Computer Science 112
Data Structures

Lecture 05:
Circular Linked Lists
Doubly-Linked Lists

Review: Linked List Methods

- public static IntNode last(IntNode front)
- public static IntNode append(IntNode a,
  IntNode b)

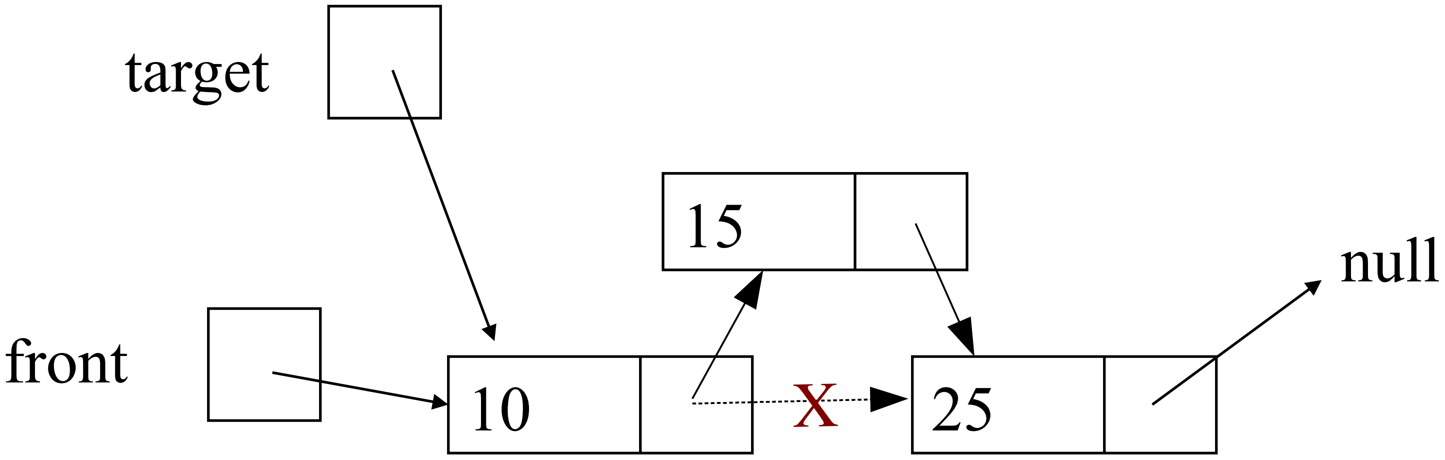
** There was a bug in append !! **

see LLApp versions/rev 2.5/LLApp.java

fixed in LLApp versions/rev 3/LLApp.java

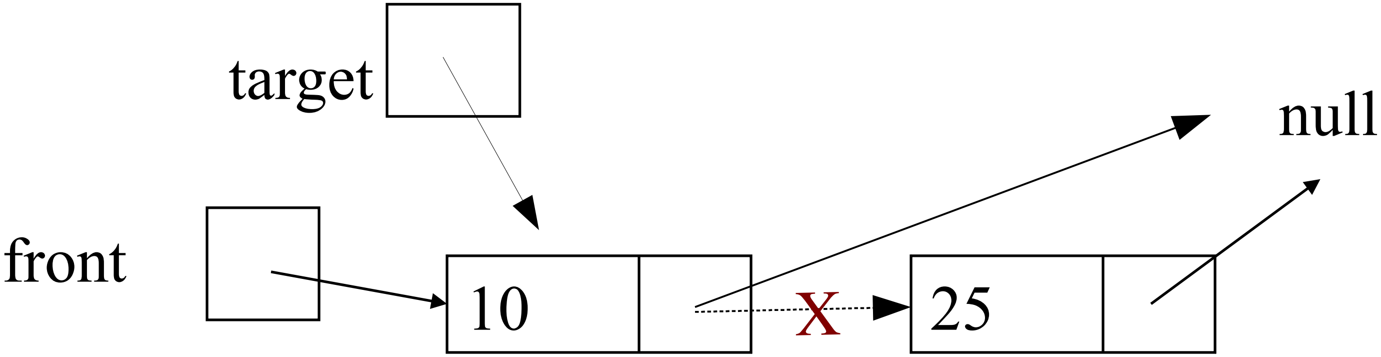
addAfterNode

```
public static void addAfterNode(IntNode front,
                                IntNode target,
                                 int item){
   target.next = new IntNode(item, target.next);
}
```



