Computer Science 112
Data Structures

Lecture 04:
More Linked-List Methods
Linked Lists of Strings

Methods

See LLAp.java, rev. 2

- **IntNode addAtFront(int data, IntNode front){**
- **void printList(IntNode front)**
- **IntNode deleteFront(IntNode front)**
- **boolean search(IntNode front, int target)**
- **boolean addAfter(IntNode front,**
  int target,
  int item)
- **IntNode delete (IntNode front, int target)**

Methods

- **addAfterNode**
- **deleteAfterNode**
- **deleteNode**
- **findLast**
- **append**
-

addAtFront

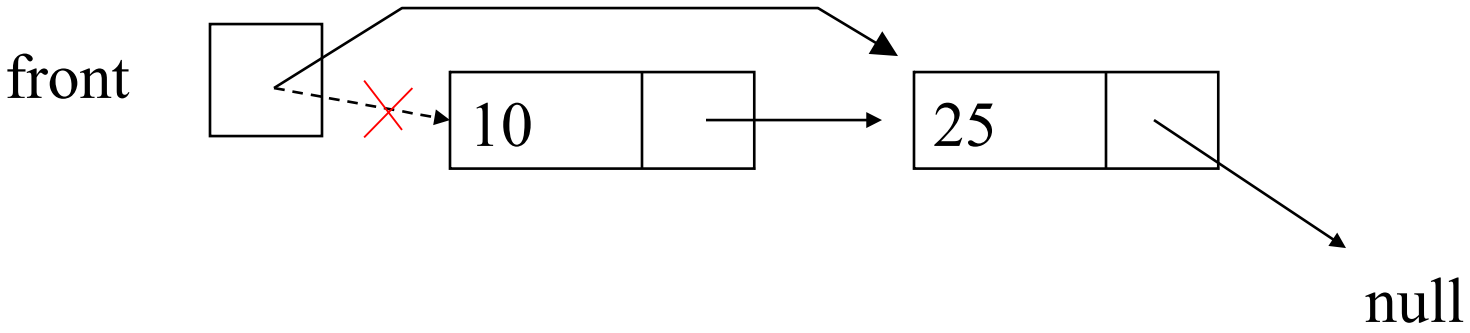
```
public static IntNode addAtFront(int data, IntNode front){
    front = new IntNode(data, front);
    return front;
}
```

printList

```
public static printList(IntNode front){
    for (IntNode ptr = front; // first node
        ptr != null; // continue if not at null
        ptr = ptr . next){ // go to next node
      System.out.println(ptr . data);
    }
```

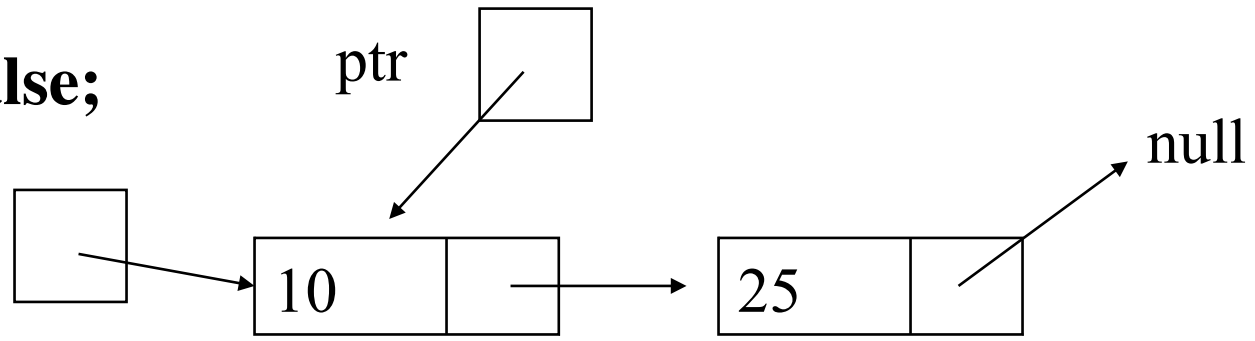
deleteFront

```
IntNode deleteFront(IntNode front){
 front = front.next;
 return front;
}
```



