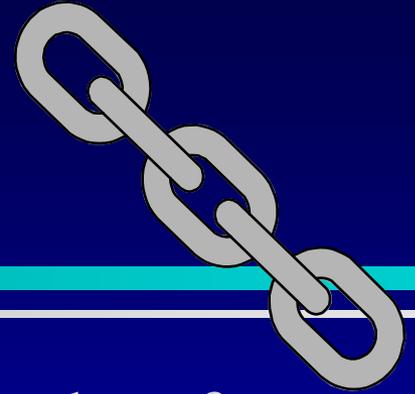
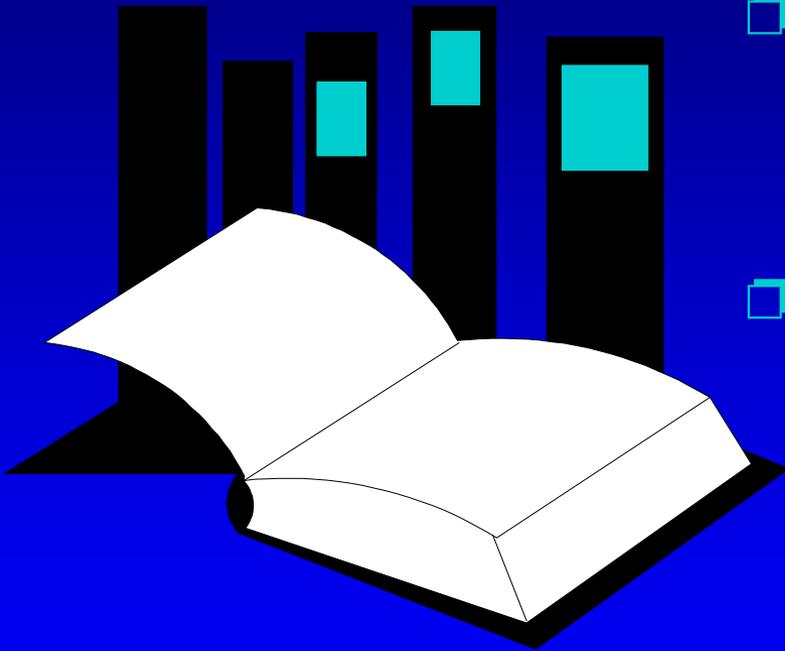


# Linked Lists in Action



- ❑ Chapter 5 introduces the often-used data public classure of linked lists.
- ❑ This presentation shows how to implement the most common operations on linked lists.



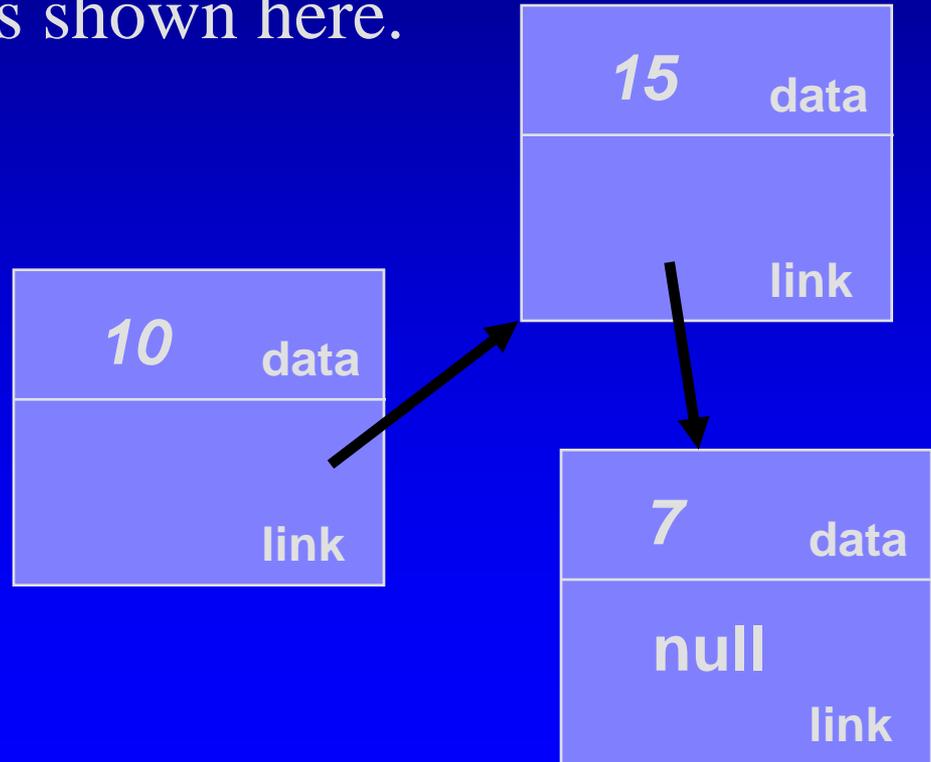
## CHAPTER 4

Data public classures and Other Objects

# Declarations for Linked Lists

- ❑ For this presentation, each node in the linked list is a class, as shown here.

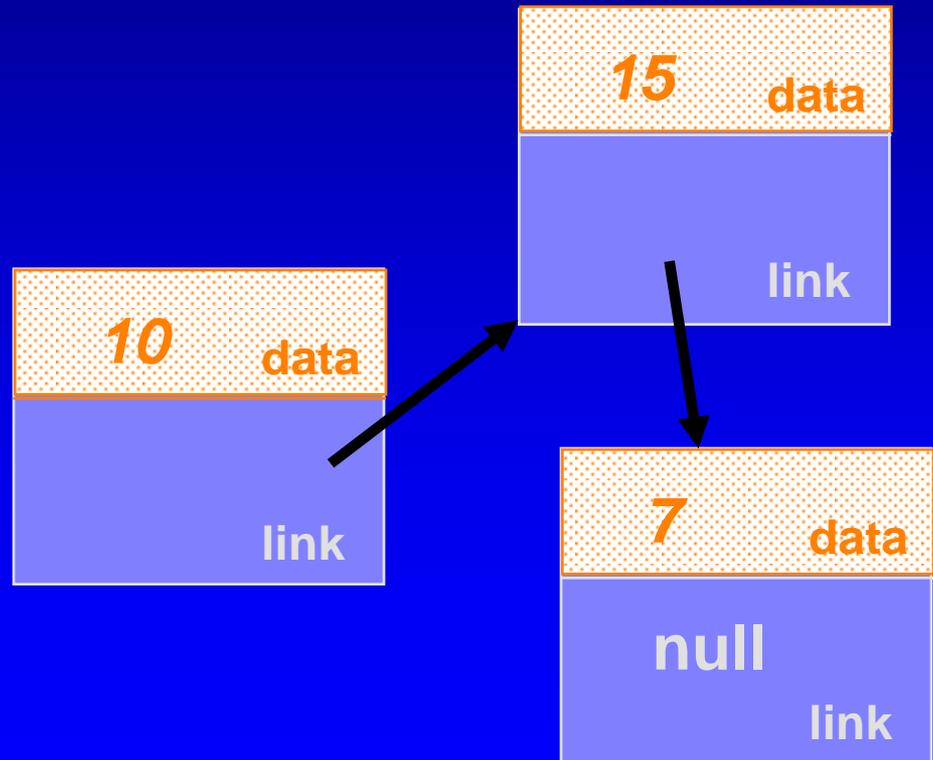
```
public class IntNode
{
    private int data;
    private IntNode link;
    ...
}
```



# Declarations for Linked Lists

- ❑ The data portion of each node is an int.

