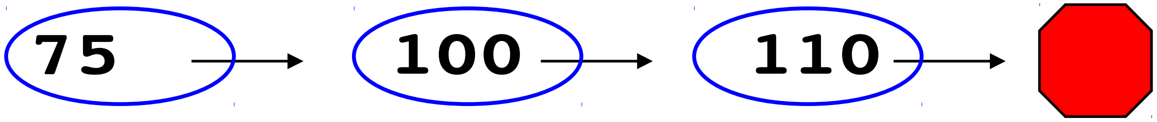
Computer Science 112
Data Structures

Lecture 03:
More operations on linked lists

Linked Lists

- **Suppose what you store is “what comes next”**



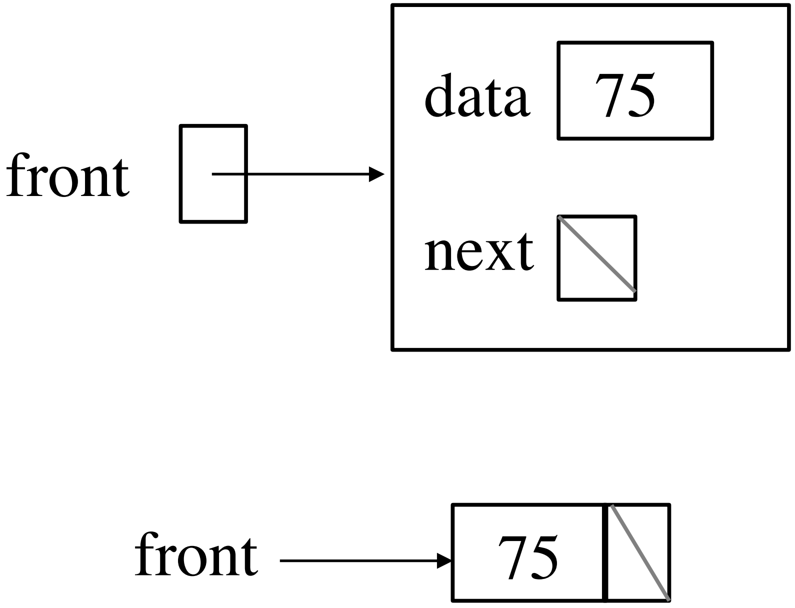
Nodes

- A node is an object that has
  - a field for data
  - a field to refer to the next node in the linked list

```
public class IntNode{
   int data;
   IntNode  next;
   public IntNode(int data, IntNode next) {
      this.data = data;
      this.next = next;
   }
}
```

a *reference* *to* an IntNode object, or null

A One-Element List



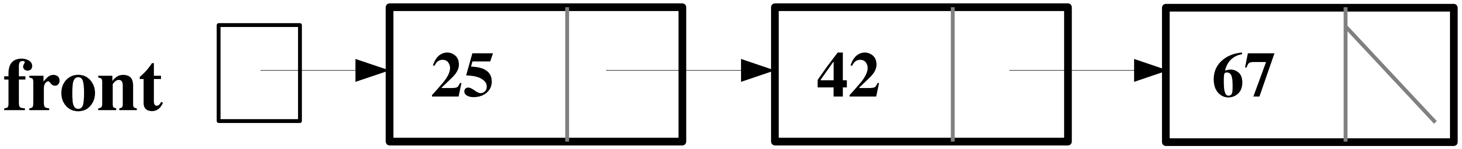
Adding to the front of a list

```
IntNode temp = new IntNode(20,  null);
temp.next = front;
front = temp;
```