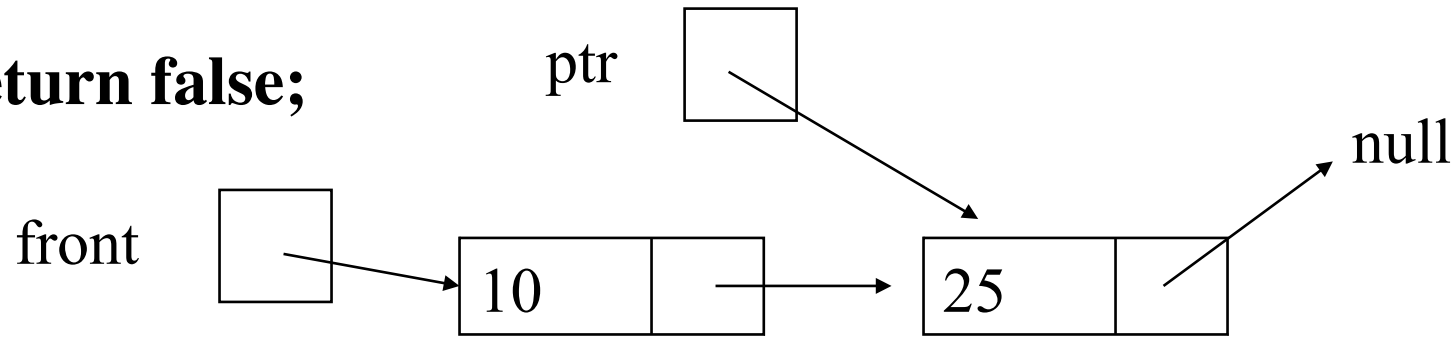
search

```
public static boolean search(IntNode front, int target) {
    for (IntNode ptr = front; ptr != null; ptr = ptr.next) {
        if (target = = ptr.data) {
            return true;
        }
    }
    return false;
}
```



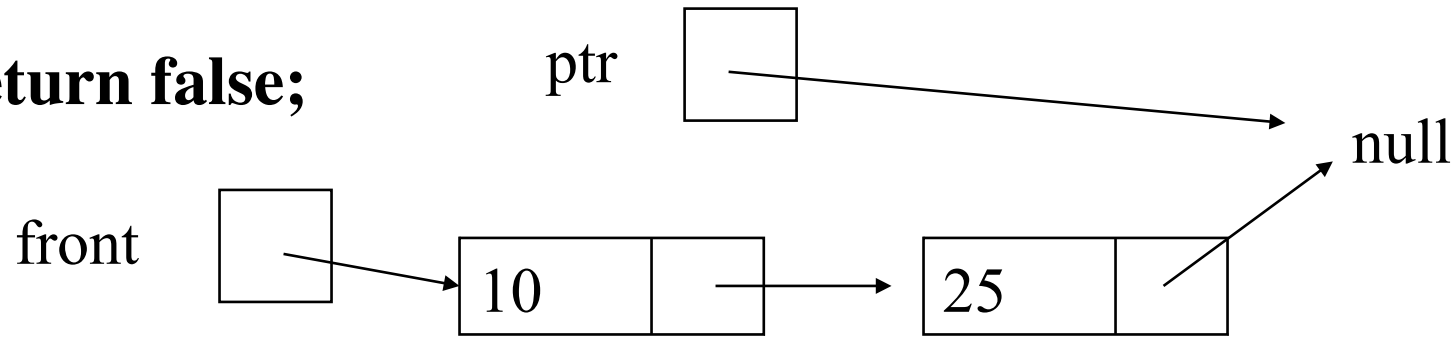
search

```
public static boolean search(IntNode front, int target) {
    for (IntNode ptr = front; ptr != null; ptr = ptr.next) {
        if (target = = ptr.data) {
            return true;
        }
    }
    return false;
}
```



search

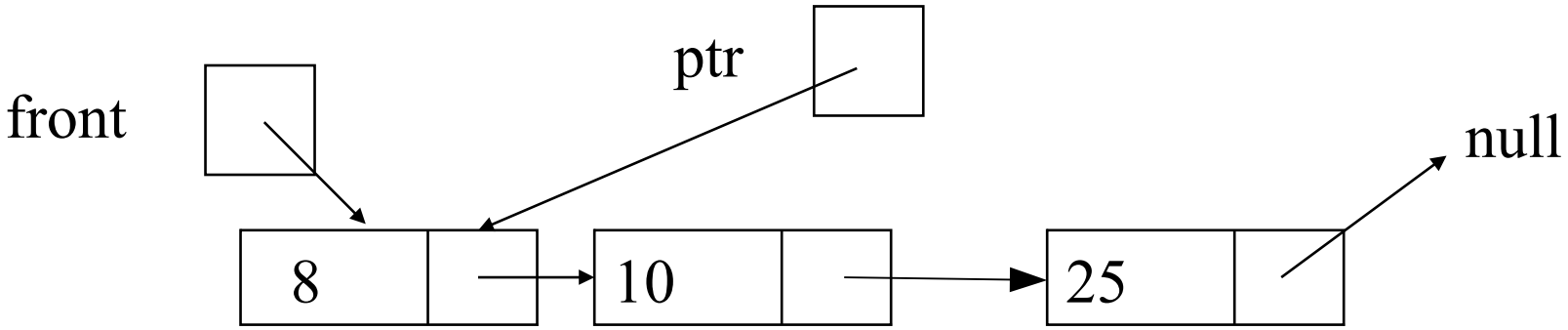
```
public static boolean search(IntNode front, int target) {
    for (IntNode ptr = front; ptr != null; ptr = ptr.next) {
        if (target = = ptr.data) {
            return true;
        }
    }
    return false;
}
```