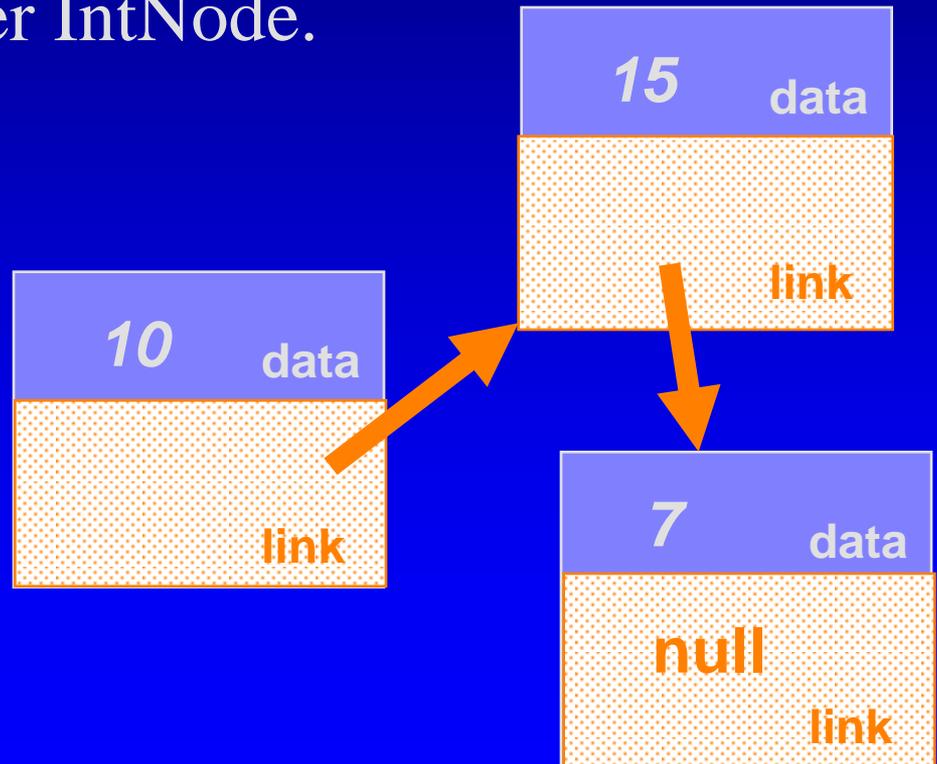
```
public class IntNode
{
    private int data;
    private IntNode link;
    ...
}
```



# Declarations for Linked Lists

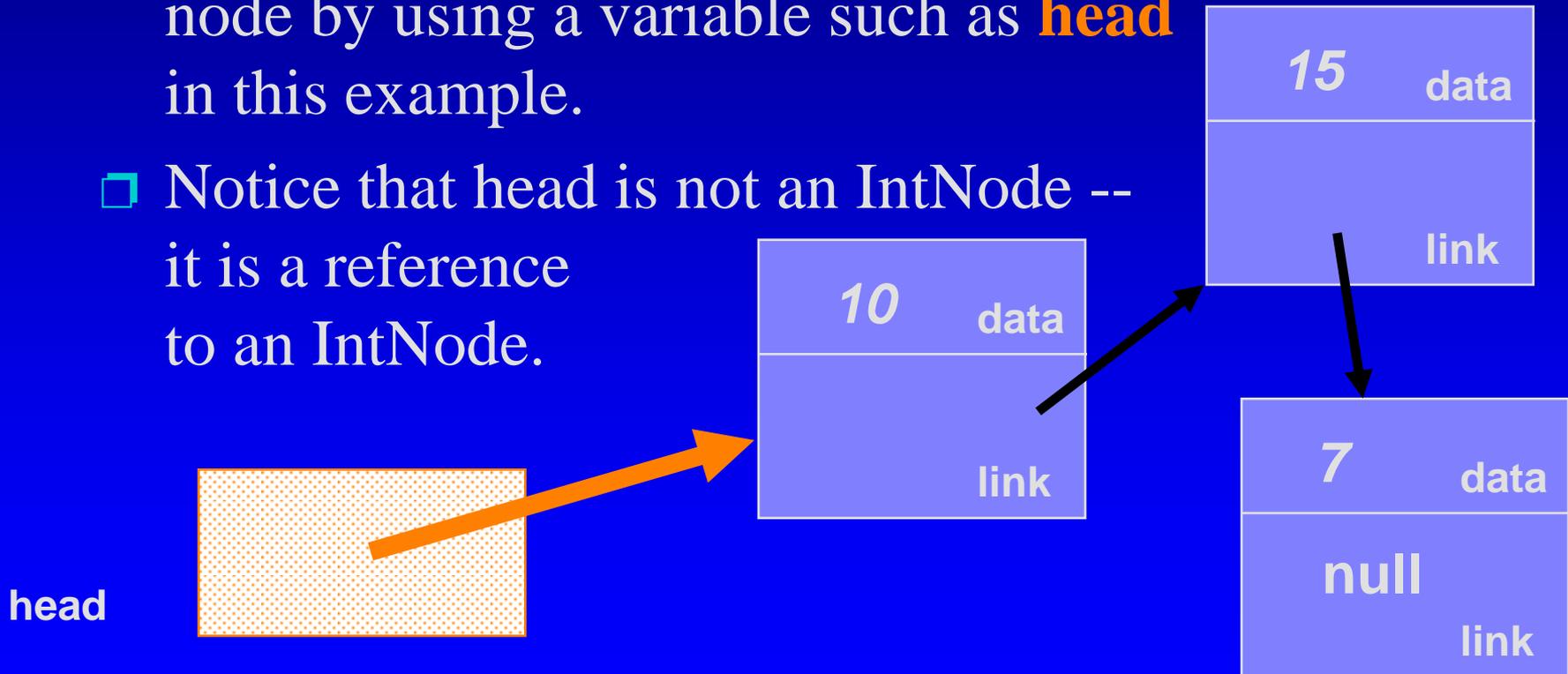
- ❑ Each IntNode also contains a link which refers to another IntNode.

```
public class IntNode
{
    private int data;
    private IntNode link;
    ...
}
```



# Declarations for Linked Lists

- ❑ A program can keep track of the front node by using a variable such as **head** in this example.
- ❑ Notice that head is not an IntNode -- it is a reference to an IntNode.