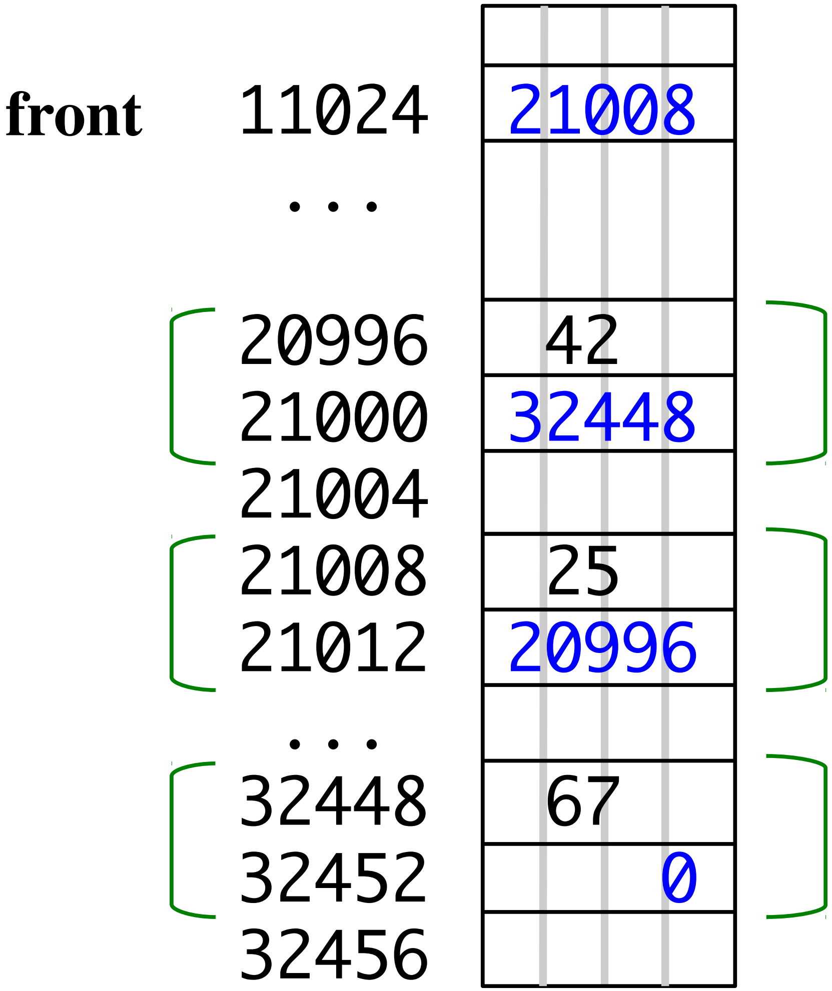
or, in one line:

```
front = new IntNode(20, front);
```