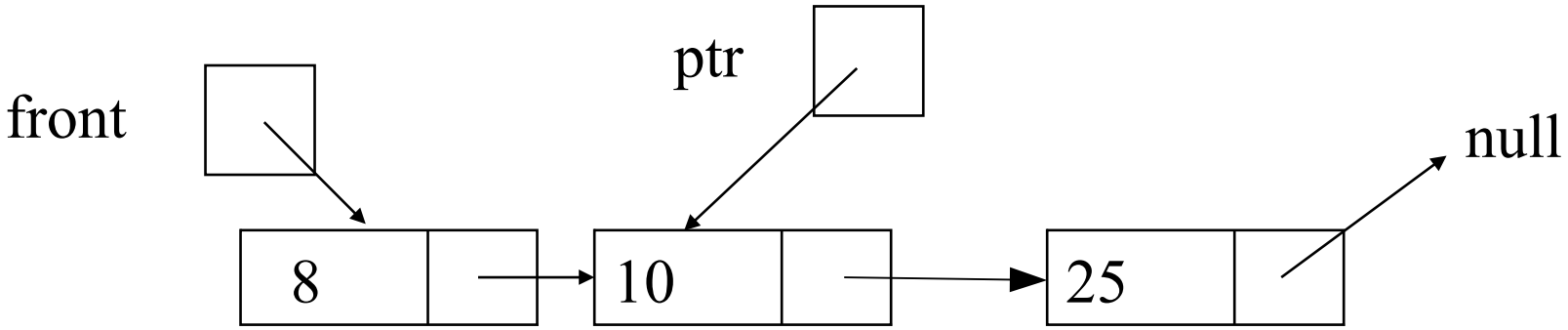
addAfter

```
public static boolean addAfter(IntNode front,
                               int target,
                               int item){
  for (IntNode ptr = front; ptr != null; ptr = ptr.next){
    if (ptr.data == target){
      ptr.next = new IntNode(item, ptr.next);
      return true;
  } }
  return false;
}
```

