```java
/**
 * NodeStack.java
 * A linked-list based implementation of the Stack ADT.
 *  Based on code by Michael Goodrich and Roberto Tamassia
 * presented in "Data Structures & Algorithms in Java".
 *
 * @author Hawk Weisman
 * @see Stack
 * @see Node
 *
 * PLEDGE:
 */
import java.util.EmptyStackException;

public class NodeStack<E> implements Stack<E> {
    protected Node<E> topNode;  // the head node of the linked list
    protected int size;         // number of elements in the stack

    /**
     * 0-param constructor
     */
    public NodeStack () {
        topNode = null;
        size = 0;
    }

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

    /**
     * Tests for emptiness
     * @return true if the stack is empty, false otherwise
     */
    public boolean empty () {
        if (topNode == null) {
            return true;
        } else {
            return false;
        }
    }

    /**
     * Pushes an element to the stack.
     * @param Element to be pushed
     */
    public void push (E element) {
        Node<E> pushedElement = new Node<E> (element, topNode); // create a new
            node and link it in
        topNode = pushedElement;
        size++;
    }
```

```
/**
 * 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();
    } else {
        return topNode.getElement();
    }
}

/**
 * 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();
    } else {
        E temp = topNode.getElement();
        topNode = topNode.getNext();        //unlink the old top
        size--;
        return temp;
    }
}

/**
 * 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();
    } else {
        E tempA = pop();
        E tempB = pop();
        push(tempA);
        push(tempB);

    }
}

/**
 * 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("[ ");
    Node<E> currentNode = topNode;
    E currentElement;
    for (int i = 0; i < size-1; i++) {
```

```
            returnString.append(currentNode.getElement().toString()  + ", ");
            currentNode = currentNode.getNext();
        }
        returnString.append(topNode.getElement().toString() + " ]");
        return returnString.toString();
    }
}
```