A three-element list

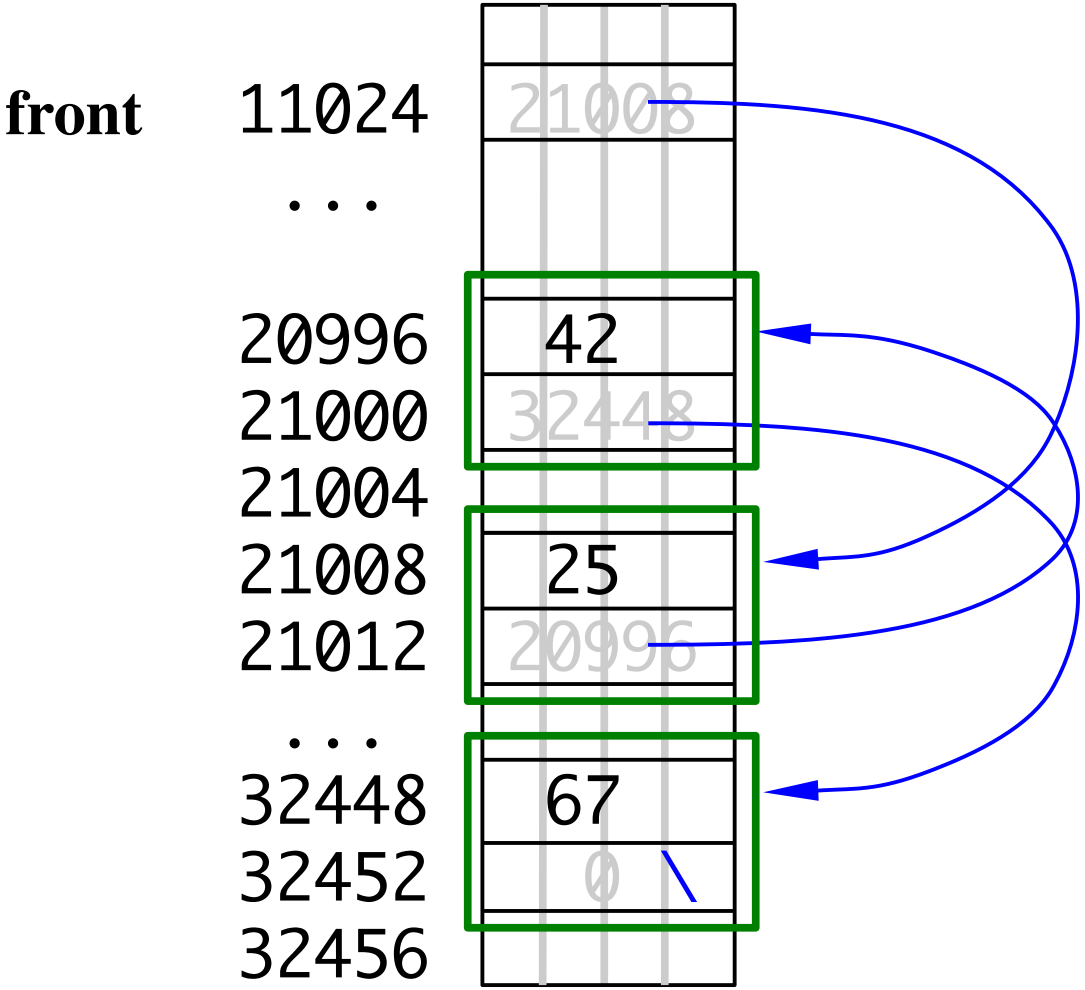


In Memory



front	11024	21008
	. . .	
	20996	42
	21000	32448
	21004	
	21008	25
	21012	20996
	. . .	
	32448	67
	32452	0
	32456	

In Memory



front 11024 21008

...

20996 42

21000 32448

21004

21008 25

21012 20996

...

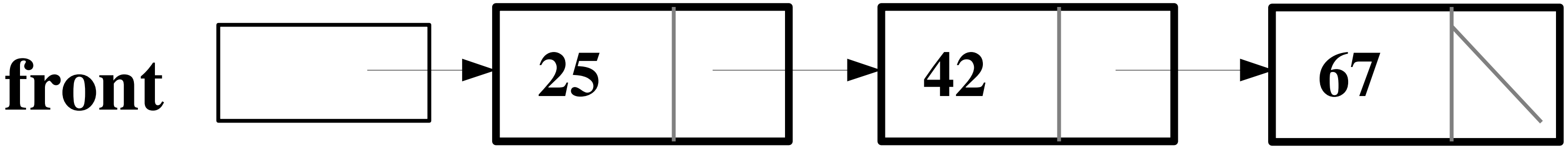
32448 67

32452 0

32456

Accessing a Three-Element List

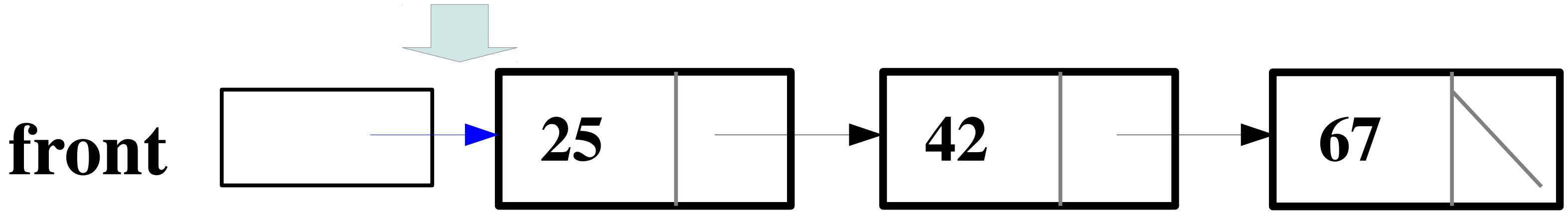
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