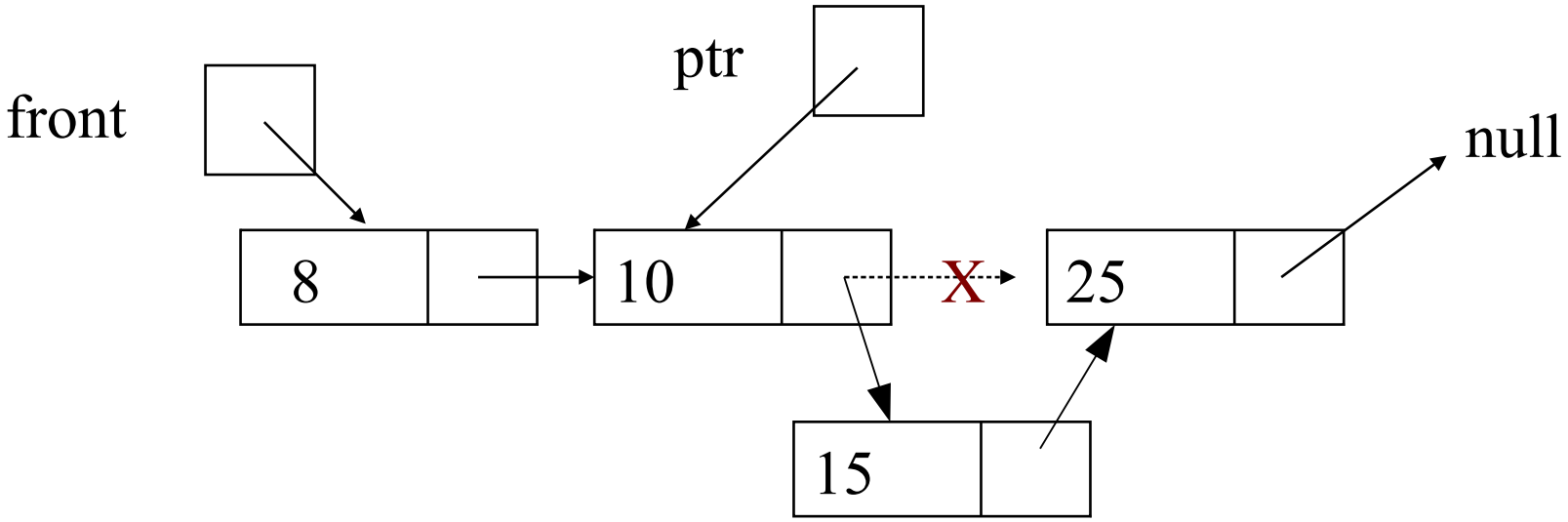
addAfter



addAfter



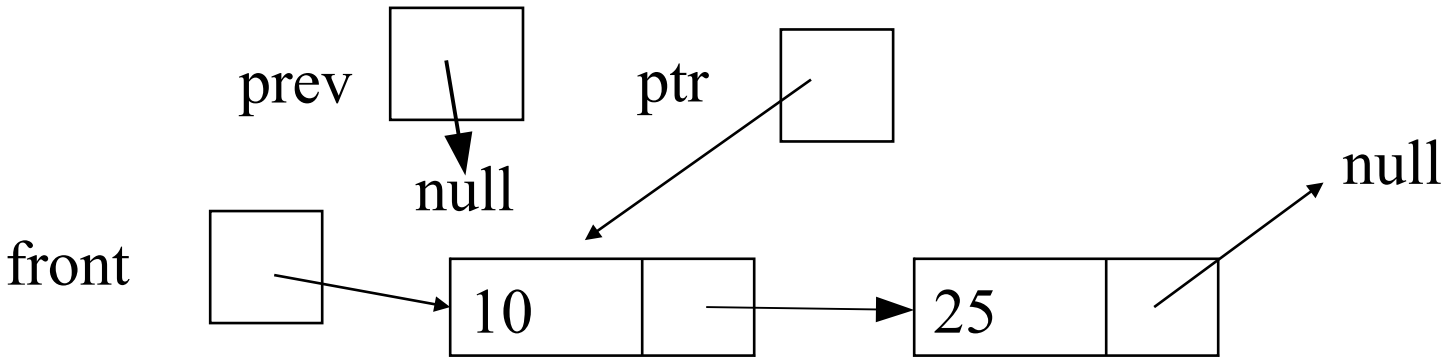
addAfter



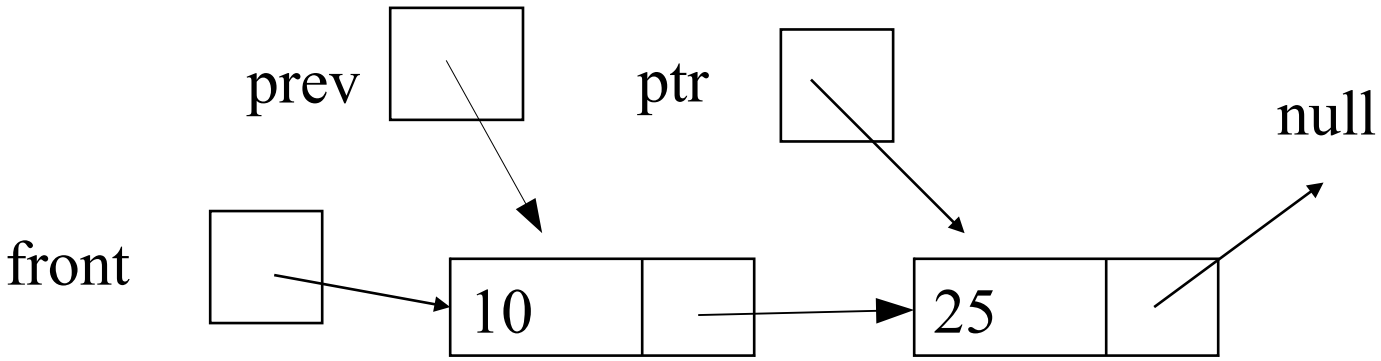
delete

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

