



Lists (2)

CS212:Data Structure
1st semester 2011/12
Lecture 5-6

ADT List – Array

```
public class ArrayList<T>
{
    private int maxsize;
    private int size;
    private int current;
    private T[] nodes;

    /** Creates a new instance of ArrayList */
    public ArrayList(int n) {
        maxsize = n;
        size = 0;
        current = -1;
        nodes = (T[]) new Object[n]; }
}
```



ADT List Implementation

```
public boolean full () {  
    return size == maxsize; }  
public boolean empty ()  
    return size == 0; }  
public boolean last () {  
    return current == (size-1); }  
public void findfirst () {  
    current = 0; }  
public void findnext () {  
    current++; }
```

ADT List Implementation

```
public T retrieve () {  
    return nodes[current]; }  
public void update (T val) {  
    nodes[current] = val; }  
public void insert (T val) {  
    for (int i = size-1; i > current; --i) {  
        nodes[i+1] = nodes[i];  
    }  
    current++;  
    nodes[current] = val;  
    size++;  
}
```

ADT List Implementation

```
public void remove () {  
    for (int i = current + 1; i < size; i++) {  
        nodes[i-1] = nodes[i];  
    }  
    size--;  
    if (size == 0) current = -1;  
    else if (current == size) current = 0;  
}  
}
```

ADT List

- ▶ How to use the ADT List?
- ▶ Example: You are required to implement a static method to search for an element e in a list L , and if e is present make current pointer point to e . Use operations of ADT List.
- ▶ The implementation of ADT is available to you as a Java class.



Using ADT List

```
public class TestArrayList {  
    public static void main ( String[] args ) {  
        ArrayList<String> al = new ArrayList<String>(10);  
        String s1= "xyz", s2 = "abc";  
        al.insert(s1);  
        al.insert(s2);  
        al.findfirst();  
        System.out.println(al.retrieve());  
        System.out.println(al.full());  
  
        System.out.println(length(al));  
        System.out.println("Hello, World");  
    }  
}
```



Using ADT List

```
public static <T> int length(ArrayList<T> l)
{
    int count = 0;
    l.findfirst();
    while (l.last() == false) {
        count++;
        l.findnext();
    }
    return count;
}
```

A static method
to find the
length
of a list.

Note: it has
been
implemented

using the
methods of
ADT List



Comparison: Linked & Array Based Lists

- ▶ Comparison on the basis of number of Operations in *worst case* time complexity of insert & remove operations. All other operations have time complexity (one operation).
- ▶ Linked List: insert– ??; remove – ??
- ▶ Array List: insert – ??; remove – ??
- ▶ Best case time complexities?

Comparison: Linked & Array Based Lists

▶ Linked list.

- Disadvantage: A pointer at each node so more memory needed. For traversal, pointer hopping is required.

▶ Array based list.

- Advantages: arrays are faster; no pointers to be stored – less memory. No pointer hopping
- Disadvantage: list size must be known in advance. Insert – $O(n)$. Remove – $O(n)$.

List: Other Implementations

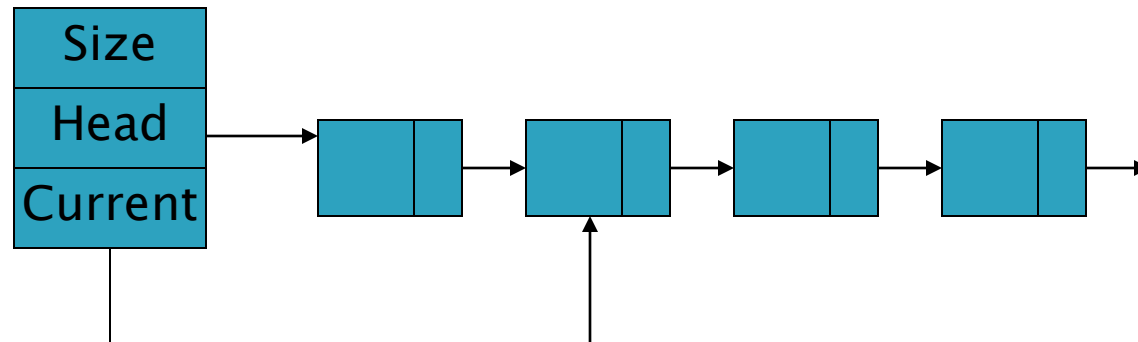
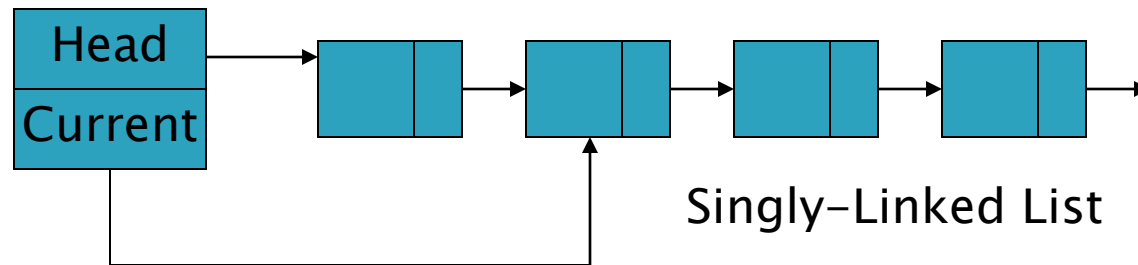
▶ Singly-linked list.

