```java
/**
  *  Program 2
  *  Stack is an implementation of LinearList
  *  CS310-01
  *  3/13/2019
  *  @author  Karl Parks cssc1506
  */

package data_structures;

import java.util.Iterator;

public class Stack<E extends Comparable<E>> implements Iterable<E>{
  private LinearList<E> list;

    public Stack() {
      list = new LinearList<E>();
    }

  /* inserts the object obj into the stack
   */
  public void push(E obj) {
    list.addLast(obj);
    return;
  }

   /* pops and returns the element on the top of the stack
   */
  public E pop() {
    list.removeLast();
    return null;
  }

   /* returns the number of elements currently in the stack
   */
  public int size() {
    return list.size();
  }

   /* return true if the stack is empty, otherwise false
   */
  public boolean isEmpty() {
    return list.isEmpty();
  }
```

```java

   /* returns but does not remove the element on the top of the stack
   */
  public E peek() {
    return list.peekLast();
  }

   /* returns true if the object obj is in the stack,
   * otherwise false
   */
  public boolean contains(E obj) {
    return list.contains(obj);
  }

   /* returns the stack to an empty state
   */
  public void makeEmpty() {
    list.clear();
  }

   /* removes the Object obj if it is in the stack and
   * returns true, otherwise returns false.
   */
  public boolean remove(E obj) {
    boolean tmp = list.contains(obj);
    list.remove(obj);
    return tmp;
  }

   /* returns a iterator of the elements in the stack. The elements
   * must be in the same sequence as pop() would return them.
   */
  @Override
  public Iterator<E> iterator() {
    return list.iterator();
  }
}
```