

Abstract Data Types

Abstract Data Types in Java

List is an ADT. A specific instance of a list, e.g. a List of integers, would be specified in Java by `List<int>`, a List of Strings as `List<String>` etc.

There are different implementations of the List ADT in the Java library, for example an Array based List (`ArrayList<int>`, `ArrayList<String>`, ...) and a Linked List (`LinkedList<int>`, `LinkedList<String>`, ...)

In Java we can declare and allocate a List, specifying which implementation we want, with the code:

```
1 List<int> myArrayList = new ArrayList<int>();  
2 List<int> myLinkedList = new LinkedList<int>();
```

From this point on, you can use any of the predefined List methods on the `myArrayList` or `myLinkedList` variables

Abstract Data Types Revisited

Recall that An *abstract data type* is

- a type
- with associated operations
- whose representation is hidden to the user

While a *List of integers* contains the type *integer*, the type of *List of integers* is not *integer*. It is a more complex “*container type*”. This is usually specified constructively: that is, we identify every possible value of type *List of integers* by specifying how to create each one. We do this by providing a list of constructor operations that create an empty *List of integers* and construct new values of type *List of integers* out of old ones

We also need to specify all other operations that any user of our ADT can depend on

List Abstract Data Type

Here is a possible list of operations for a List ADT (many variations are possible)¹

- Constructors:
 - `EmptyList` : returns an empty List
 - `MakeList(element, list)` , adds an element at the front of a list.
- Accessors
 - `first(list)` : returns the first element of the list²
 - `rest(list)` : returns the list excluding the first element²
 - `isEmpty(list)` : reports whether the list is empty

From these, all other operations (e.g. find the n^{th} element of the list, append one list onto another) can be implemented without requiring any other access to the List implementation details.

¹Read chapters 1 and 2 of the module handouts

²Triggers error if the list is empty

List Operations: last element

in Pseudocode:

```
1 last(lst) {  
2     if ( isEmpty(lst) )  
3         error(" Error: _empty_list_in_last")  
4     elseif ( isEmpty(rest(lst) )  
5         return first(lst)  
6     else  
7         return last(rest(lst))
```

List Operations: getElementByIndex

in Pseudocode:

```
1 getElementByIndex(index, lst) {  
2   if ( index < 0 or isEmpty(lst) )  
3     error(" Error: _index_out_of_range")  
4   elseif ( index == 0 )  
5     return first(lst)  
6   else  
7     return getElementByIndex(index - 1, rest(lst))
```

List Operations: append

in Pseudocode:

```
1 append(lst1 , lst2) {  
2   if ( isEmpty(lst1) )  
3     return lst2  
4   else  
5     return MakeList( first(lst1),  
6                       append(rest(lst1), lst2) )
```