- Design Variations: (i) Count of elements may be kept i.e. *size*. Why? (ii) pointer to tail may be kept i.e. *last*. Why?

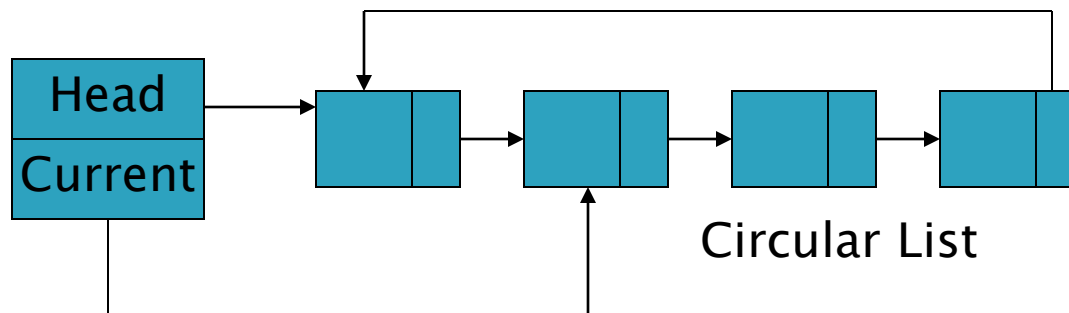
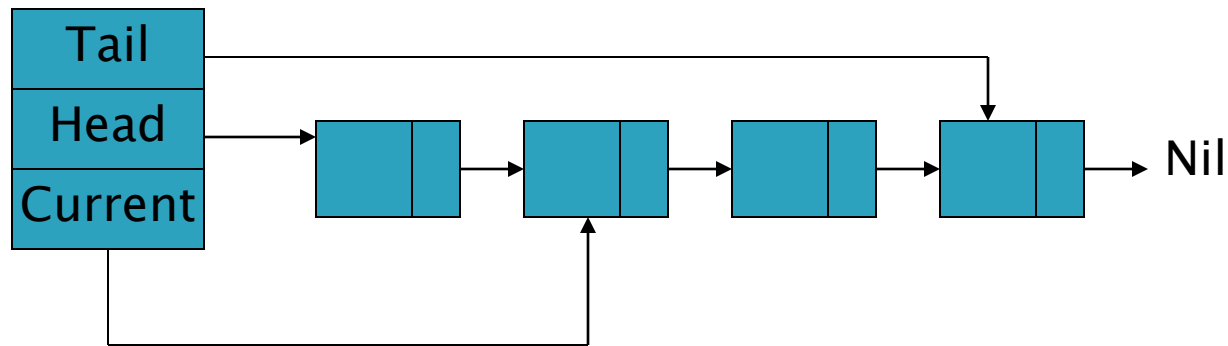
▶ Doubly-Linked list. each node has two pointers: next node and previous node.

- Advantages: it is efficient to go from a node to its previous node or move back-to-front in the list; operations insert – $O(1)$; remove – $O(1)$. (We will cover this topic in detail later)