deleteAfterNode

```
public static void deleteAfterNode(
                                    IntNode front,
                                    IntNode target)
        target.next = target.next.next;
}
```

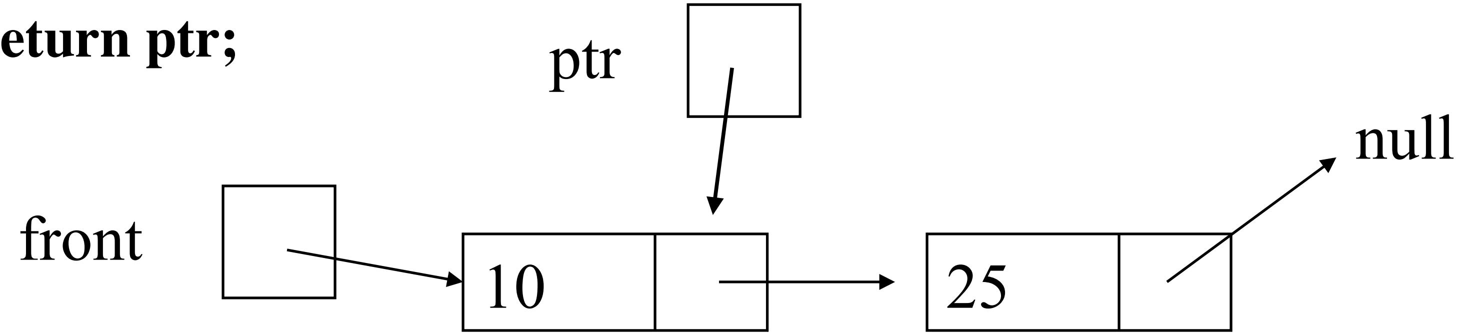


deleteNode

```
public static IntNode deleteNode(IntNode front, IntNode target) {
    IntNode ptr=front, prev=null;
    while (ptr != null && ptr != target) {
        prev = ptr;
        ptr = ptr.next;  }
    if (ptr == null) {
       return front;
   } else if (ptr == front) {
       return ptr.next;  }
   prev.next = ptr.next;
   return front;}
```

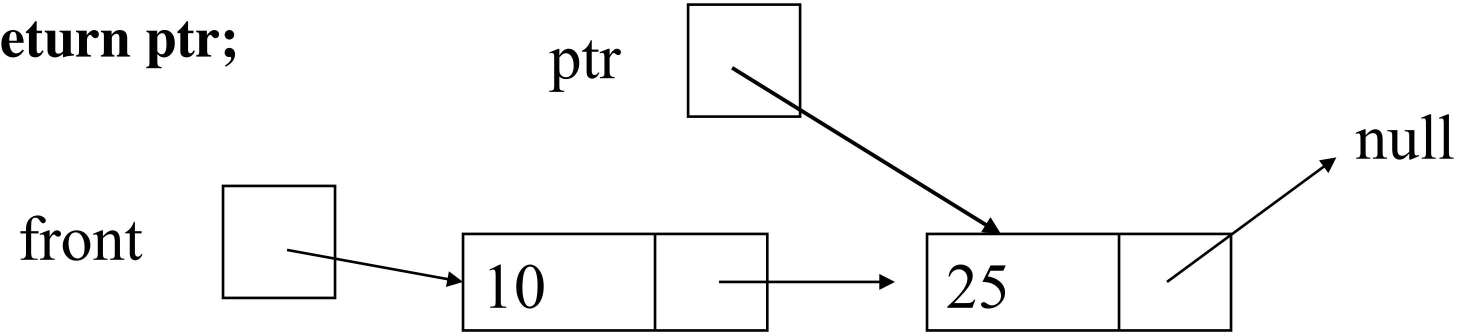
findLast

```
public static IntNode last(IntNode front){
   if (front == null){
       return null;
   } else {
      IntNode ptr;
      for (ptr = front; ptr.next != null; ptr = ptr.next){
      }
      return ptr;
   }
}
```