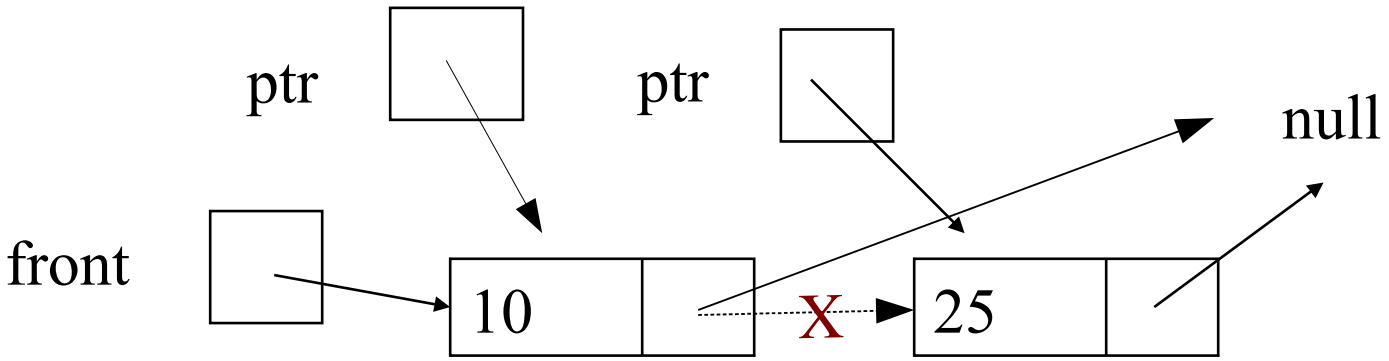
delete



delete

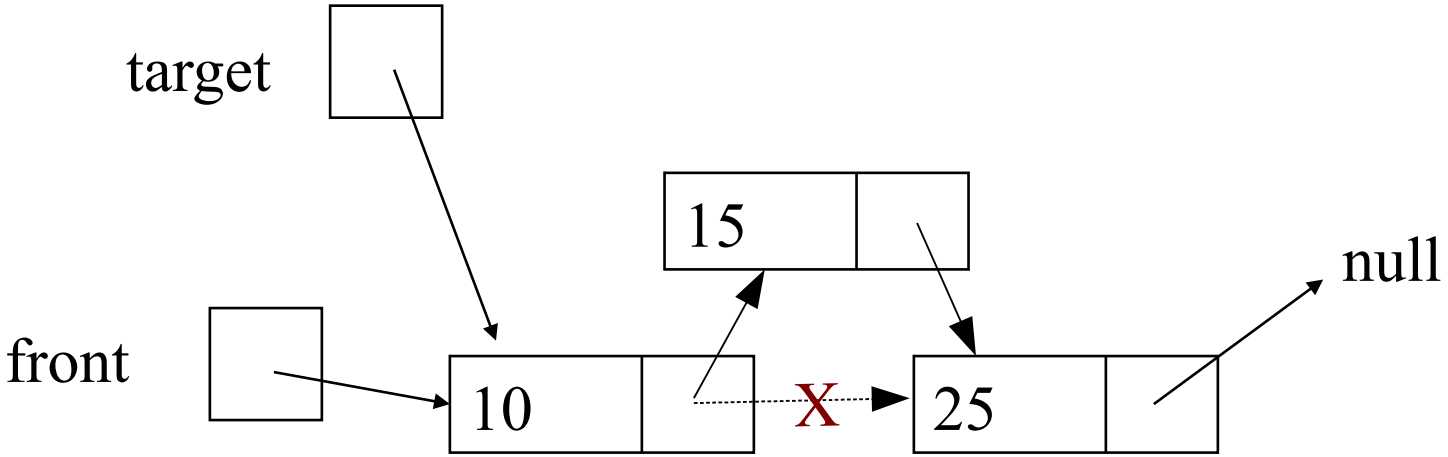


delete



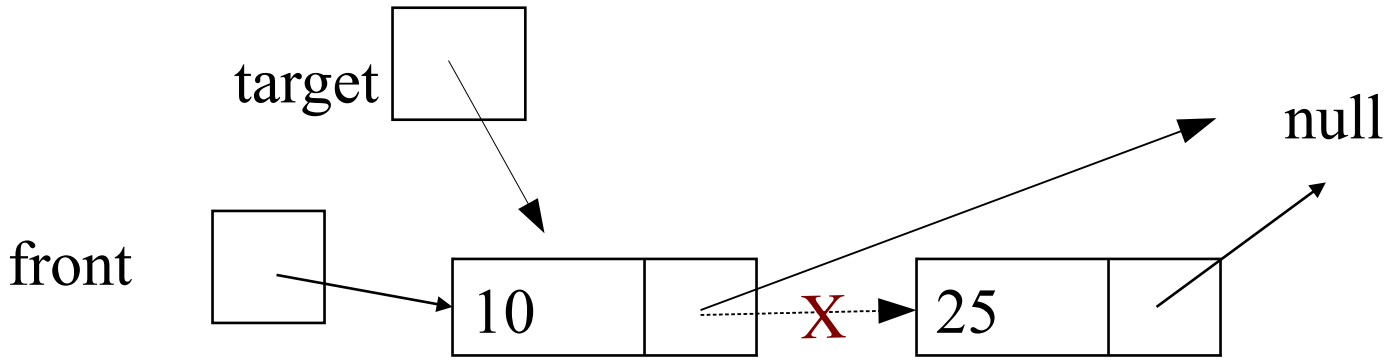
addAfterNode

```
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;
}
```