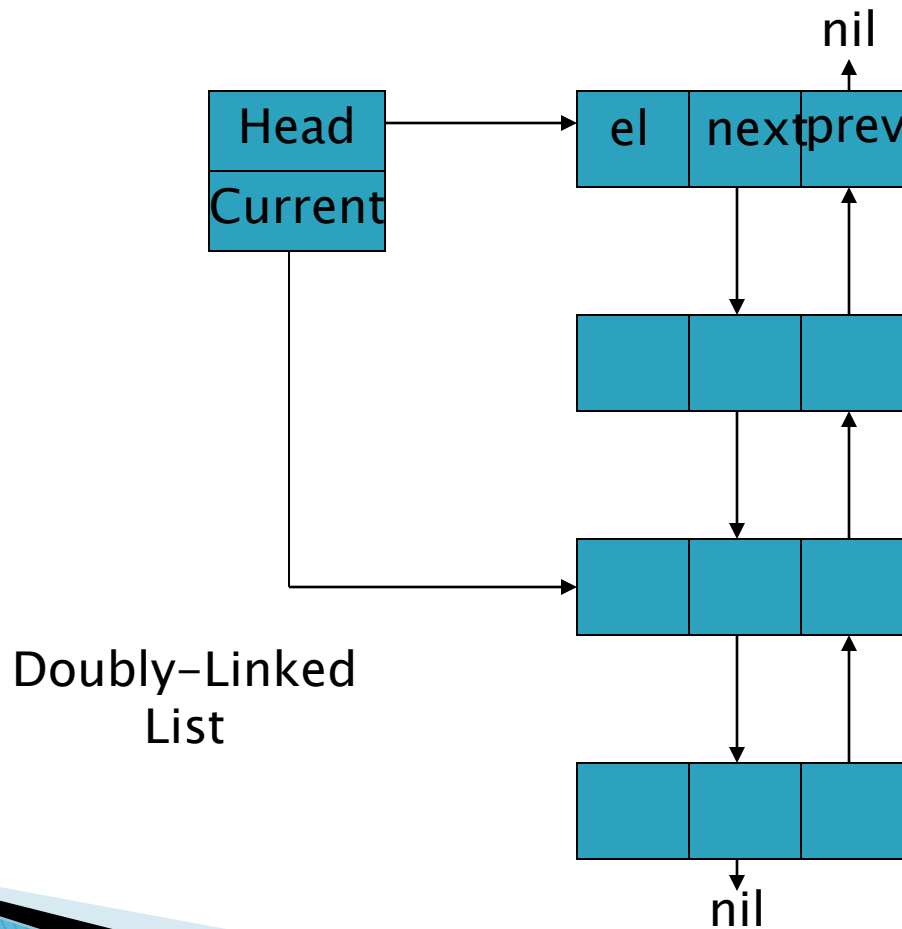
List: Singly Linked



List: Singly Linked & Circular



List: Doubly-Linked



List: Other Implementations

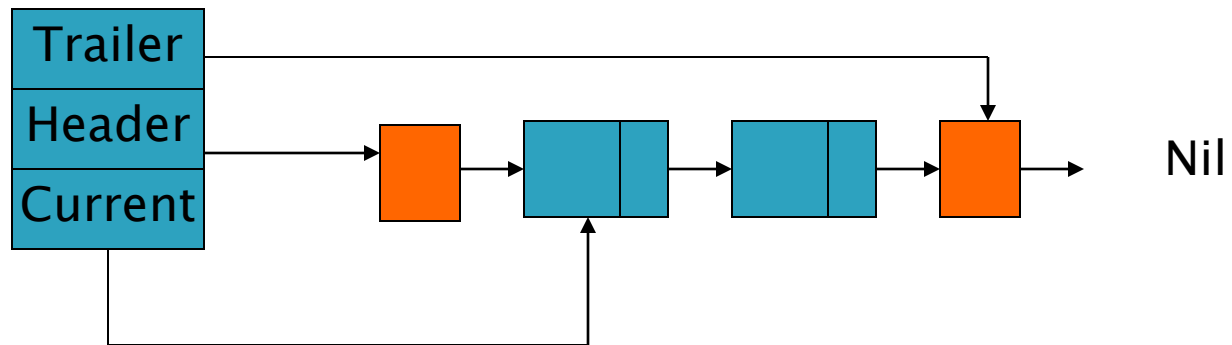
▶ List with Sentinel Nodes.

- Has special header & trailer nodes that do not store data
- All nodes that store data have previous & next nodes – so no special cases for insert & remove.
- Advantage– simplifies code

▶ Circular List.

- tail pointer made to point to the head → tail has next node. Advantage: simpler code.

List: Sentinel Nodes



List with sentinel nodes

Search Item (member method)

```
public boolean find(T key)
{
    node<T> Temp=current;
    current = head;
    while (current.next != null) {
        if(current.data.equals(key))
            return true;
        currnet=current.next;
    }
    current=Temp;
    return false;
}
```

Member method that will search for item “Key” if found the current item will be the key and it will return true o/w it will return false and the current item will not change



Search Item (Static method)

```
public static boolean find(ArrayList<T> l, T key)
{
    l.findfirst();
    while (l.last() == false) {
        if(l.retrieve().equals(key))
            return true;
        l.findnext();
    }
    return false;
}
```

