

# CHAPTER 4 : ABSTRACT DATA TYPES (ADTs)

Informally an Abstract Data Type is a collection of data together with a set of well defined operations on that data. More precisely an ADT consists of two things:

- (1) A set  $S$  of 'mathematical structures' called states or values, this is the data
- (2) An associated set of operations which can be applied to the states in  $S$ .

By contrast a Data Structure is a construct within a programming language (such as an array or primitive data type) which stores a collection of data.

Data structures are the raw materials out of which an ADT is built.

## Ex. List of Integral ADT

Here  $S$  is the set of all finite sequences of integers of a fixed maximum length. (MAX\_LENGTH)

$$S = \{ \dots, (123), (7513), \dots \}$$

(IN GENERAL WE COULD CONSIDER LISTS OF OTHER TYPES OF . SUCH AS STRINGS, OR EVEN NON-SPECIFIC DATA OBJECTS.)

THE LIST OPERATIONS ARE SPECIFIED AS

- CREATE AN EMPTY LIST: `createList()`
- DETERMINE WHETHER A LIST IS EMPTY: `isEmpty()`
- DETERMINE LENGTH OF A LIST: `size()`
- ADD AN ITEM TO A GIVEN POSITION IN A LIST: `add(index, item)`
- DELETE AN ITEM FROM A GIVEN POSITION IN A LIST: `remove(index)`
- RESET A LIST TO THE EMPTY STATE: `removeAll()`
- GET THE ITEM AT A GIVEN POSITION IN THE LIST: `get(index)`

IT IS UNDERSTOOD THAT LIST INDICES ARE IN THE RANGE

$$1 \leq \text{index} \leq \text{size}() \leq \text{MAX\_LENGTH}$$

WHEN AN ITEM IS ADDED AT A GIVEN POSITION ALL THE ITEMS TO ITS RIGHT ARE SHIFTED RIGHT, i.e. THEIR INDICES INCREASE BY 1.

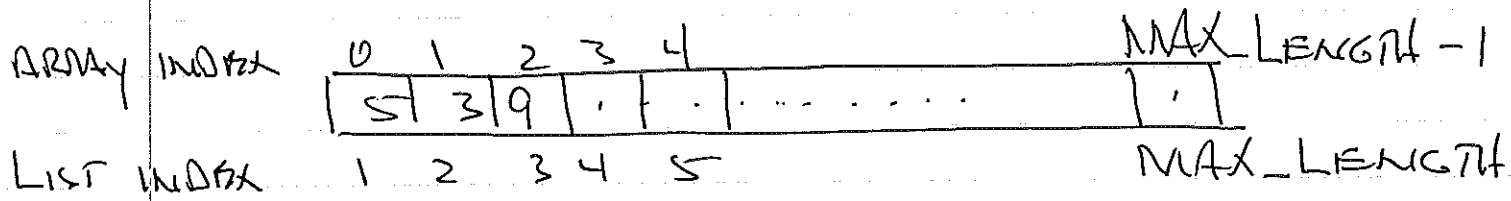
likewise when an item is deleted from a given position, all items to its right are shifted left, i.e. indices are decremented.

For instance, consider the following sequence of operations

<u>OPERATION</u>	<u>STATE</u>	<u>RETURN VAL</u>
createList()	()	
isEmpty()	()	true
add(1, 5)	(5)	
add(2, 3)	(5 3)	
add(3, 9)	(5 3 9)	
add(2, 7)	(5 7 3 9)	
size()	(5 7 3 9)	4
get(2)	(5 7 3 9)	7
remove(3)	(5 7 9)	
isEmpty()	(5 7 9)	false
removeAll()	()	

operations which alter the state of an ADT are sometimes called MANIPULATION PROCEDURES (OR MUTATORS). Those which only report on some aspect of the state of an ADT are called ACCESS FUNCTIONS.

THE INTEGER LIST ADT CAN BE IMPLEMENTED WITH AN ARRAY AS THE UNDERLYING DATA STRUCTURE.

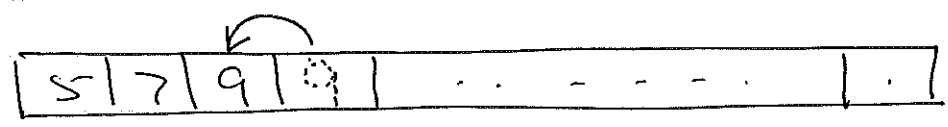


↓ add(2, 7)



7 ↗

↓ remove(3)



↑ size()

WE WILL SEE THAT THERE ARE OTHER WAYS TO IMPLEMENT THE INTEGER LIST ADT.

