```
 * ArrayStack.java
import java.util.EmptyStackException;

public class ArrayStack<E> implements Stack<E> {

    protected int maxSize;                          // actual capacity
    protected static final int MAX_SIZE = 1000;     // default capacity
    protected E stack[];                            // array that contains
the stack elements
    protected int top = -1;                         // index for the top
element of the stack

    /**
     * 0-param constructor
     */
    public ArrayStack () {
        this(MAX_SIZE);
    }

    /**
     * 1-param constructor
     * @param maxSize the maximum size of the array
     */
    public ArrayStack (int maxSize) {
        this.maxSize = maxSize;
        stack = (E[]) new Object[maxSize];
    }

    /**
     * Returns the number of elements in the stack
     * @return the number of elements in the stack
     */
    public int size ()  {
        return (top+1);
    }

    /**
     * Tests for emptiness
     * @return true if the stack is empty, false otherwise
     */
    public boolean empty () {
        return(top < 0);
    }

    public void push (E element) throws FullStackException {
        if (top == maxSize)
             throw new FullStackException("Cannot push, the stack is
full");
        stack[++top] = element;
    }

    /**
     * Peeks at (returns) the top element of the stack
     * without removing it.
     * @return the top element in the stack
     * @throws EmptyStackException if the stack is empty
     */
    public E peek () throws EmptyStackException {
        if (empty())
            throw new EmptyStackException();
        return stack[top];
    }

    /**
     * Returns and removes the top element of the stack.
     * @return the top element in the stack
     * @throws EmptyStackException if the stack is empty
     */
    public E pop () throws EmptyStackException {
        if (empty())
            throw new EmptyStackException();
        E element = stack[top];
        stack[top--] = null;
        return element;
    }

    /**
     * Swaps the top two elements of the stack.
     * @throws EmptyStackException if the stack is empty or contains
one element
     */
    public void swap () throws EmptyStackException {
        if (empty() || size() == 1) {
             throw new EmptyStackException();
         }
        E temp = stack[top];
        stack[top] = stack[top-1];
        stack[top-1] = temp;
    }

    /**
     * returns a String representing the state of this Stack
     * @return a String representing the state of this stack
     */
    public String toString () {
        StringBuilder returnString = new StringBuilder("[");
        if (size() == 1)
             return (returnString.append(" " + stack[0] +
" ]").toString());
        if (size() > 1)
            returnString.append(" " + stack[top]);
            for (int i = size()-2; i > 0; i--) {
                returnString.append(", " + stack[i]);
            }
            returnString.append(", " + stack[0] + " ]");
        return returnString.toString();
    }
}
```