# Declarations for Linked Lists

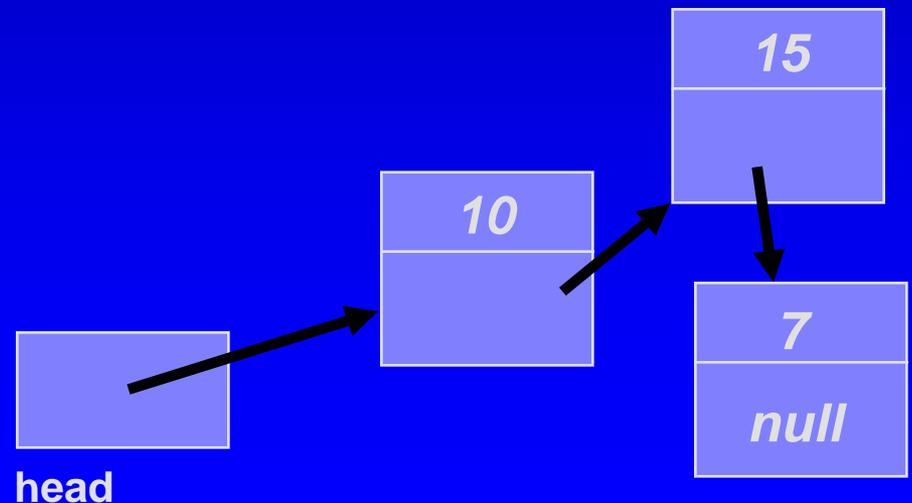
- ❑ A program can keep track of the front node by using an IntNode reference variable such as **head**.
- ❑ Notice that head is not an IntNode -- it is a reference to an IntNode.
- ❑ We represent the empty list by storing **null** in the head reference.

head



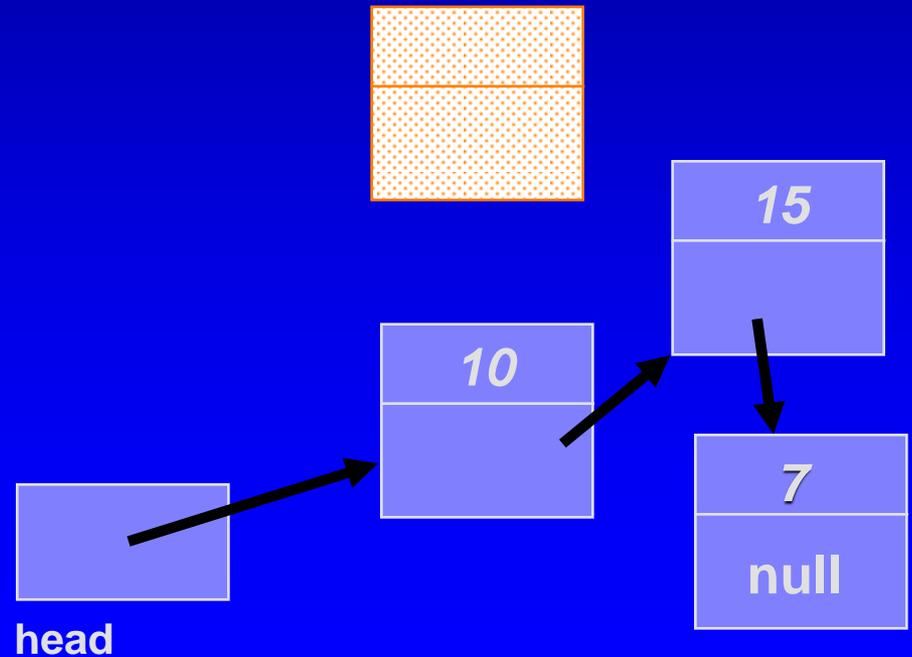
# Inserting an IntNode at the Front

We want to add a new entry, 13, to the **front** of the linked list shown here.



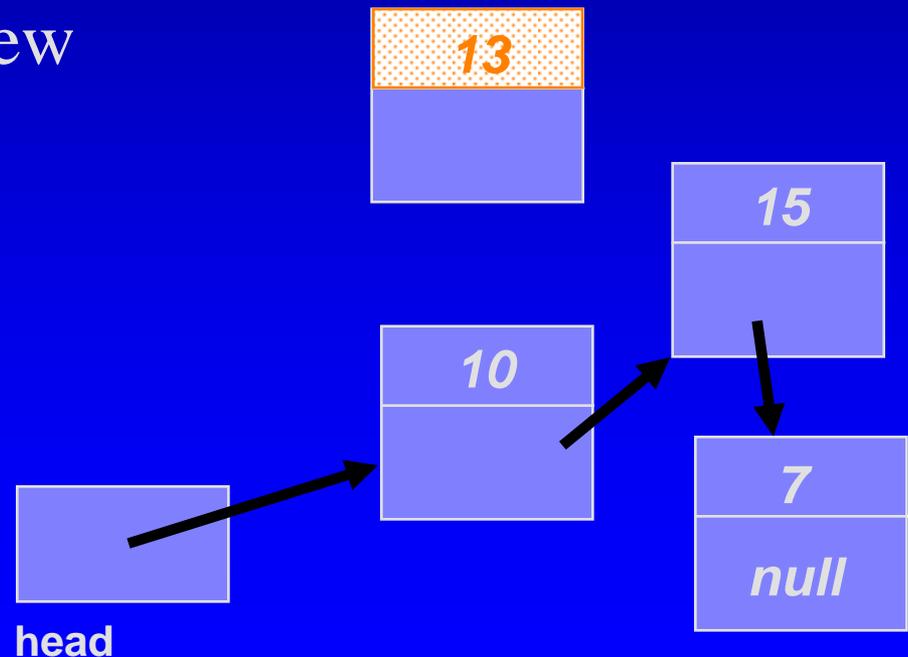
# Inserting an IntNode at the Front

- 1 Create a new node...



