PROGRAMMING II

Linked List

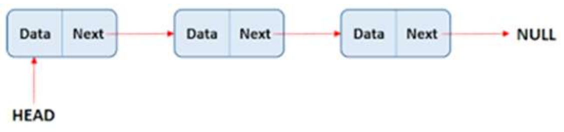
Sherina Sally
Department of ICT,
Faculty of Technology,
University of Colombo

Why Linked List?

- Problems with arrays
  - Unordered array - searching is slow, deletion is slow
  - Ordered array - insertion is slow, deletion is slow
  - Arrays have a fixed length and a specific position: accessed by an index
- Linked lists solves some of these problems
  - How?

Introduction

- A linear data structure that includes a series of connected nodes
- Each node is connected through a link
- A node stores the data value and the address of the next node
- The first node has a special name called HEAD
- The last node points to NULL
- Each node in the list can be accessed linearly by traversing from head to tail
- Linked lists can be of multiple types: singly, doubly, and circular linked list



3

Singly Linked List

```
class Node {
      private  int item ;
      private Node next ;
}
```

- A recursive data structure
- The Node class is referred to itself
- The node object has a value and a reference (next) to the next node in the list

```
public static void main ( String [] args ) {
      Node n1 = new Node();
      Node n2 = new Node();
      Node n3 = new Node();

      n1.item = 10;
      n1.next = n2;
      n2.item = 20;
      n2.next = n3;
      n3.item = 30;

      System.out.println ("N3 = " + n2.next.item);
}
```

4

Singly Linked List …(2)

```
class Node { //class with constructors

private  int item ; // data
private Node next ; // reference to the next node

public Node (int d) {  // constructor
        item = d;
        next = null ;
}

public Node () {  // constructor
        item = 0;
        next = null ;
}

}
```

5

Singly Linked List …(3)

```
class LinkedList {

private Node head ;

//initial linked list
public LinkedList () {
        head = null;
}

//check if the linked list is empty
public boolean isEmpty() {
        return (head == null);
}

}
```

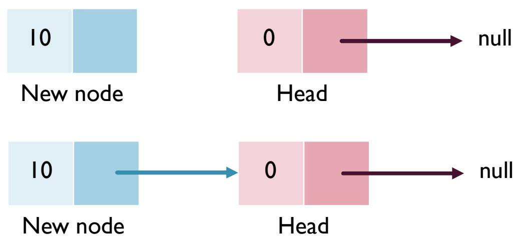
6

Operations

- Insert a node
- Search for a node
- Delete a node
- Display the linked list

Insert at the Beginning

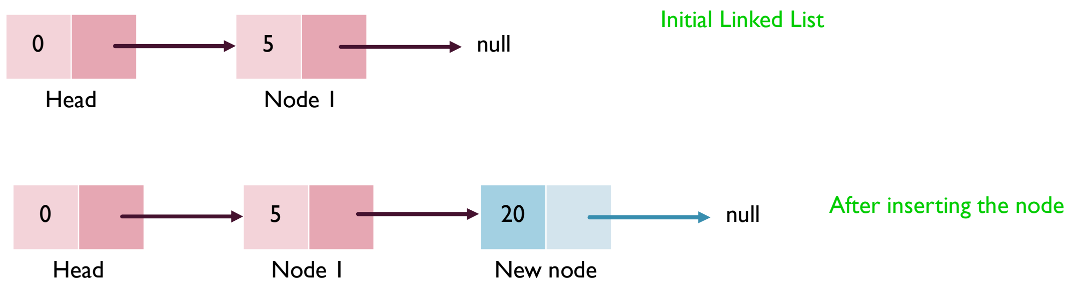
- Add a node before the head node
- Link the new node to the previous head node



```
class LinkedList {

private Node head ;

public void insertAtStart (Node newNode){

    Node n = head;
     if (n != null){
         newNode.next = n;
         n = newNode;
     }
        head = n;
    }

}
```