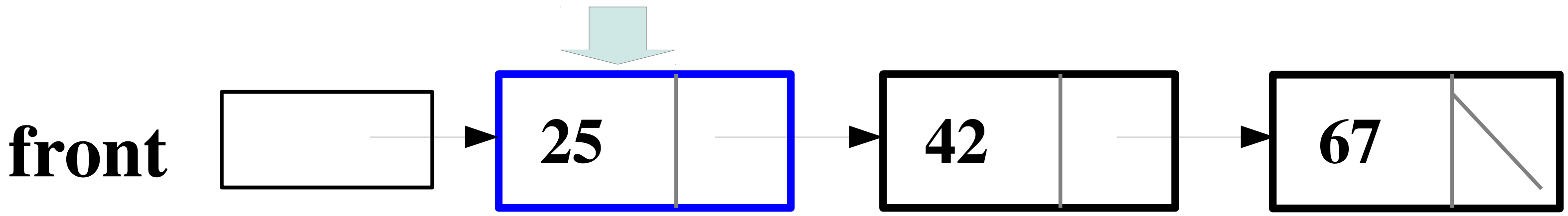
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

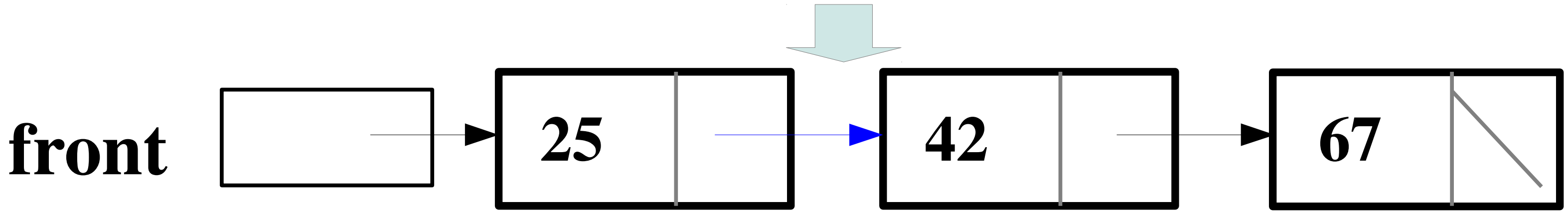
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

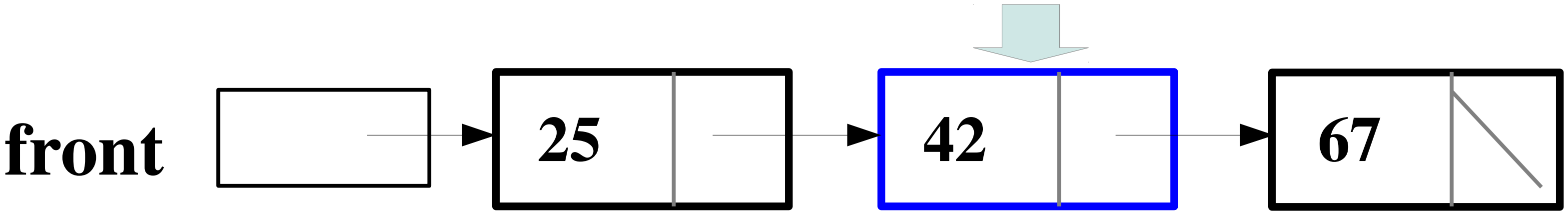
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