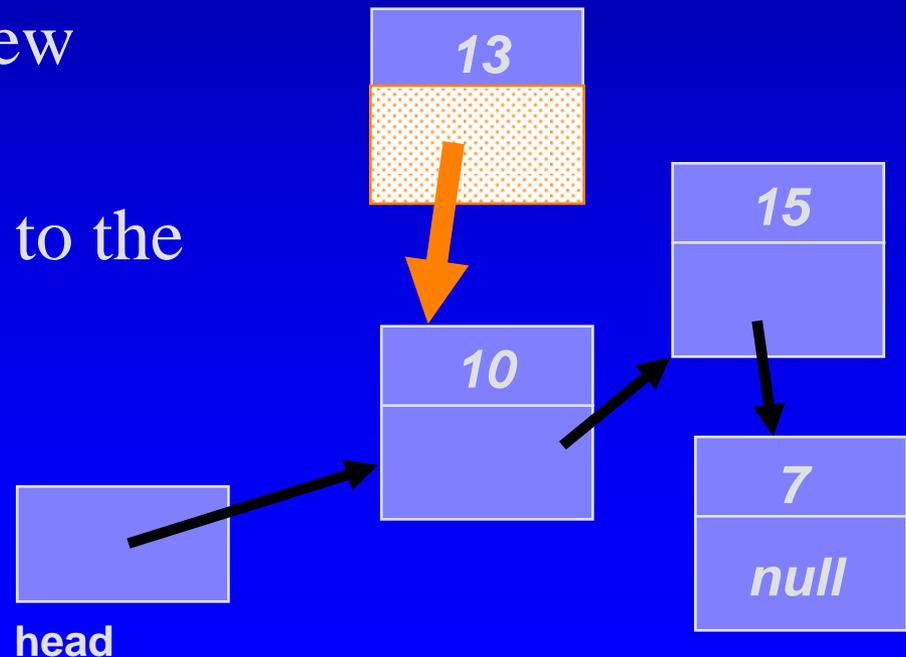
# Inserting an IntNode at the Front

- ① Create a new node...
- ② Place the data in the new node's data field.



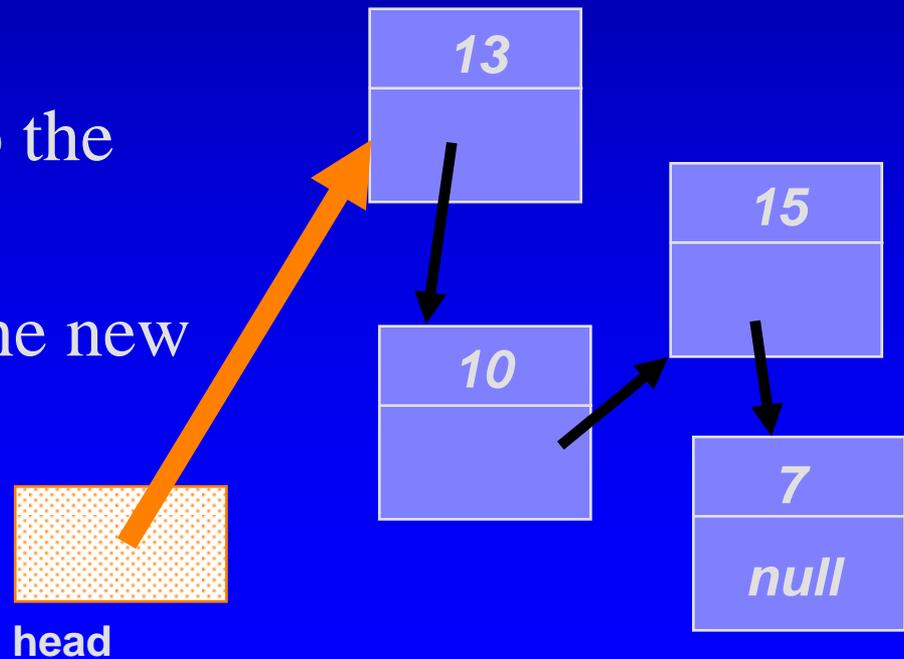
# Inserting an IntNode at the Front

- 1 Create a new node...
- 2 Place the data in the new node's data field...
- 3 Connect the new node to the front of the list.



# Inserting an IntNode at the Front

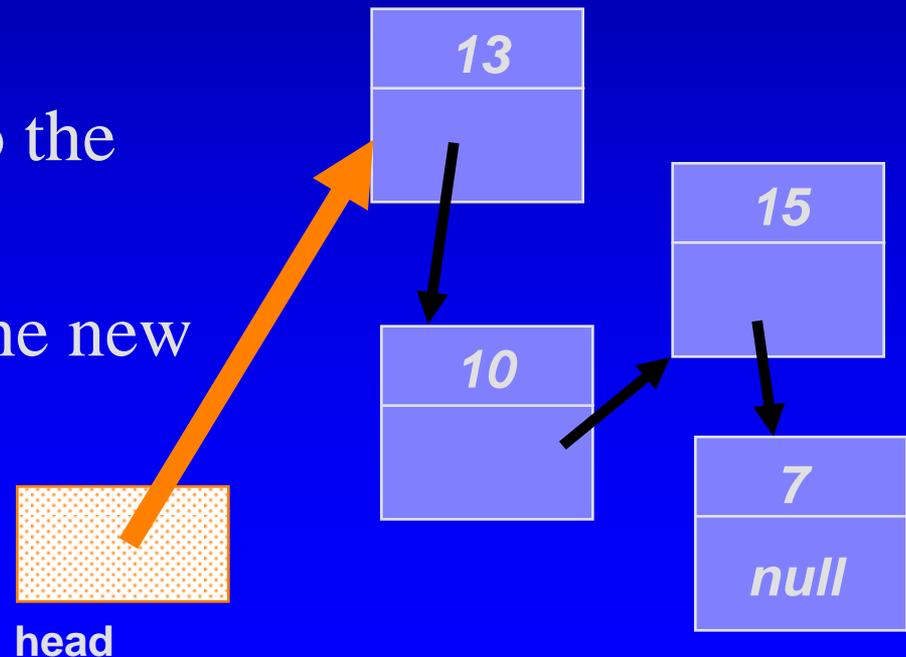
- 1 Create a new node...
- 2 Place the data in the new node's data field....
- 3 Connect the new node to the front of the list.
- 4 Make the head refer to the new head of the linked list.



# Inserting an IntNode at the Front

```
head = new IntNode(13, head);
```

- 1 Create a new node...
- 2 Place the data in the new node's data field....
- 3 Connect the new node to the front of the list.
- 4 Make the head refer to the new head of the linked list.



# Inserting an IntNode at the Front

```
public IntNode(int initialData, IntNode initialLink)
{
    data = initialEntry;
    link = initialLink;
}
```

# Inserting an IntNode at the Front

```
public IntNode(int initialData, IntNode initialLink)
{
    data = initialEntry;
    link = initialLink;
}
```

*Does the constructor work  
correctly for the first  
node on a new list ?*

# Inserting an IntNode at the Front

```
public IntNode(int initialData, IntNode initialLink)
{
    data = initialEntry;
    link = initialLink;
}
```

*Suppose head is null  
and we execute the  
assignment shown here:*

```
head = new IntNode(13, head);
```

*null*

head

# Inserting an IntNode at the Front

```
public IntNode(int initialData, IntNode initialLink)
{
    data = initialEntry;
    link = initialLink;
}
```

```
head = new IntNode(13, head);
```

