JAVA NOTES

DATA STRUCTURES AND ALGORITHMS

Terry Marris July 2001

13 LIST INTERFACE

13.1 LEARNING OUTCOMES

By the end of this lesson the student should be able to

- describe the characteristic features of a list
- describe a list interface
- explain list methods

13.2 INTRODUCTION

In the last few lessons we have looked at the linear data structures stacks and queues. Now we look at a third linear data structure, the list.

We have all maintained lists such as a list of things to do, a list of items to buy and a list of subjects to revise. Sometimes the order of the items in the list does not matter; we place the next new item onto the end of the list. Sometimes order matters e.g. a list of jobs to do in order of importance; we insert the next new item in its right place in the list.

A list is a linear data structure. Items may be added to and removed from the list in any order and in any position.

13.3 THE LIST INTERFACE

The list interface is shown below. The list is indexed; the objects are indexed in the usual perverse programmers' way of counting from zero upwards.

```
/* List.java  Terry Marris  30 July 2001  */

public interface List {
  public int size();
  /* returns the number of elements in this list.  */


  public String add(Object obj);
  /*  adds the given object to the end of this list and
      returns success.  */


  public String remove(Object obj);
  /* removes the given object from this list and
     returns success
     if the given object is in this list, otherwise
     returns failure.
     Requires equals to be defined for the given object.
  */


  public Object get(int index);
  /* returns the object at the given indexed node.
     index must be within 0..size()-1.
  */


  public String add(int index, Object obj);
  /* inserts the given object at the given index position
     in this list and returns success.
     requires index to be in the range 0..size().
  */


  public String remove(int index);
  /* removes the node at the given index and returns success.
     requires index to be in the range 0..size()-1
  */


  public int indexOf(Object obj);
  /* returns the index of the given object, or -1 if the given
     object is not in this list.
  */
}
```

13.4 REVIEW

13.5 FURTHER READING

In the next lesson we look at an implementation that uses a linked list of nodes.

13.6 EXERCISES

1 Explain why index is restricted to 0..size() in add(int, Object) but restricted to 0..size()-1 in remove(int). Draw a diagram (or two) to illustrate your answer.