Stack Interface* and Implementation

```
public interface Stack
{
    // postcondition: returns true if stack is empty, false otherwise
  boolean isEmpty();

    // precondition:  stack is [e1, e2, ..., en] with n >= 0
    // postcondition: stack is [e1, e2, ..., en, x]
  void push(Object x);

    // precondition:  stack is [e1, e2, ..., en] with n >= 1
    // postcondition: stack is [e1, e2, ..., e(n-1)]; returns en
    //                 throws an unchecked exception if the stack is empty
  Object pop();

    // precondition:  stack is [e1, e2, ..., en] with n >= 1
    // postcondition: returns en
    //                 throws an unchecked exception if the stack is empty
  Object peekTop();
}


public class ArrayStack implements Stack
{
  private java.util.ArrayList array;

  public ArrayStack()
  {
    array = new java.util.ArrayList();
  }

  public void push(Object obj)
  {
    array.add(obj);
  }

  public Object pop()
  {
    return array.remove(array.size() - 1);
  }

  public Object peekTop()
  {
    return array.get(array.size() - 1);
  }

  public boolean isEmpty()
  {
    return array.size() == 0;
  }
}
```

* Adapted from the College Board's *AP Computer Science AB: Implementation Classes and Interfaces*.

```
public class ListStack implements Stack
{
  private java.util.LinkedList list;

  public ListStack()
  {
    list = new java.util.LinkedList();
  }

  public boolean isEmpty()
  {
    return list.isEmpty();
  }

  public void push(Object obj)
  {
    list.addFirst(obj);
  }

  public Object pop()
  {
    return list.removeFirst();
  }

  public Object peekTop()
  {
    return list.getFirst();
  }
}
```