*null*

head

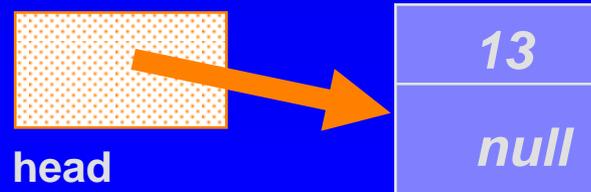
*13*

*null*

# Inserting an IntNode at the Front

```
public IntNode(int initialData, IntNode initialLink)
{
    data = initialEntry;
    link = initialLink;
}
```

```
head = new IntNode(13, head);
```

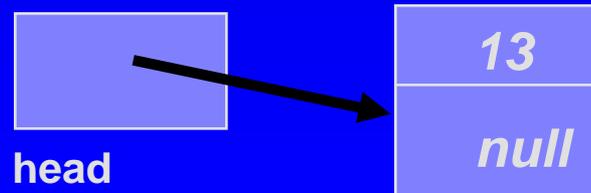


# Inserting an IntNode at the Front

```
public IntNode(int initialData, IntNode initialLink)
{
    data = initialEntry;
    link = initialLink;
}
```

When the statement finishes, the linked list has one node, containing 13.

```
head = new IntNode(13, head);
```



# Caution!

- ❑ Always make sure that your linked list methods work correctly with an empty list.



# Pseudocode for Inserting IntNodes

---

- ❑ IntNodes are often inserted at places other than the front of a linked list.
- ❑ There is a general pseudocode that you can follow for any insertion function. . .

# Pseudocode for Inserting IntNodes

---

- ① Determine whether the new node will be the first node in the linked list. If so, then there is only one step:

```
head = new IntNode(newEntry, head);
```

# Pseudocode for Inserting IntNodes

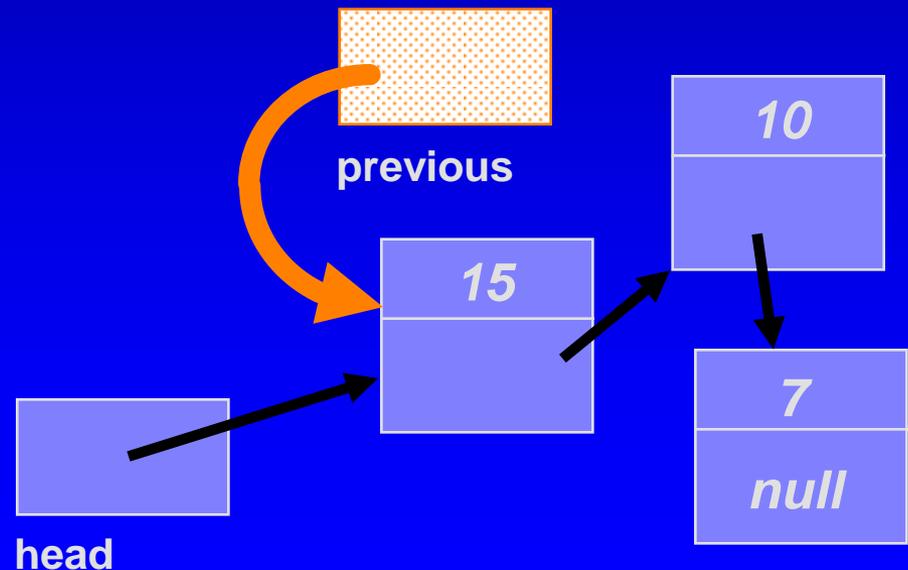
---

- ② Otherwise (if the new node will not be first):
  - Start by setting a reference named **previous** to refer to the node which is just **before** the new node's position.

# Pseudocode for Inserting IntNodes

- ② Otherwise (if the new node will not be first):
  - ❑ Start by setting a reference named **previous** to refer to the node which is just **before** the new node's position.

In this example, the new node will be the second node

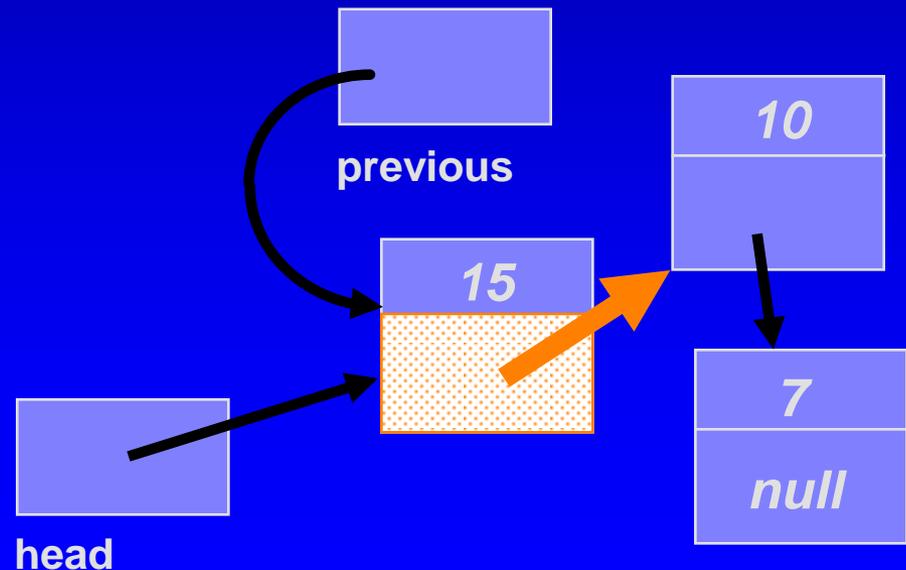


# Pseudocode for Inserting IntNodes

- ② Otherwise (if the new node will not be first):
  - ❑ Start by setting a reference named `previous` to refer to the node which is just before the new node's position

Look at the link which is in the node `previous`

*What is the name of this link?*

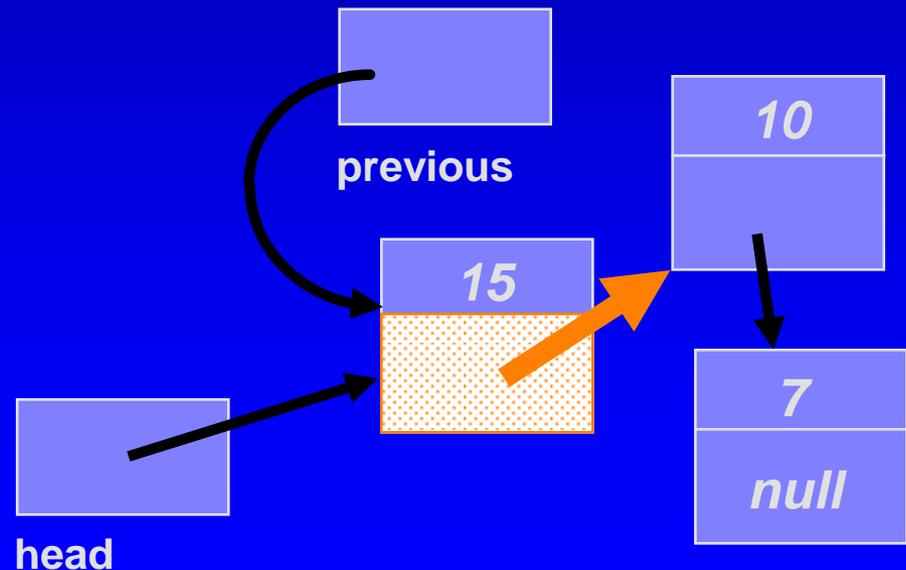


# Pseudocode for Inserting IntNodes

- ② Otherwise (if the new node will not be first):
  - ❑ Start by setting a reference named `previous` to refer to the node which is just before the new node's position