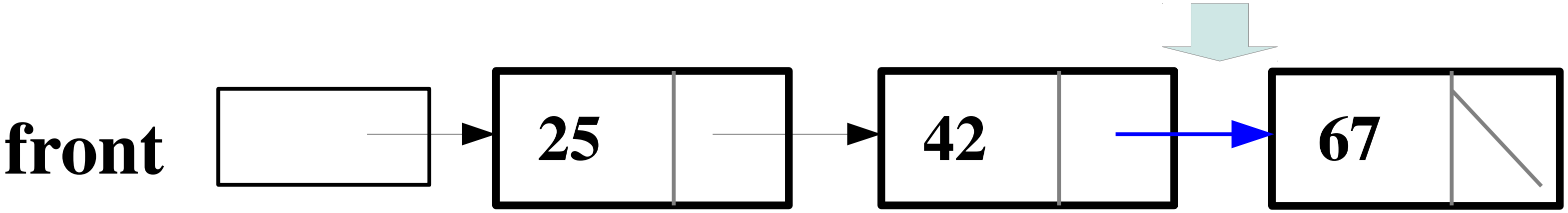
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

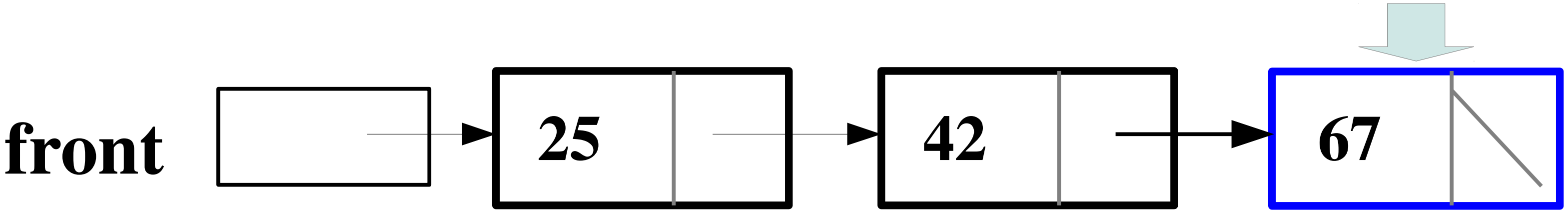
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

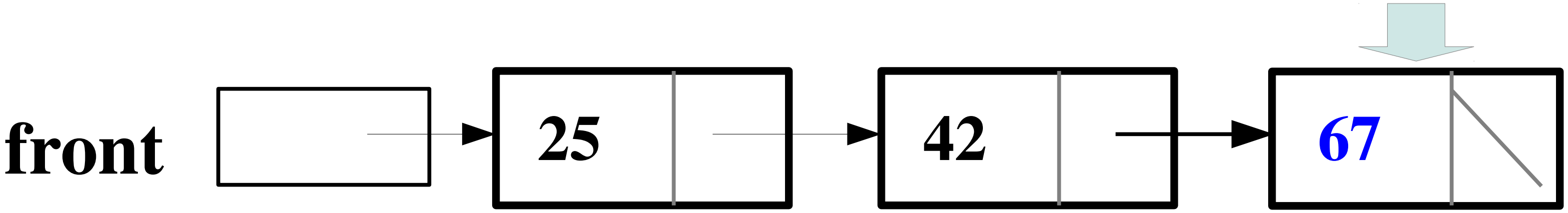
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

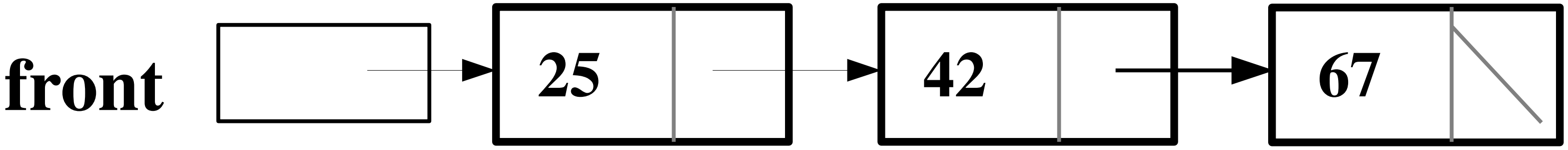
How do you get to the 67?