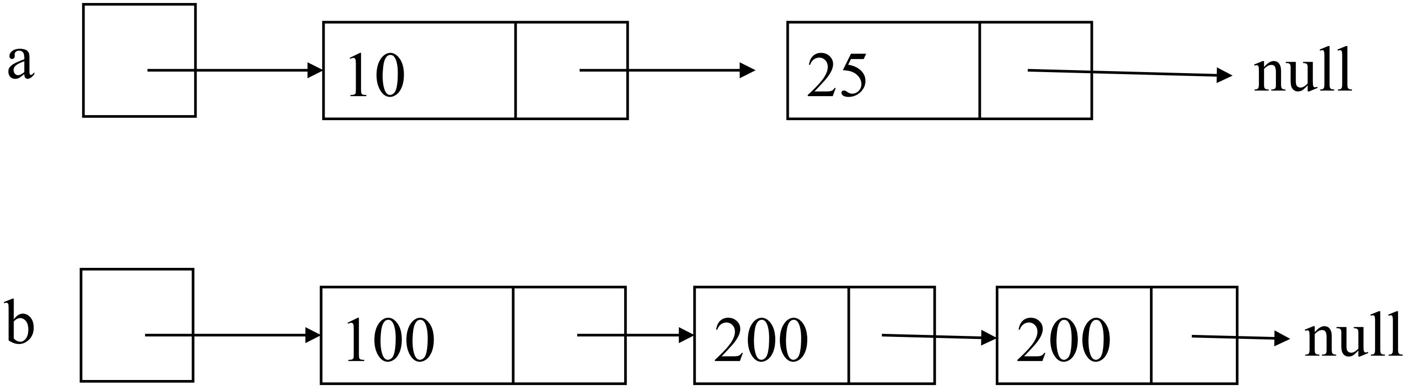
findLast

```
public static IntNode last(IntNode front){
    if (front == null){
        return null;
    } else {
       IntNode ptr;
       for (ptr = front; ptr.next != null; ptr = ptr.next){
       }
       return ptr;
    }
}
```



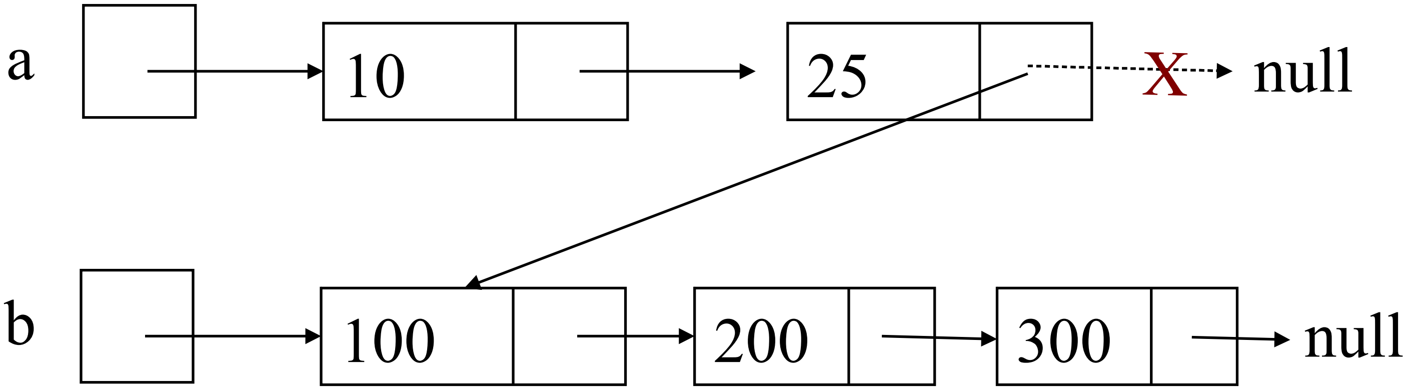
append

```
public static void append(IntNode a,
                          IntNode b){
   last(a).next = b;
}
```



