Singly Linked List 3

Computer Science Department

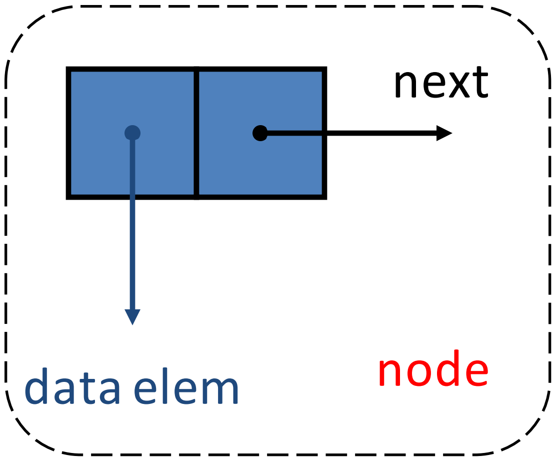
Eastern Washington University

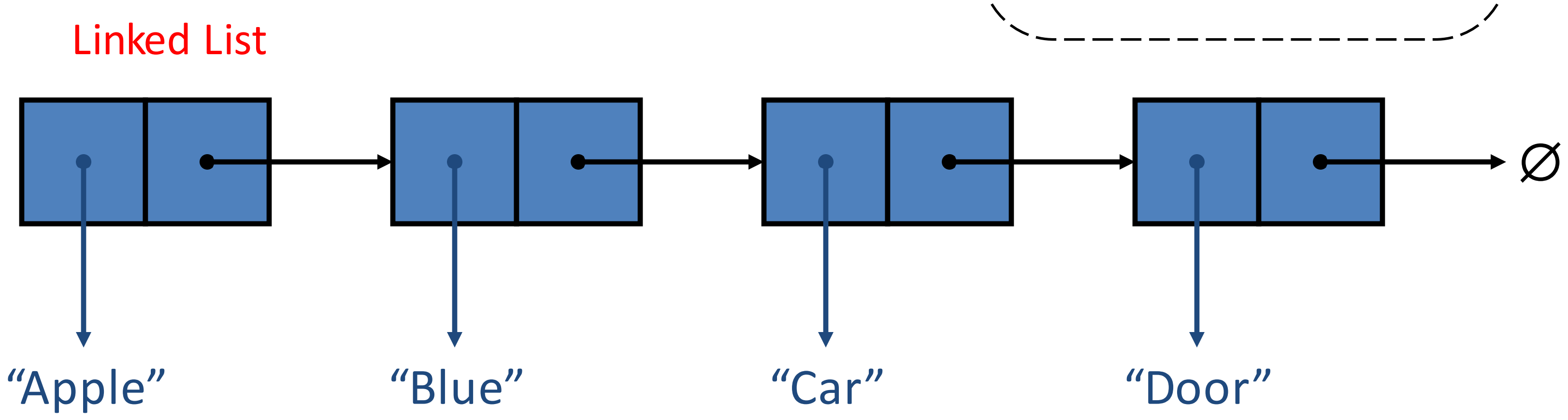
Yun Tian (Tony) Ph.D.

Review

- Concept of Singly Linked List
- How to implement it?
  - LinkdList Class
  - Inner Node Class
  - addFirst()
  - addLast()
  - remove(index)

Concept of Singly Linked List





LinkedList Class Implementation

```
public class LinkedList implements Iterable<Object> {
        private class Node {
                private Object data;
                private Node next;

                private Node( Object data, Node next ) {
                        this.data=data;
                        this.next = next;
                }
                private Node( Object data ) {
                        this(data, null);
                }
                private Node() {} // Can we leave out this empty constructor?
        }//end of node

        private Node head;
        private int size;

        //………….to be continued on next page
```

Today Class

- More operation on LinkedList
  - toString()
  - Remove(Object obj)

LinkedList Class Implementation

```
    public LinkedList() {
        this.head = null;
        this.size = 0;
    }


public String toString() {
    String result = "”;
    //walk through the list
    //explain how cur changes,
    for ( Node cur = this.head; ___Cur != null___; cur = cur.next ) {
       result += cur.data + ",\n";
    }
    return result;
  }
```

Remove data element

- **public boolean remove (Object dataToRemove)**
- We have to go through each list node to check whether the data in the current node equals to the dataToRemove or not.
  - We need a loop to do this.
- The loops stops at the node if we find one.
- Otherwise, the iterator goes through all nodes and reach the end of the list, the walker reference variable becomes null.

Remove data element

- **public boolean remove (Object dataToRemove)**
- At least several cases to handle
  - If the target dataToRemove is not found in the list. (edge case)
  - If the first list node contains the target data that we like to delete.(edge case)
  - The target node is in the middle of the list, or is the last node in the list. (normal case)

LinkedList Class Implementation

```
public boolean remove ( Object dataToRemove ) {
        if( isEmpty() || dataToRemove == null )
            return false;
        Node cur = this.head, prev = null;
        while (_cur != null && ! cur.data.equals(dataToRemove)__)
        {
            prev = cur;
            cur = cur.next;
        }
        if( cur == null ) // not existing
            return false;
        //edge case
        if( prev == null ) //remove from front
        {
            this.head = this.head.next;
            this.size --;
            return true;
        }
        prev.next = cur.next;
        this.size --;
        return true;
}
```

LinkedList Class Implementation

```
public boolean remove ( Object dataToRemove ) {
        if( isEmpty() || dataToRemove == null )
            return false;
        Node cur = this.head, prev = null;
        while (_cur != null && ! cur.data.equals(dataToRemove)_ )
        {
            prev = cur;
            cur = cur.next;
        }
        [redacted] not existing

        //edge case
        if(__prev == null__ )  //remove from front
        {
            this.head = this.head.next;
            this.size --;
            return true;
        }
        prev.next = cur.next;
        this.size --;
        return true;
}
```

What if we forget to handle this edge case?

LinkedList Class Implementation

```
    public boolean remove ( Object dataToRemove ) {
            if( isEmpty() | | dataToRemove == null )
                return false;
            Node cur = this.head, prev = null;
            while (_cur != null && ! cur.data.equals(dataToRemove)___)
{
                prev = cur;
                cur = cur.next;
            }
            if( _cur == null___ ) // not existing
                return false;
            [redacted] ve from front
            ;
            prev.next = cur.next;
            this.size --;
            return true;
    }
```

What if we forget to handle this edge case?

LinkedList Class Implementation

```
public boolean removeAll ( Object dataToRemove ) {
    if( isEmpty() || dataToRemove == null )
            return false;
     Node cur = this.head, prev = null;
     boolean deleted = false;
     while ( cur != null ) {
         if(cur.data.equals(dataToRemove)) {
             if(cur == this.head) {
                 this.head = this.head.next;
             }
             else {
                 prev.next = cur.next;
             }
             cur = cur.next;
             this.size --;
             deleted = true;
         }
         else {
             prev = cur;
             cur = cur.next;
         }
    }//end of while
    return deleted;
}
```

Summary

- toString()
- Remove first occurrance of a data element from linked list.
- Remove all occurrence of a data element.

Next class

- – List Iterator
- – List with Dummy Node(aka head node)