CSE 101 1-4-22

ADT Handout:

Defn

an ADT (abstract data type) consists of

(1) a set S of "mathematical structures", called states.

(2) an associated set of operations on states. Roughly:

- manipulation procedures: change state.
- Access functions: return information about a state.

Ex Integer Queue

States

$Q = \{$ finite sequence of integers $\}$

Operations

- manipulation Procedures.

  Enqueue(): insert at back of queue

  Dequeue(): delete element at front.

- Access functions

  getFront(): return front element.

  getLength(): return length of sequence

  isEmpty(): return true iff seq. is empty.

History of states for any instance of Queue

Ex

op.	state	return
----	( )	
Enqueue(5)	(5)	-
Enqueue(1)	(5, 1)	-
Enqueue(7)	(5, 1, 7)	-
Dequeue( )	(1, 7)	-
Enqueue(3)	(1, 7, 3)	-
getLength( )	(1, 7, 3)	3
isEmpty( )	(1, 7, 3)	false
getFront( )	(1, 7, 3)	1
⋮	⋮	⋮

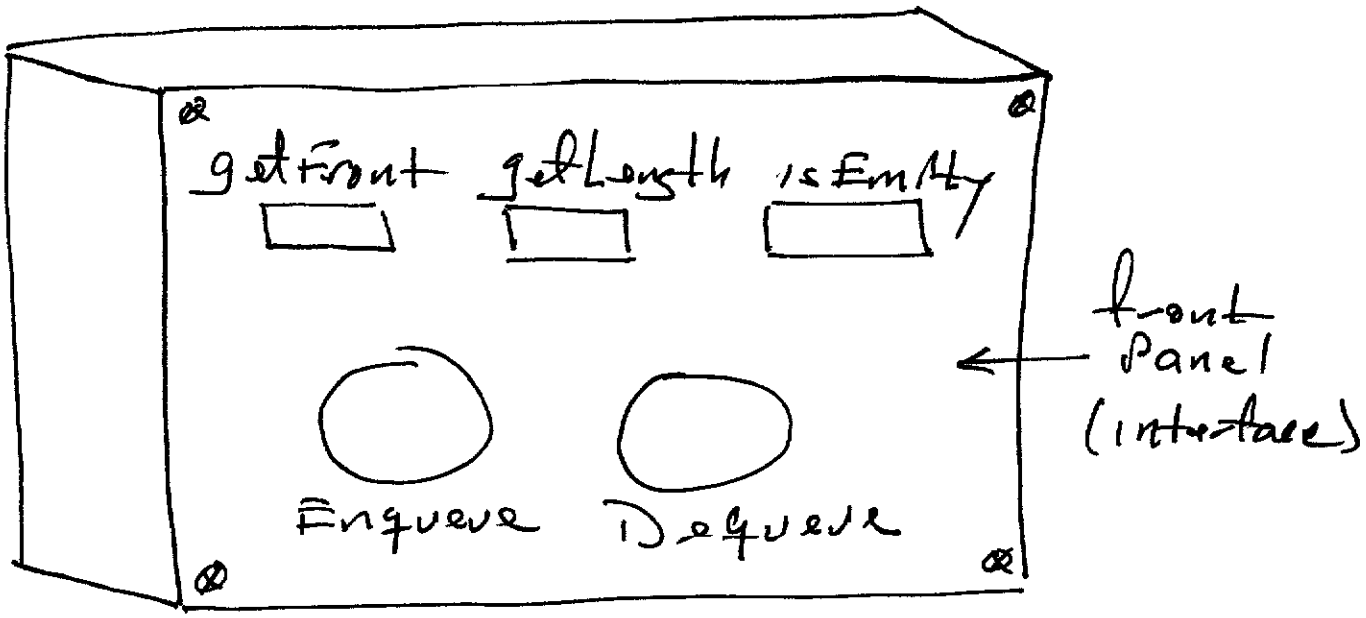
note: Dequeue() is undefined on the empty state ( ), so is getFront( ).

we define preconditions: conditions that must be true before an op. is executed.

Dequeue( ) has Pre: ! isEmpty( )

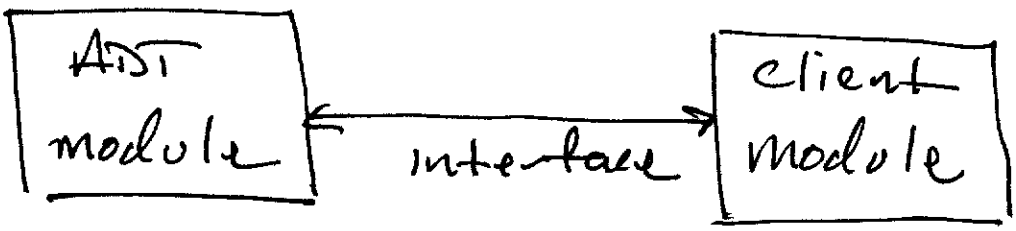
getFront( ) " " " "

Black Box Picture



Information hiding: The use only interacts through the interface.

A <u>module</u> is a part of a prog. that is isolated from the rest



Ex. Queue.c ← Queue ADT module

Queue.h ← interface

QueueTest.c ← client module

note:

typedef foo bar;

bar becomes an alias for foo

typedef struct QueueObj* Queue;

in Queue.h

In Queue.c

```
typedef struct NodeObj{
    int data;
    struct NodeObj* next;
}NodeObj;
```