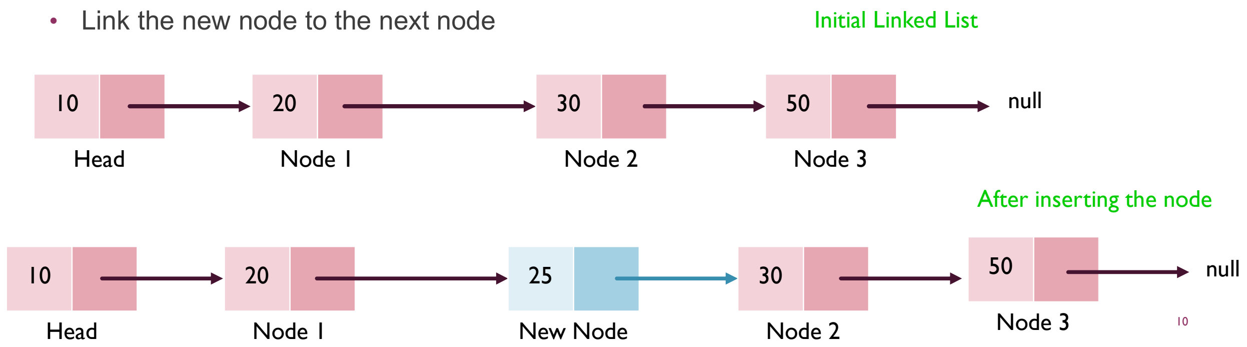
Insert at the End

- Add a node after end node
- The new node points to the null



Insert at the Middle

- Locate the place where the node has to be inserted
- Link the previous node to the new node
- Link the new node to the next node



Insert at the Middle …(2)

```
class LinkedList {

public void insertAtMiddle (Node prev, Node n){

    if(prev.next != null){
          n.next = prev.next;
          prev.next = n;
        }
        prev.next = n;
}

}
```

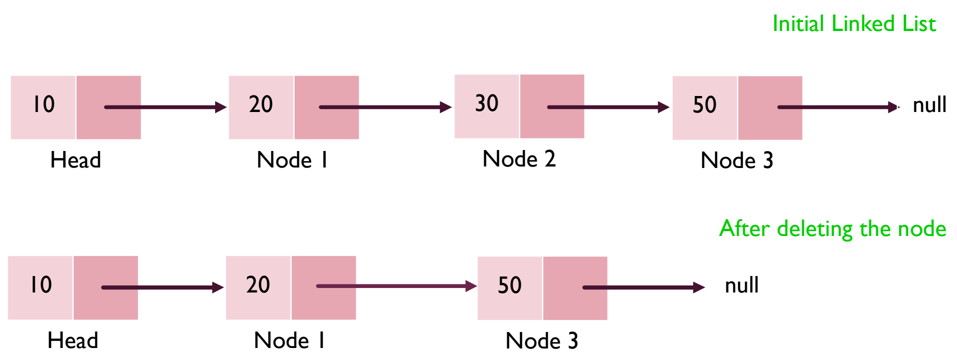
Search in a Linked List

- Traverse the list, checking for the value to be searched

```
1) algorithm Contains($head$, $value$)
2)     Pre: $head$ is the head node in the list
3)          $value$ is the value to search for
4)     Post: the item is either in the linked list, true; otherwise false
5)     $n \leftarrow head$
6)     while $n \neq \emptyset$ and $n$.Value $\neq value$
7)         $n \leftarrow n$.Next
8)     end while
9)     if $n = \emptyset$
10)        return false
11)    end if
12)    return true
13) end Contains
```

Delete a Node

- Locate the place where the node has to be deleted
- Link the previous node to the next node



13

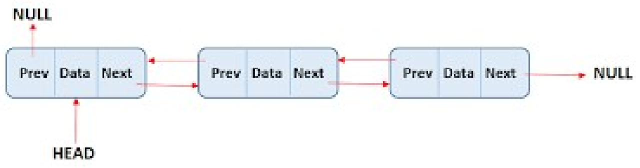
Delete a Node …(2)

```
class LinkedList {

public static void deleteNodeAfter (int data){

      Node prev = null;
      Node curr = head;

      while(true){
         if(curr.next != null && curr.item == data){
            prev.next = curr.next;
            break;
         }
         prev = curr;
         curr = curr.next;
      }
      }
```