~~AN ADT IS AN ABSTRACT MATHEMATICAL ENTITY WHICH EXISTS APART FROM ANY PROGRAMMING LANGUAGE or DEVICE.~~

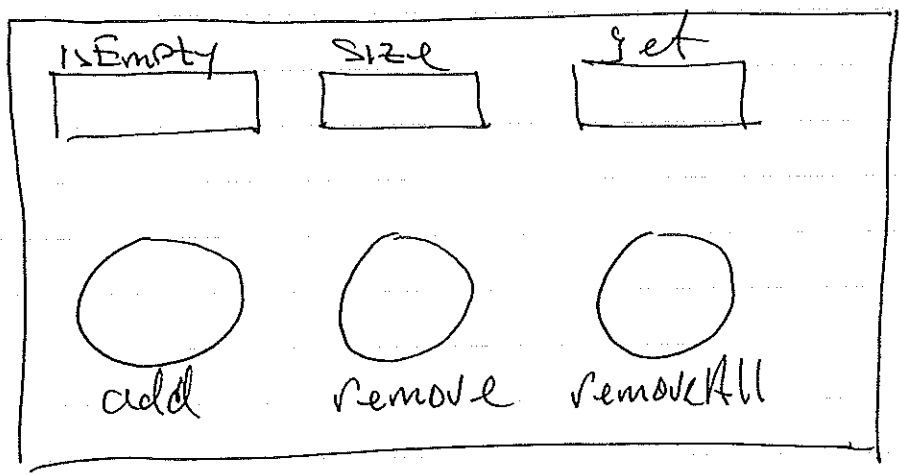
~~Shell Box Analogy~~

HOWEVER THE LIST ADT IS IMPLEMENTED, WE NOTE THAT CERTAIN OPERATIONS ARE UNDEFINED ON CERTAIN STATES.

FOR INSTANCE `add(.,.)` CANNOT BE CALLED IF `size() == MAX-LENGTH`, AND `remove()` IS UNDEFINED IF `!isEmpty()` IS TRUE.

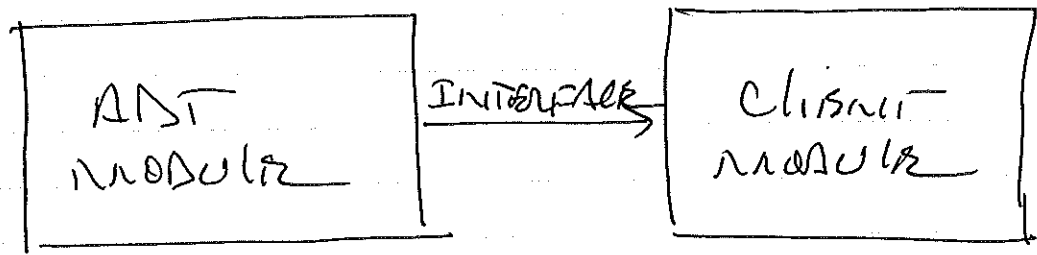
THEREFORE WE ESTABLISH PRE-CONDITIONS FOR EACH OF THE ADT OPERATIONS. WE ALSO ESTABLISH POST-CONDITIONS, I.E. CONDITIONS WHICH MUST BE TRUE AFTER AN OPERATION IS PERFORMED. E.G. `add(.,.)` HAS THE POST CONDITION `!isEmpty()`.

IT IS SOMETHING HELPFUL TO THINK OF AN ADT AS BEING A BLACK BOX EQUIPPED WITH A CONTROL PANEL WITH BUTTONS WHICH CAN BE PRESSED (MANIPULATION PROCEDURES) AND INDICATORS WHICH CAN BE READ (ACCESS FUNCTIONS).



THE USER OF THE ADT (CALLS ITS CLIENT) SHOULD NEVER BE ALLOWED TO SEE THE STRUCTURE INSIDE THE BLACK BOX. THIS IS THE IDEA OF INFORMATION HIDING: CLIENTS HAVE NO ACCESS TO AN ADT'S IMPLEMENTATION DETAILS.

A MODULE IS A PART OF A PROGRAM WHICH IS ISOLATED FROM THE REST OF A PROGRAM BY A WELL DEFINED INTERFACE. GENERALLY WE HAVE A SEPARATE MODULE FOR EACH ADT.



WE THINK OF MODULES AS PROVIDING SERVICES TO CLIENTS. A CLIENT CAN BE ANYTHING (PROGRAM, USER, ANOTHER MODULE) THESE SERVICES ARE SAID TO BE EXPORTS TO CLIENT.

INFORMATION HIDING MEANS THE CLIENT CAN ACCESS A MODULE'S SERVICES ONLY THROUGH THE INTERFACE.

IN JAVA AN ADT MODULE IS EMBODIED IN A CLASS.

- JAVA CLASS -

A JAVA CLASS CONSISTS OF

FIELD - MEMBER VARIABLES

METHOD - MEMBER FUNCTIONS

WHICH CONSTITUTE THE MATHEMATICAL STRUCTURE, AND THE OPERATIONS OF AN ADT, RESPECTIVELY.

FIELD & METHOD CAN BE DECLARED AS PRIVATE OR PUBLIC.

- PRIVATE IS USED TO ENFORCE INFORMATION HIDING & MODULARITY.

- PUBLIC MEMBER CONSTITUTE THE INTERFACE

NOTE: ALL MEMBERS ARE PRIVATE BY DEFAULT.

AN OBJECT IS AN INSTANCE OF A CLASS.

EVERY CLASS HAS AT LEAST ONE CONSTRUCTOR WHICH IS A METHOD WHICH ALLOCATES MEMORY FOR AN OBJECT, AND CAN INITIALIZE FIELDS TO CERTAIN VALUES.

A CONSTRUCTOR HAS THE SAME NAME AS THE CLASS, AND IS CALLED BY THE NEW OPERATOR.