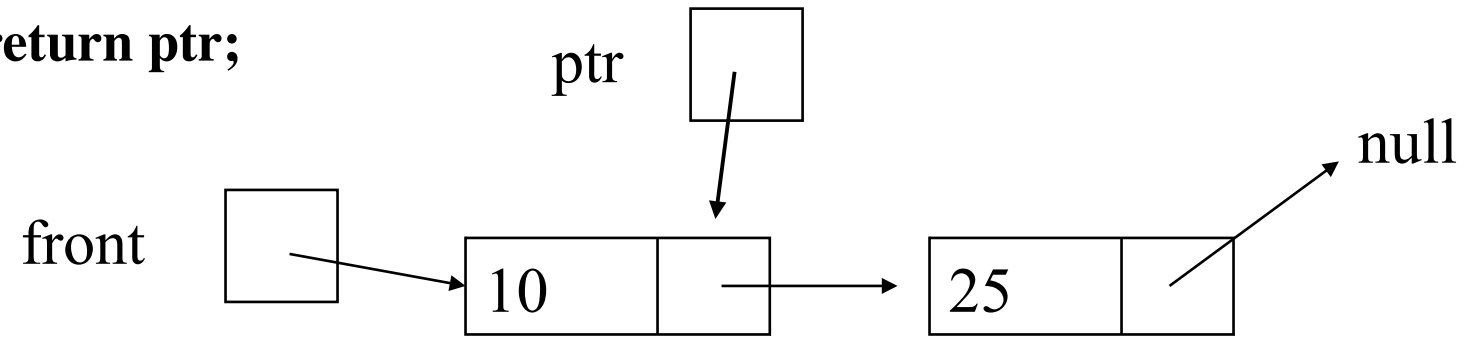


deleteNode

```
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;}
```

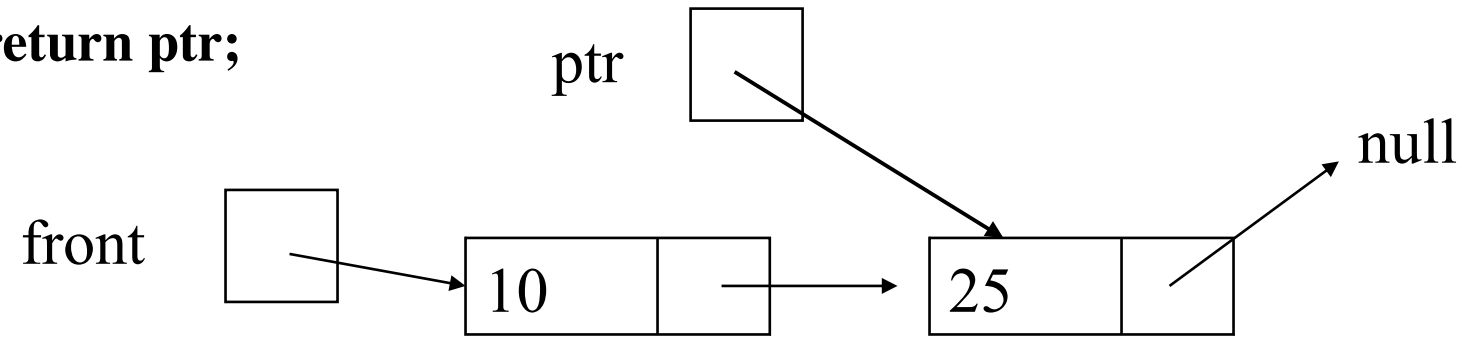
findLast

```
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;
    }
}
```



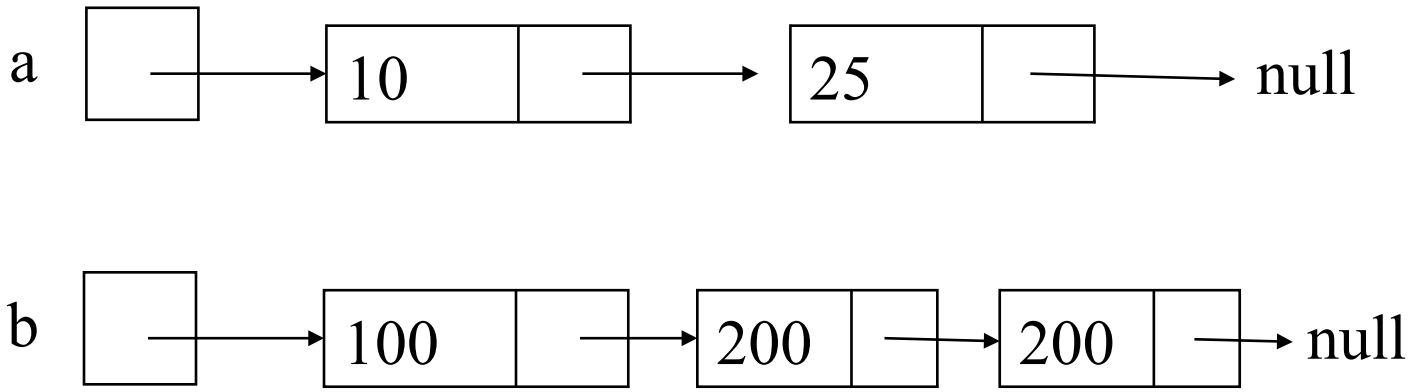
findLast

```
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;
    }
}
```