front . next . next . data

Accessing a Three-Element List

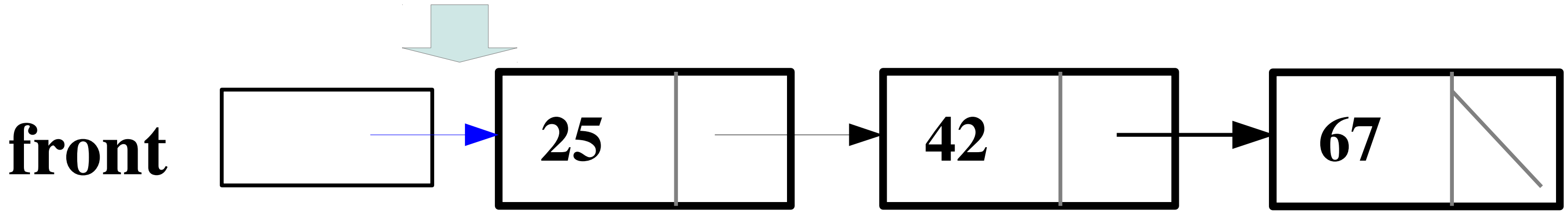
How do you get a reference to the 2nd node?



front . next

Accessing a Three-Element List

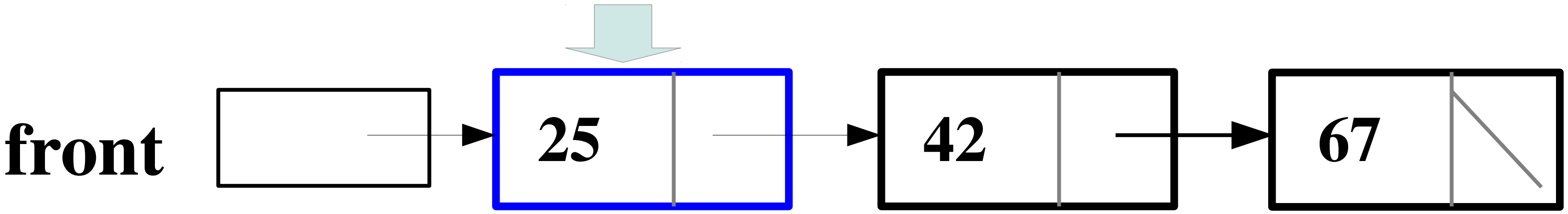
How do you get a reference to the 2nd node?



front . next

Accessing a Three-Element List

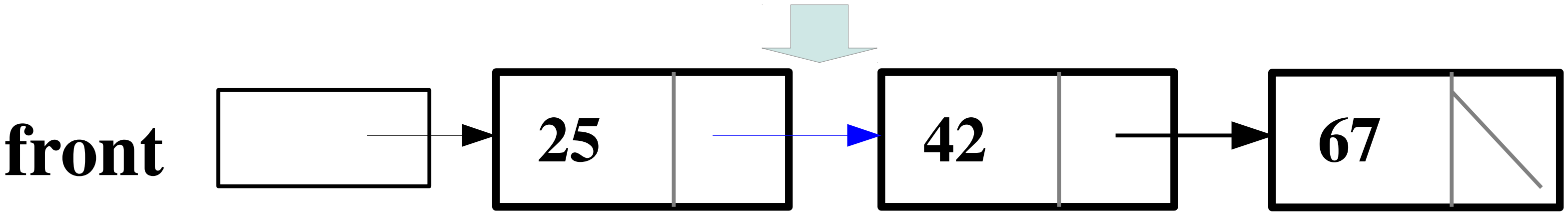
How do you get a reference to the 2nd node?



front . next

Accessing a Three-Element List

How do you get a reference to the 2nd node?



front . next

addAtFront as a Method

```
// does NOT WORK
public static void addAtFront(int data, IntNode front){
    front = new IntNode(data, front);
}
public static void main(String [ ] args){
   IntNode front = null;
   addAtFront(6, front);
   printList(front);   // prints nothing
```

addAtFront as a Method

```
// WORKS
public static IntNode addAtFront(int data, IntNode front){
    front = new IntNode(data, front);
    return front;
}
public static void main(String [ ] args){
   IntNode front = null;
   front = addAtFront(6, front);
   printList(front);   // prints 6
```

