```java
/**
 * Stack.java
 * Interface for the stack ADT to be implemented using both arrays and linked-
     lists.
 * Based on code by Michael Goodrich and Roberto Tamassia
 * presented in "Data Structures & Algorithms in Java".
 *
 * @author Hawk Weisman
 * @see EmptyStackException
 * @see NodeStack
 * @see ArrayStack
 *
 * PLEDGE:
 */

import java.util.EmptyStackException;

public interface Stack<E> {

    /**
     * Returns the number of elements in the stack
     * @return the number of elements in the stack
     */
    public int size ();

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

    /**
     * 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;

    /**
     * Pushes an element to the stack.
     * @param Element to be pushed
     * @throws FulLStackException if the stack is an ArrayStack that is full
     */
    public void push (E element)
        throws FullStackException;

    /**
     * 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;
```

```
    /**
     * Swaps the top two elements of the stack.
     * @throws EmptyStackException if the stack is empty or contains one element
     */
    public void swap ()
        throws EmptyStackException;

    /**
     * returns a String representing the state of this Stack
     * @return a String representing the state of this stack
     */
    public String toString ();

}
```