void main(String args[]) {
      Stack st = new Stack();
      System.out.println("stack: " + st);
      showpush(st, 42);
      showpush(st, 66);
      showpush(st, 99);
      showpop(st);
      showpop(st);
      showpop(st);
      try {
         showpop(st);
      } catch (EmptyStackException e) {
         System.out.println("empty stack");
      }
   }
}
```

This would produce the following result:

```
stack: [ ]
push(42)
stack: [42]
push(66)
stack: [42, 66]
push(99)
stack: [42, 66, 99]
pop -> 99
stack: [42, 66]
pop -> 66
stack: [42]
pop -> 42
stack: [ ]
pop -> empty stack
```