```
public class ListIndexOutOfBoundsException
             extends IndexOutOfBoundsException {
  public ListIndexOutOfBoundsException(String s) {
    super(s);
  }  // end constructor
}  // end ListIndexOutOfBoundsException
```

Also, the exception *ListException* is needed when the array storing the list becomes full. Here is the exception *ListException*:

```
public class ListException extends RuntimeException {
  public ListException(String s) {
    super(s);
  }  // end constructor
}  // end ListException
```

```
// ************************************************************
public interface ListInterface {
  public boolean isEmpty();
  public int size();
  public void add(int index, Object item)
                  throws ListIndexOutOfBoundsException,
                         ListException;
  public Object get(int index)
                    throws ListIndexOutOfBoundsException;
  public void remove(int index)
                     throws ListIndexOutOfBoundsException;
  public void removeAll();
}  // end ListInterface
```

Implementation file

```
// ********************************************************
// Array-based implementation of the ADT list.
// ********************************************************
public class ListArrayBased implements ListInterface {
  private static final int MAX_LIST = 50;
  private Object items[];  // an array of list items
  private int numItems;  // number of items in list

  public ListArrayBased() {
    items = new Object[MAX_LIST];
    numItems = 0;
  }  // end default constructor

  public boolean isEmpty() {
    return (numItems == 0);
  } // end isEmpty

  public int size() {
    return numItems;
  }  // end size

  public void removeAll() {
    // Creates a new array; marks old array for
    // garbage collection.
    items = new Object[MAX_LIST];
    numItems = 0;
  } // end removeAll

  public void add(int index, Object item)
                  throws ListIndexOutOfBoundsException {
    if (numItems > MAX_LIST) {
      throw new ListException("ListException on add");
    }  // end if
    if (index >= 0 && index <= numItems) {
      // make room for new element by shifting all items at
      // positions >= index toward the end of the
      // list (no shift if index == numItems+1)
      for (int pos = numItems; pos >= index; pos--) {
        items[pos+1] = items[pos];
      } // end for
      // insert new item
      items[index] = item;
      numItems++;
    }
    else {  // index out of range
      throw new ListIndexOutOfBoundsException(
       "ListIndexOutOfBoundsException on add");
```

```
    } // end if
  } //end add

  public Object get(int index)
                    throws ListIndexOutOfBoundsException {
    if (index >= 0 && index < numItems) {
      return items[index];
    }
    else  {  // index out of range
      throw new ListIndexOutOfBoundsException(
        "ListIndexOutOfBoundsException on get");
    }  // end if
  } // end get

  public void remove(int index)
                     throws ListIndexOutOfBoundsException {
    if (index >= 0 && index < numItems) {
      // delete item by shifting all items at
      // positions > index toward the beginning of the list
      // (no shift if index == size)
      for (int pos = index+1; pos <= size(); pos++) {
        items[pos-1] = items[pos];
      }  // end for
      numItems--;
    }
    else {  // index out of range
        throw new ListIndexOutOfBoundsException(
        "ListIndexOutOfBoundsException on remove");
    }  // end if
  } // end remove

}  // end ListArrayBased
```

The following program segment demonstrates the use of *ListArrayBased*:

```
static public void main(String args[]) {
  . . .
  ListArrayBased aList = new ListArrayBased();
  String dataItem;

  aList.add(0, "Cathryn");
  . . .
  dataItem = (String)aList.get(0);
  . . .
```