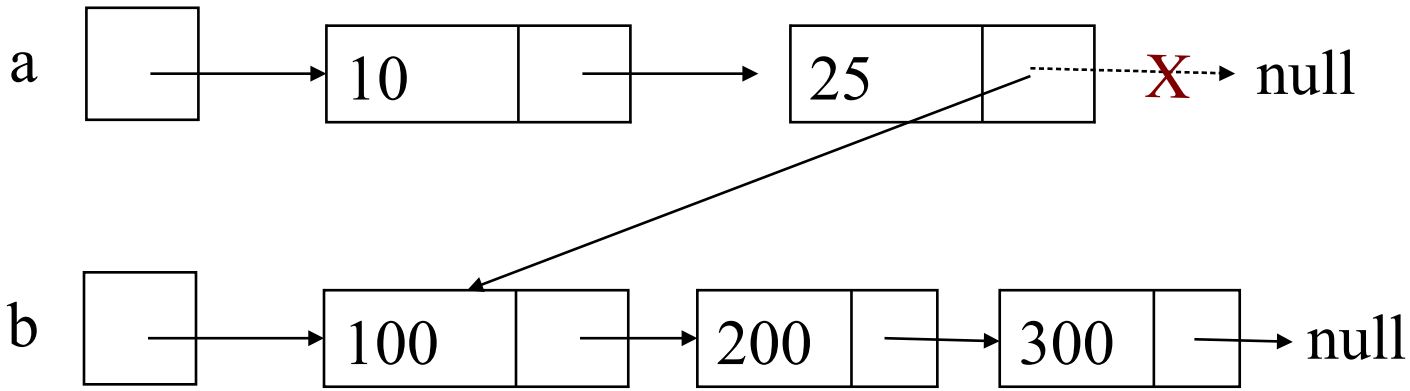
append

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



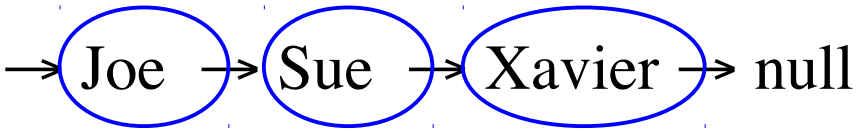
append

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



Linked Lists of Strings

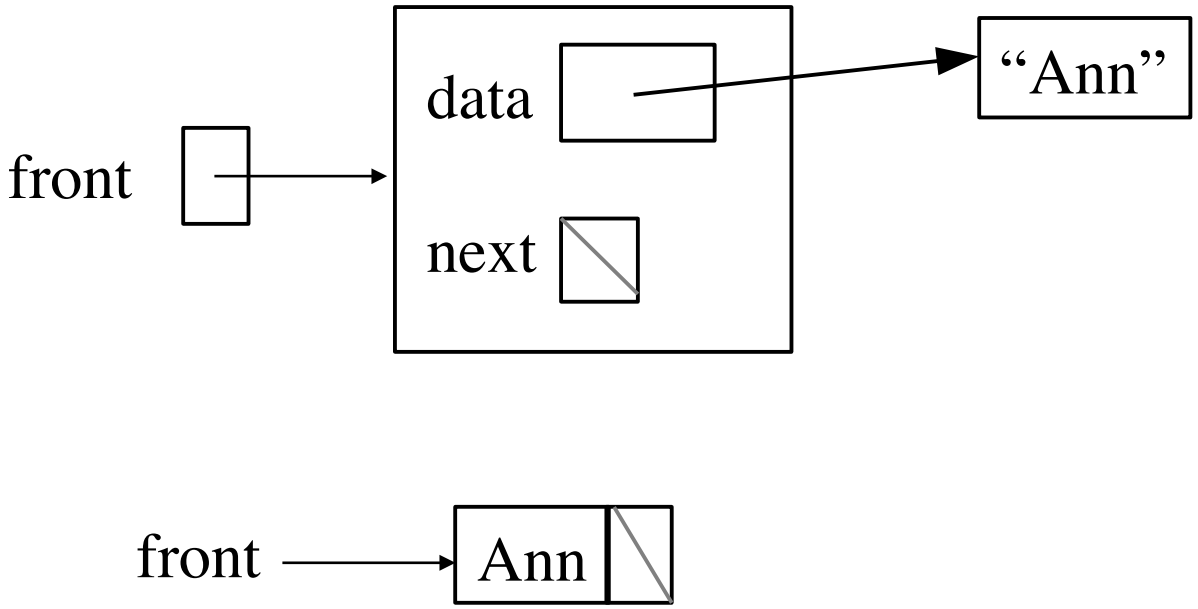
- **So far, data has been ints**
- **What changes if data is Strings?**



StringNode

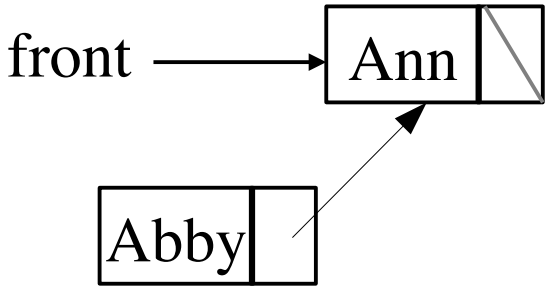
```
public class StringNode{
    String data;
    StringNode   next;
    public StringNode(String data, StringNode next){
        this.data = data;
        this.next = next;
    }
}
```

A One-Element List



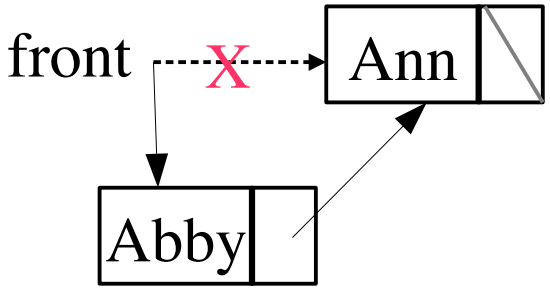
Adding to the front of a list

new StringNode("Abby", front);

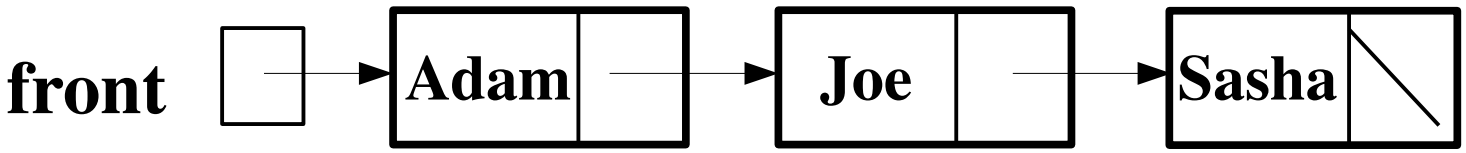


Adding to the front of a list

front = new StringNode("Abby", front);