14

Doubly Linked List

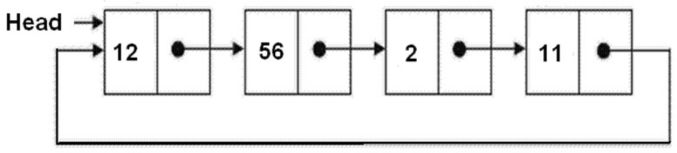
- The node has two links:
  - to the previous node
  - to the next node
- Able to traverse both forward and backward



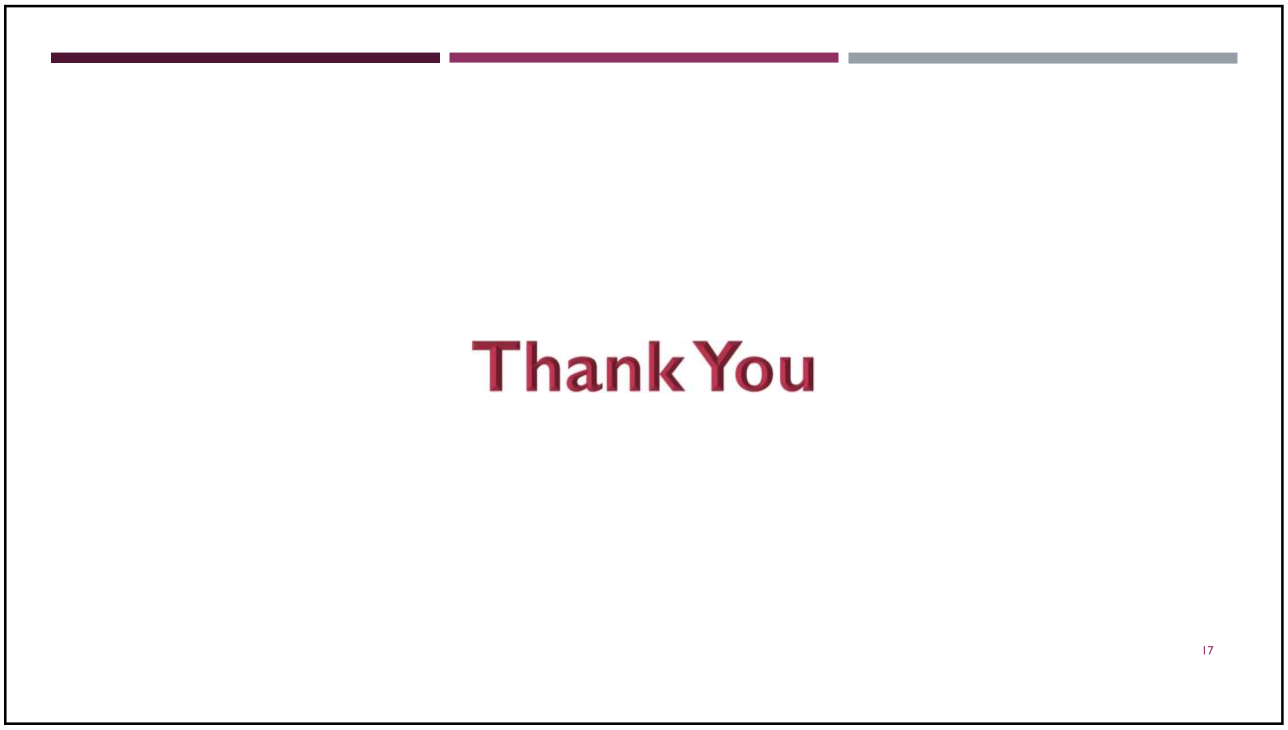
15

Circular Linked List

- The last node of the linked list is pointed to the first node
- Forms a circle
- Singly linked list or a doubly linked list can be used to create a circular linked list



16



Thank You

17