append

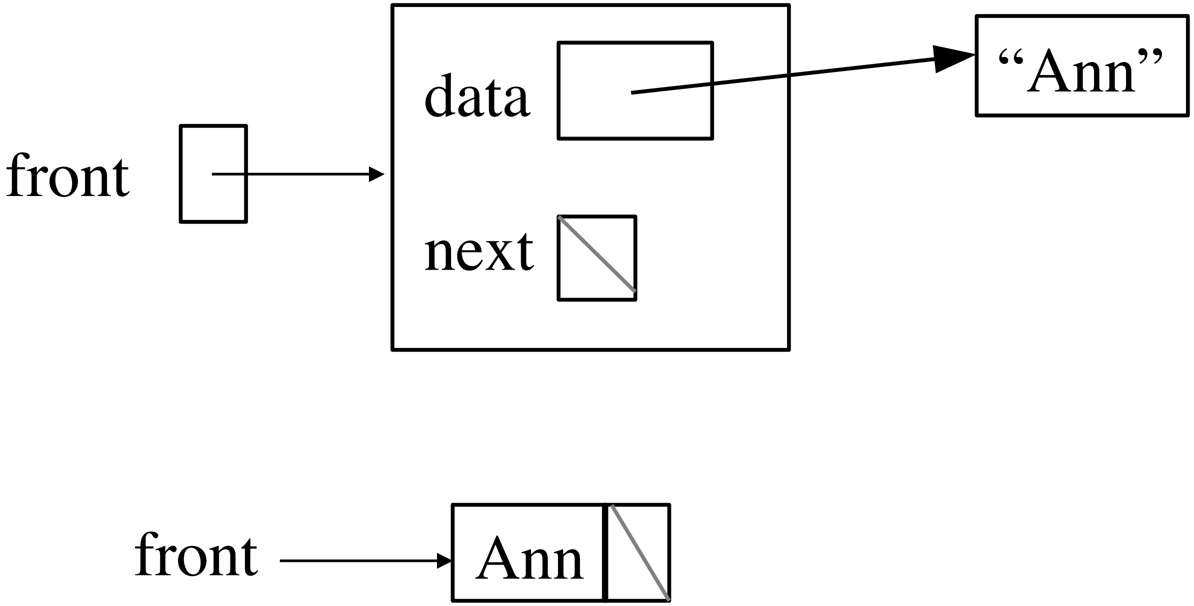
```
public static void append(IntNode a,
                          IntNode b){
  last(a).next = b;
}
```



StringNode

```
public class StringNode{
    String data;
    StringNode   next;
    public StringNode(String data, StringNode next){
        this.data = data;
        this.next = next;
   }
}
```

