12 An Abstract Data Type (ADT) is a data-storage class considered without reference to its implementation.

13 Stacks and queues are ADTs. They can be implemented using either arrays or linked lists.

14 In a doubly linked list, each link contains a reference to the previous link as well as the next link. A doubly linked list permits backward traversal and deletion from the end of the list.

Linked List Quiz

1- What is the advantage of a linked list over an array?

2- What is the advantage of a doubly ended linked list over a singly linked list?

3- What is the function of a Linkstack class?

4- Write Java method that will find the sum of all the integers in a linked list?

5- Write Java method that delete and returns the last Link of the linked list?

6- Suppose that the linked list pictured below is built from 3 Nodes as defined below.

```
class Node{
Node start;          // start -> 5 -> 7 -> 3
   int data;
   Node next;
  }
```

7- Write a method to find the maximum of the linked list?
8-Write a method to read the linked list in a reverse order?

Linked List Quiz

1. What is the advantage of a linked list over an array?

Fast insert at the start .. unlike arrays that requires moving all the items one step forward

Fast Delete .. unlike arrays that requires almost to move half of the elements a step back.

2. What is the advantage of a doubly linked list over a singly linked list?

You can traverse in either direction in a doubly linked list.

3. What is the function of a Linkstack class ?

Implements a stack using a linked list.

4. Write a method that finds the sum of all the items in a linked list.

```
int sum() {
        int sum = 0;
        Link current = front;
        while ( current != null ) {
            sum += current.iData;
            current = current.next;
        }
        return sum;
}
```

5. Write a method that deletes and returns the last item in a linked list.

```
Link deleteLastItem() {
        Link current = front;
        Link prev;

        while ( current != null ) {
            prev = current;

            if ( current.next == null ) {           // current = last node
                        prev.next = null;
                        return current;
            }

            current = current.next;
        }

        return null;                        // if the list has no items
}
```

6. Suppose the linked list pictured below is built from 3 nodes

start → 5 → 7 → 3

a. write a method to find the maximum

```
Node max() {
        Node current = start.next;
        int max = current.data;

        while ( current != null ) {
            if ( current.data < max )
                max =  current.data;
            current = current.next;
        }
        return current;
}
```

7- write a method to read the list in a reverse order.

```
void printReverse( Node head ) {

        if ( head == null )
                return;

        printReverse ( head.next );

        System.out.println( head.data );
}

inside main method :
                                    printReverse ( head.next ); // head.next is the first node
```