Reference Parameters

- **See ParamTest2.java**

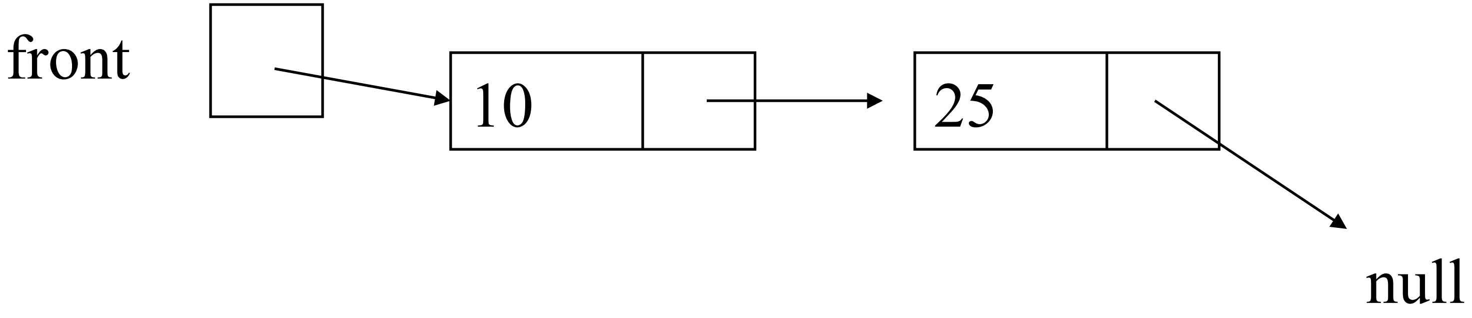
More Methods

- void printList(IntNode front)
- IntNode deleteFront(IntNode front)
- boolean search(IntNode front, int target)
- boolean addAfter(IntNode front,
  int target,
  int item) // false if target
  //   not in list
- IntNode delete (IntNode front, int target)

printList

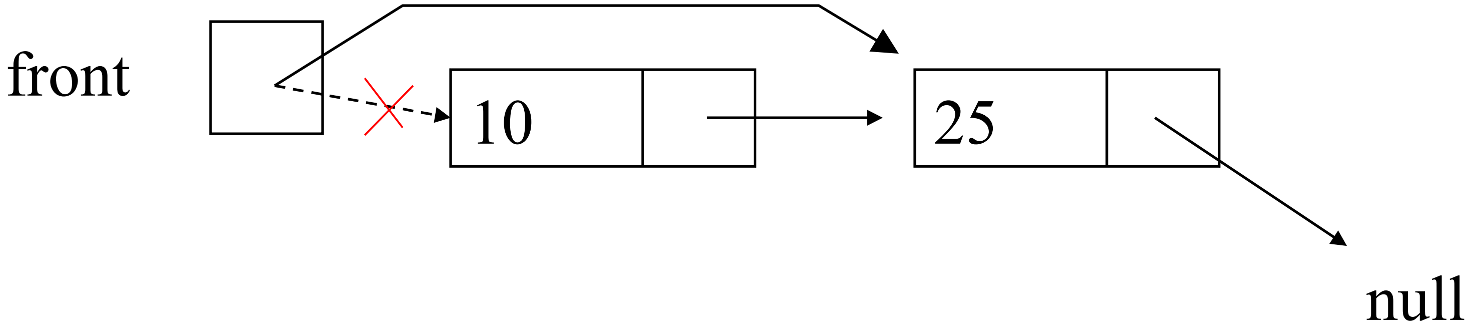
```
public static printList(IntNode front){
        for (IntNode ptr = front; // first node
                ptr != null; // continue if not at null
                ptr = ptr . next){ // go to next node
            System.out.println(ptr . data);
        }
```

Starting Point



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;
}
```

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;
}
```

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;}
```

More methods

- test if two lists are equal
- find last
- append two lists