A One-Element List



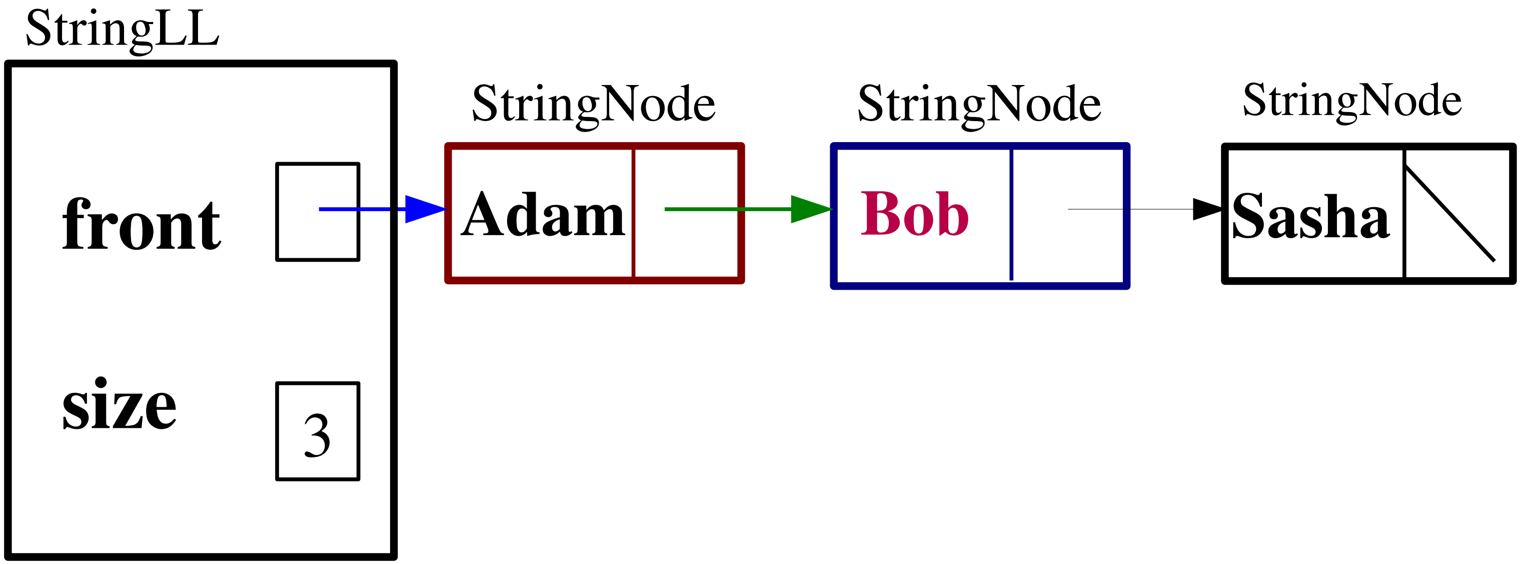
delete

```
public static StringNode delete(StringNode front,  String target) {
   StringNode ptr=front, prev=null;
   while (ptr != null && ! ptr.data.equals(target)) {
      prev = ptr;
      ptr = ptr.next;  }
   if (ptr == null) {
     return front;
  } else if (ptr == front) {
     return ptr.next;  }
  prev.next = ptr.next;
  return front;}
```

A String Linked List class

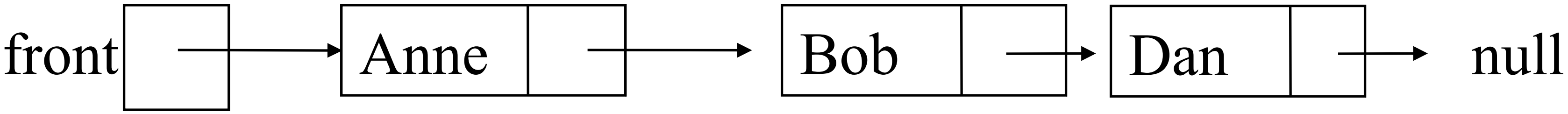
- **In order to represent list as a whole**
  - **To have an object that represents the empty list**
  - **To add extra data such as length of list**
- **You also need a class for the nodes – a good place to use a nested class**
- **See StringLL.java**

