Abstract Data Types: Lists 1

ADTs
The List ADT
A List interface and its implementation: Array List
Abstract Data Types (ADTs)

Abstract data types describe the behaviour (semantics) of a data type without specifying its implementation. An ADT is thus abstract, not concrete.

A container is a very general ADT, serving as a holder of objects. A list is an example of a specific container ADT.

An ADT is described in terms of the semantics of the operations that may be performed over it.
The List ADT

The list ADT is a container known mathematically as a finite sequence of elements. A list has these fundamental properties:
• duplicates are allowed
• order is preserved
A list may support operations such as these:
• create: construct an empty list
• add: add an element to the list
• is empty: test whether the list is empty
Our List Interface

We will explore lists using a simple interface:

```java
public interface List<T> {
    void add(T value);
    T get(int index);
    int size();
    T remove(int index);
    void reverse();
}
```
Abstract Data Types: Lists

```java
void add(T value);

T get(int index);

int size();

T remove(int index);

void reverse();

String toString();
```
Abstract Data Types: Lists

List Implementation

• Arrays
  – Fast lookup of any element
  – A little messy to grow and contract

• Linked list
  – Logical fit to a list, easy to grow, contract
  – Need to traverse list to find arbitrary element