This link is called  
**previous.link**

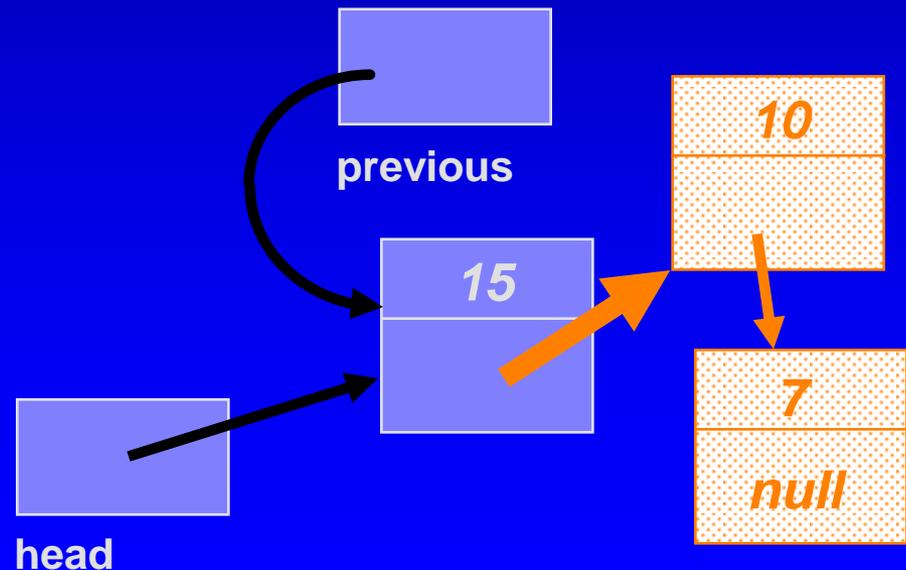
*What is the name of this link?*



# Pseudocode for Inserting IntNodes

- ② Otherwise (if the new node will not be first):
  - ❑ Start by setting a reference named `previous` to refer to the node which is just before the new node's position

**previous.link**  
refers to the head  
of a small linked  
list, with 10 and 7

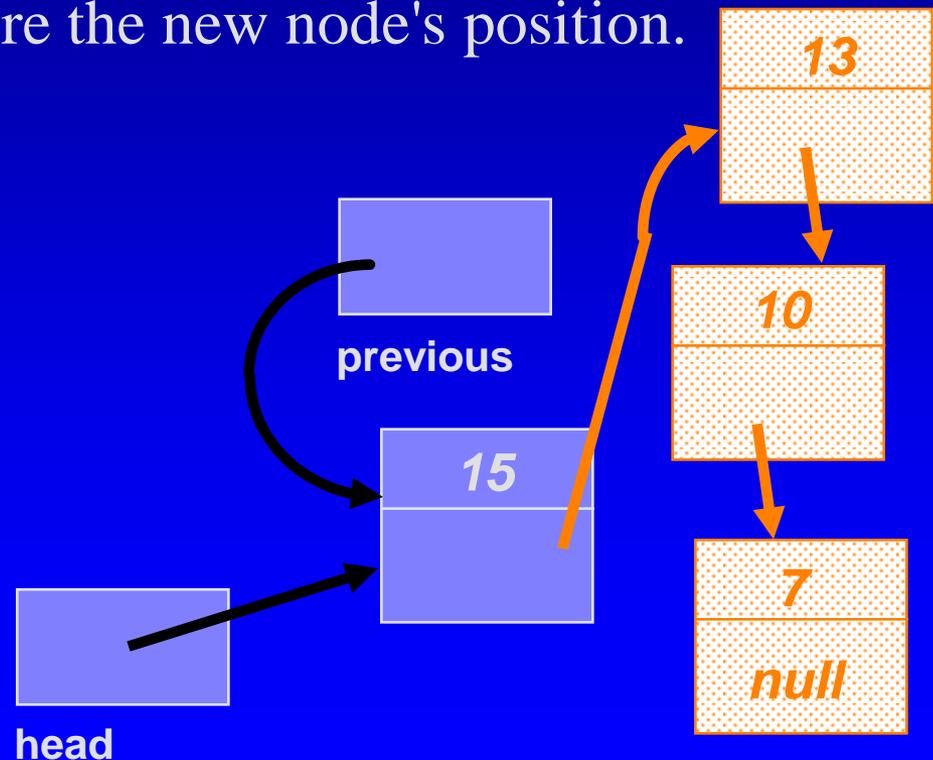


# Pseudocode for Inserting IntNodes

- ② Otherwise (if the new node will not be first):
  - ❑ Start by setting a reference named `previous` to refer to the node which is just before the new node's position.

The new node must be inserted at the front of this small linked list.