A String Linked List class

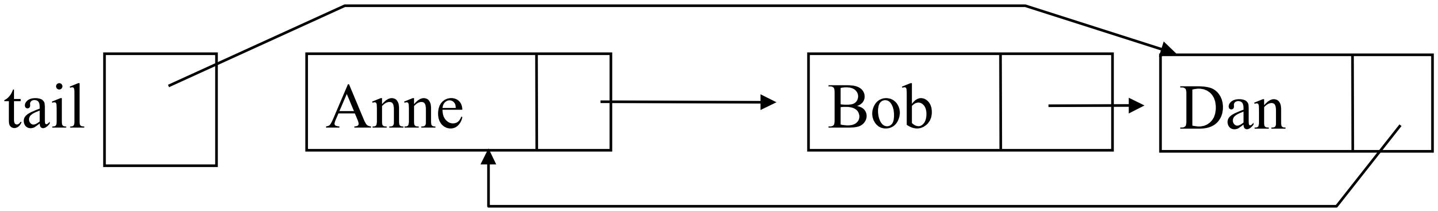


New: Varieties of Linked Lists

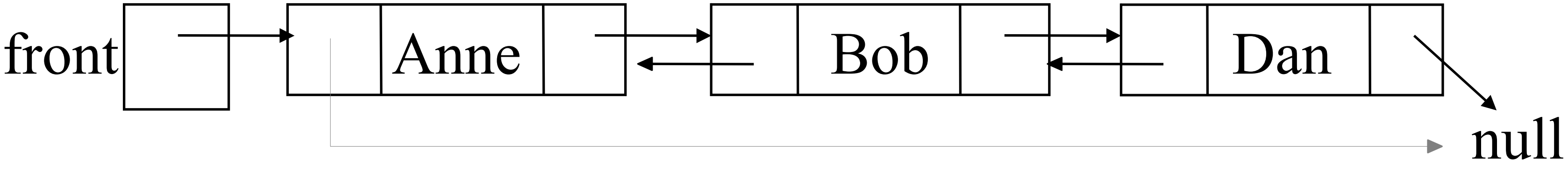
- **Singly Linked List**



- **Circular Linked List**

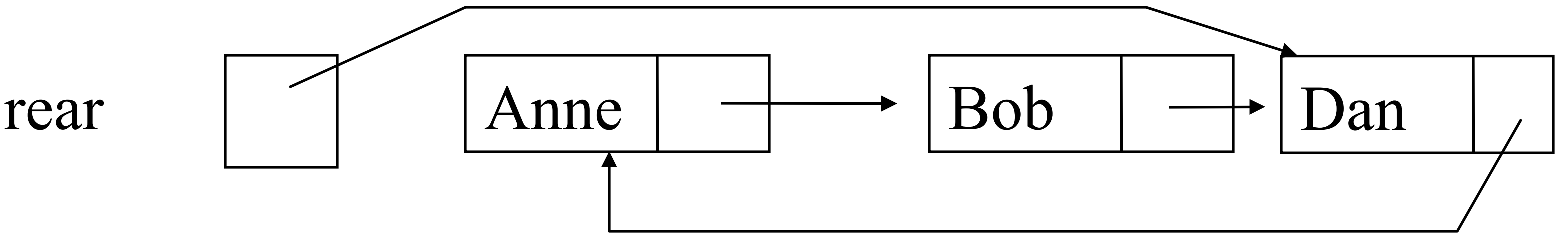


- **Doubly Linked List**

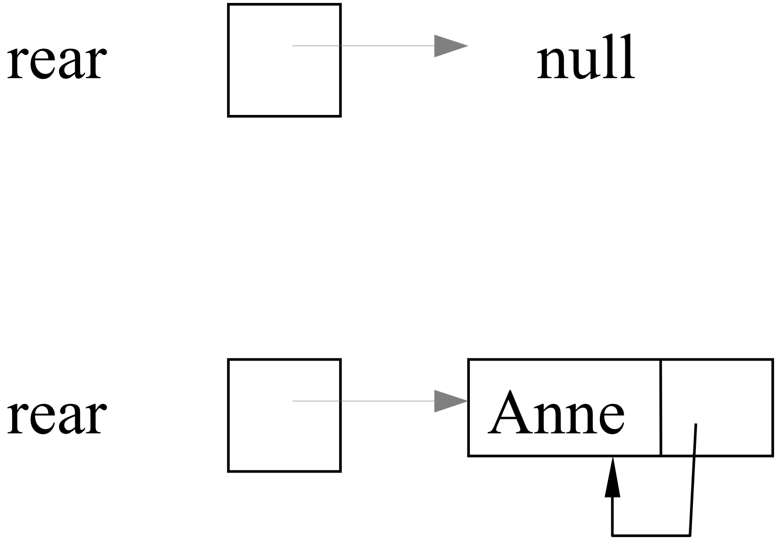


Circular List

- First node is ??
- Variable place points at last node when??