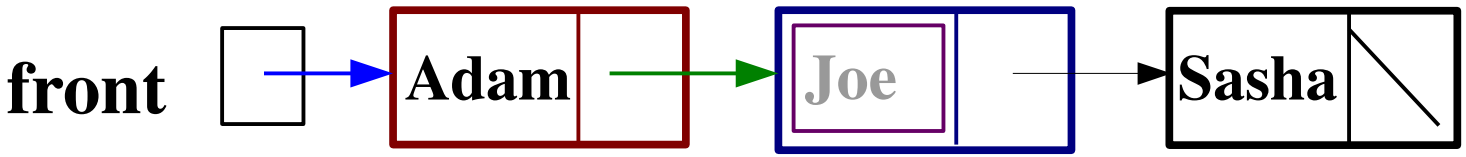


A three-element list



Change second name to Bob

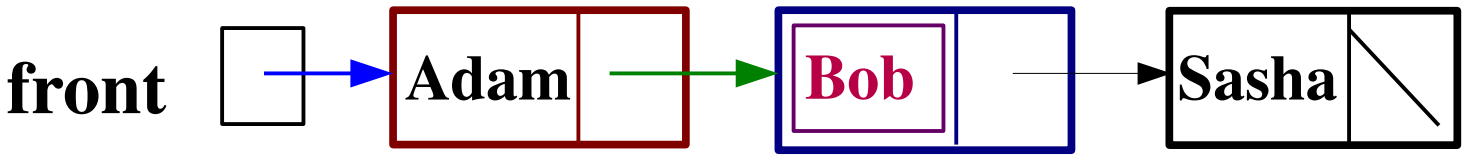
A three-element list



Change second name to Bob

front . next . data =

A three-element list



Change second name to Bob

front . next . data = "Bob"

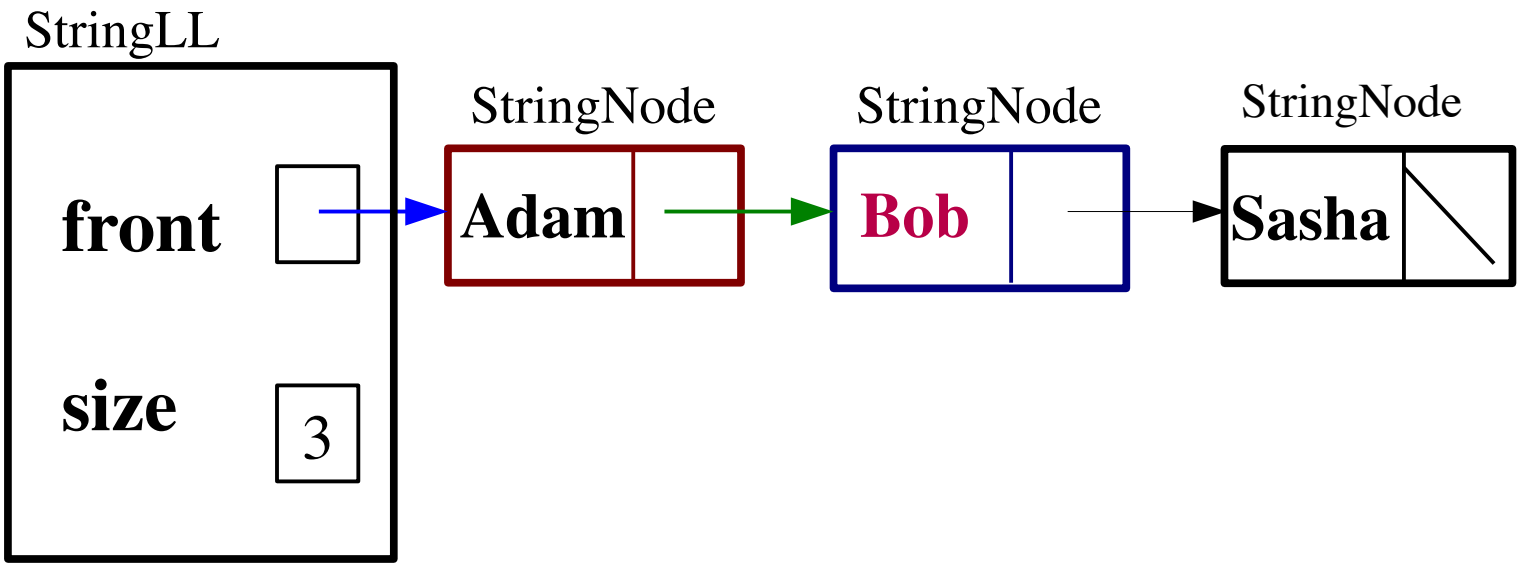
delete

```
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



An Empty List