*Write one Java statement which will do the insertion.*

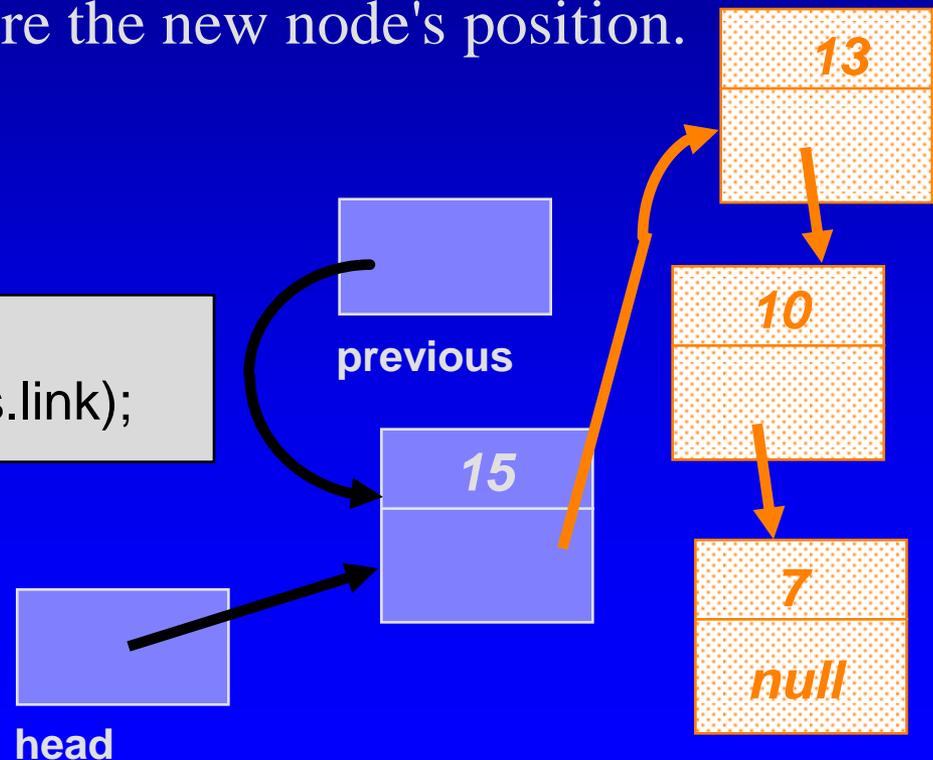


# Pseudocode for Inserting IntNodes

- ② Otherwise (if the new node will not be first):
  - ❑ Start by setting a reference named `previous` to refer to the node which is just before the new node's position.

```
previous.link =  
new IntNode(newEntry, previous.link);
```

*Write one Java statement  
which will do the insertion.*



# Pseudocode for Inserting IntNodes

- ① Determine whether the new node will be the first node in the linked list. If so, then there is only one step:

```
head = new IntNode(newEntry, head);
```

- ② Otherwise (if the new node will not be first):
  - ❑ Set a reference named `previous` to refer to the node which is just before the new node's position.
  - ❑ Execute the step:

```
previous.link =  
    new IntNode(newEntry, previous.link);
```

# Pseudocode for Inserting IntNodes

---

- ❑ The process of adding a new node in the middle of a list can also be incorporated as a separate method. This function is called `addNodeAfter` in the linked list toolkit of Section 4.2.

# Pseudocode for Removing IntNodes

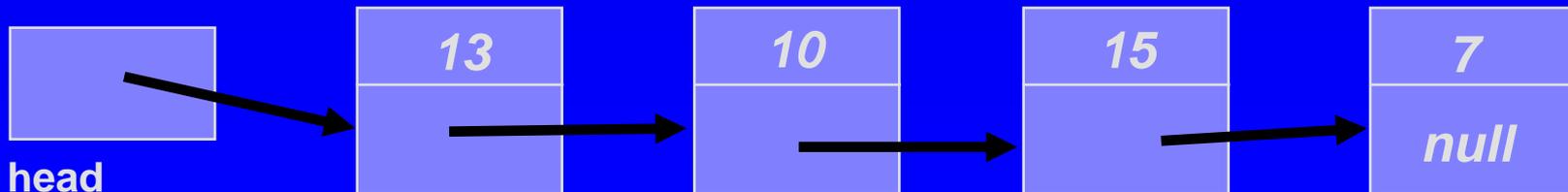
---

- ❑ IntNodes often need to be removed from a linked list.
- ❑ As with insertion, there is a technique for removing a node from the front of a list, and a technique for removing a node from elsewhere.
- ❑ We'll look at the technique for removing a node from the front of a linked list.