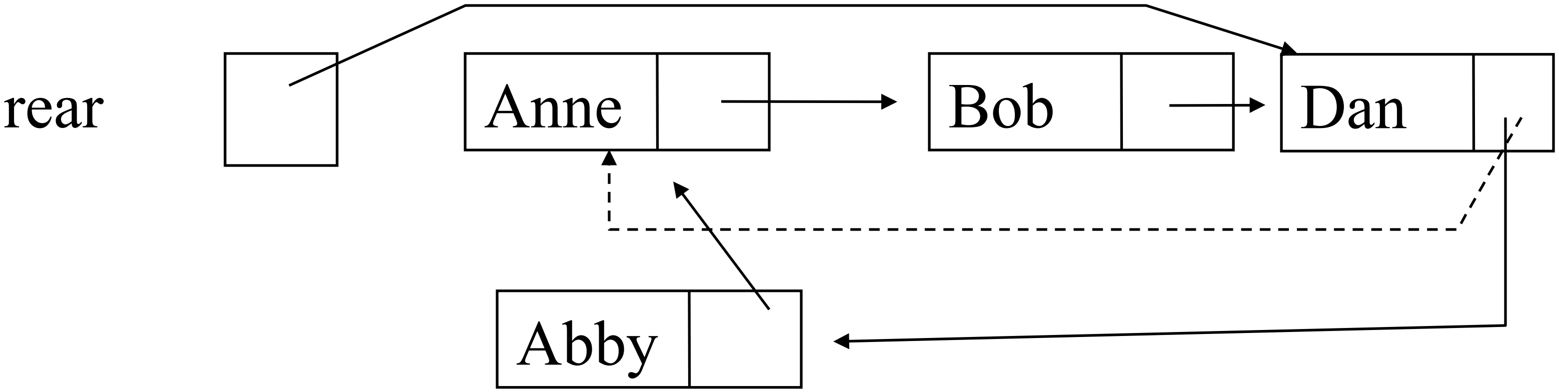


Circular List With No Nodes and With One Node

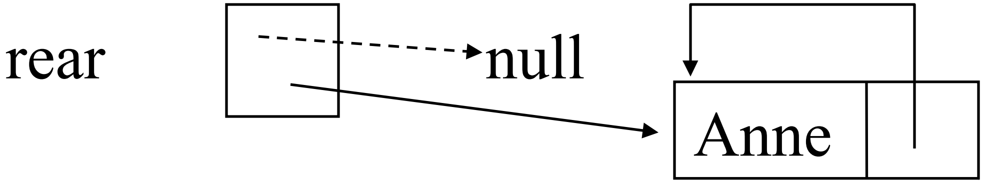


Insert at Head

- **List not empty**



- **List empty**



Insert at Head

```
if(rear == null){  // insert in empty list
     rear =  new Node(newData, null);
     rear.next = rear;
  } else {           // insert in non-empty list
     Node newNode= new Node(newData,
                                  rear.next);
     rear.next = newNode;
  }
  return rear;
```

Delete Head

- Hint: Draw pictures first. There are 3 cases:
  - list already empty
  - list has one node, becomes empty
  - list had more nodes

Add at Rear

- Hint: Draw pictures first. There are 2 cases
  - add to empty list
  - add to non-empty list

Other CLL Methods

- See resources > Steinberg > Java >
  CLLApp.java

- note finding the rear is O(1) but
- removing the rear is still O(n)

Doubly Linked Lists

- See resources > Steinberg > Java > DLLApp.java
- Note that these DLLs are not circular