# Removing the Head IntNode

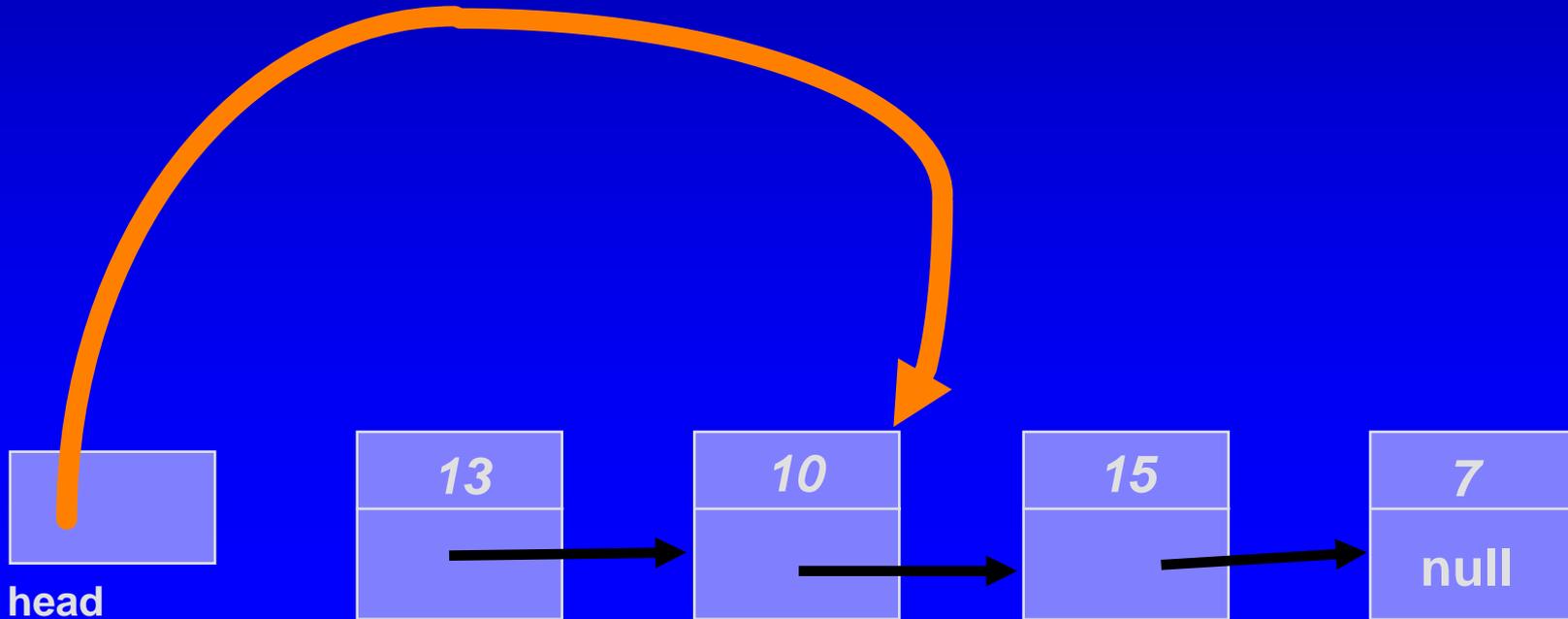
```
head = head.link;
```

*Draw the change that this statement will make to the linked list.*



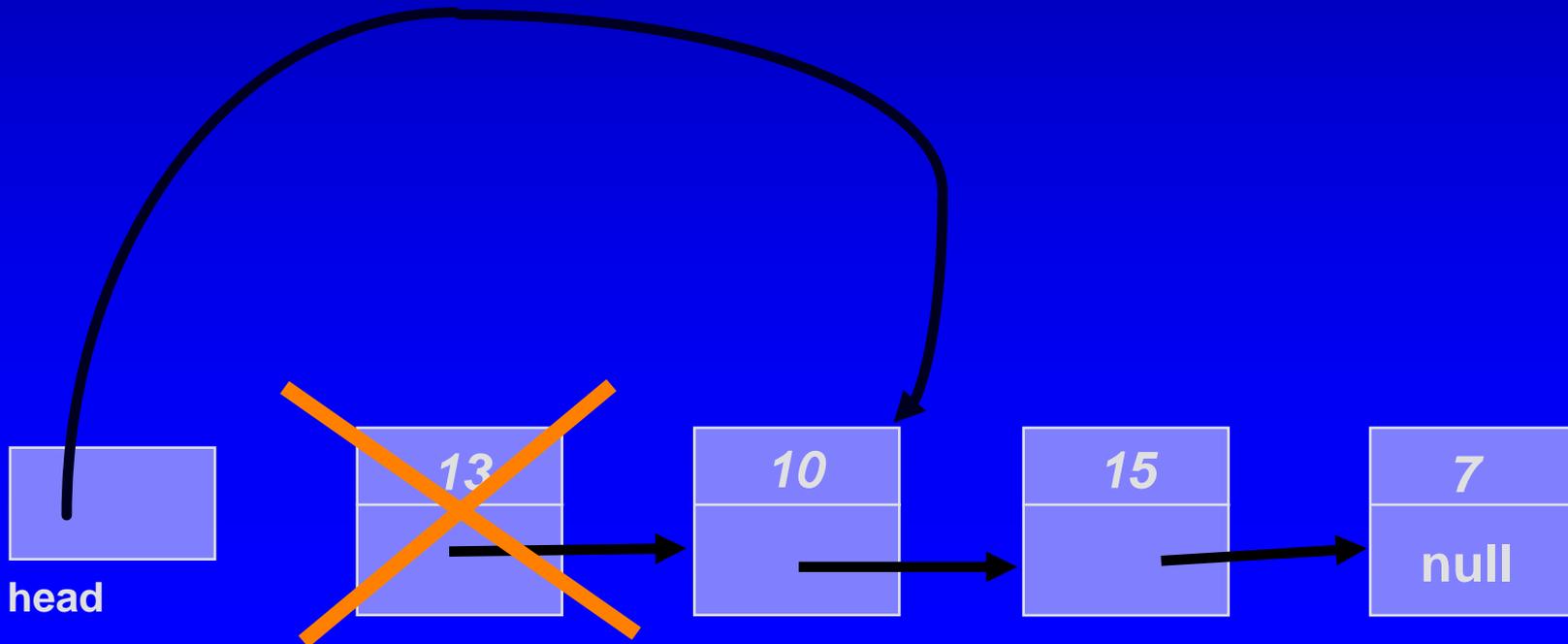
# Removing the Head IntNode

```
head = head.link;
```



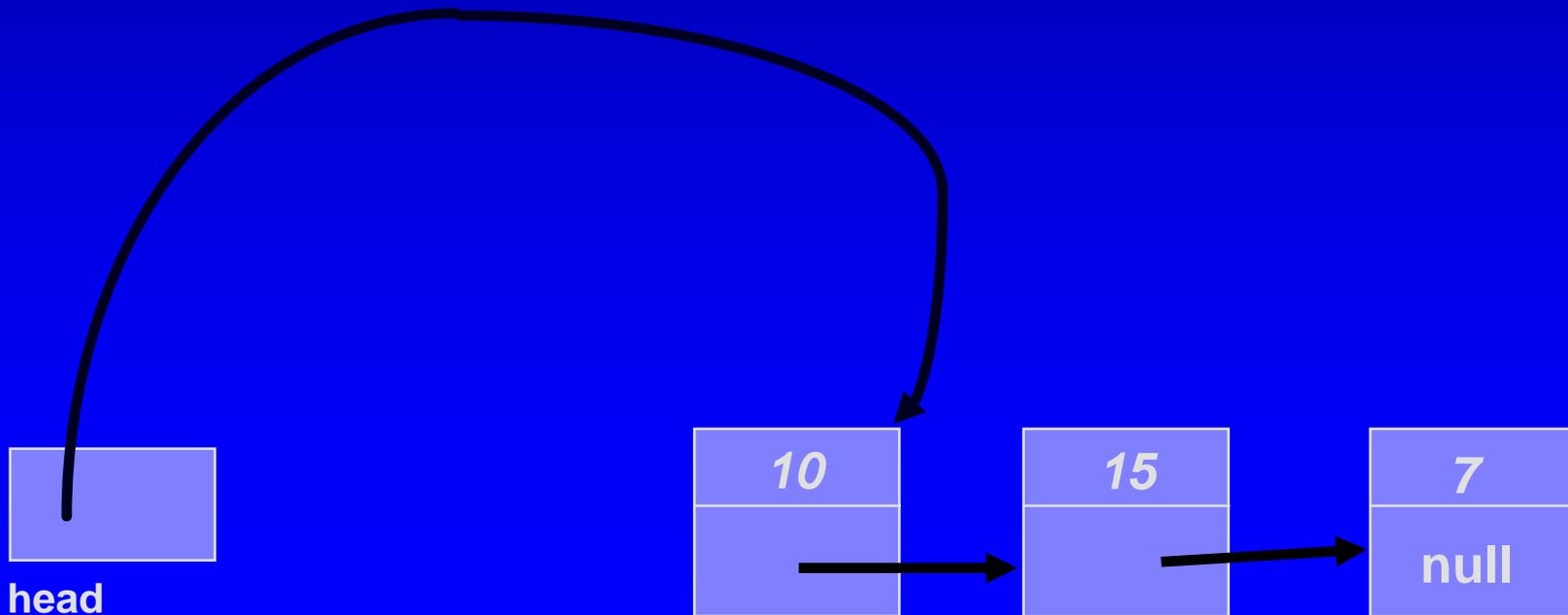
# Removing the Head IntNode

```
head = head.link;
```



# Removing the Head IntNode

Here's what the linked list looks like after the removal finishes.





# Summary

---

- ❑ It is easy to insert or remove a node at the front of a list.
- ❑ You also need a technique for inserting or removing a node elsewhere

Presentation copyright 2012, Pearson Education,  
For use with *Data public classsures and Other Objects Using Java*  
by Michael Main.

Some artwork in the presentation is used with permission from Presentation Task Force  
(copyright New Vision Technologies Inc) and Corel Gallery Clipart Catalog (copyright Corel Corporation, 3G Graphics Inc, Archive Arts, Cartesia Software, Image Club Graphics Inc, One Mile Up Inc, TechPool Studios, Totem Graphics Inc).

Students and in public classsors who use *Data public classsures and Other Objects Using Java* are welcome  
to use this presentation however they see fit, so long as this copyright notice remains intact.

