Linked Lists - Search,Deletion and Insertion

- Reading: Savitch, Chapter10

Objectives

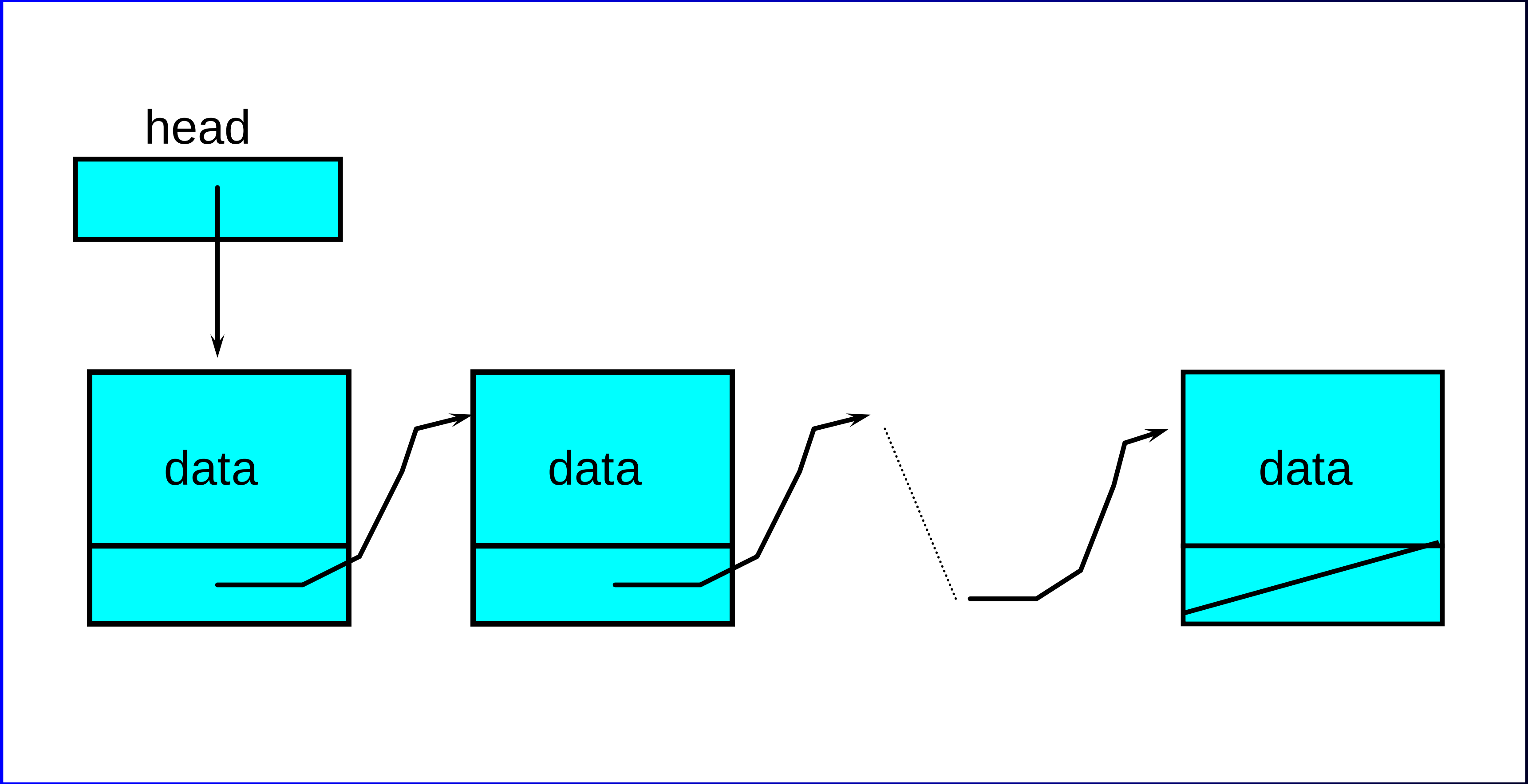
- To study the basics of linked lists
- To learn how to:
  - search for a specified node in a linked list
  - delete nodes from a linked list
  - insert nodes at any position in a linked list

Linked List

- What is a linked list?

  a variable-length collection of objects (of the same class). Each object is called a node of the linked list. Each node contains a reference to the next node.

A linked list



JAVA declaration (general)

```
class Node {
   private DataType data;
   private Node next;
   public Node(DataType _d) {
         data = _d;
         next = null
   }
   … …
}

// Version 1
```

What other methods would be required in this class?

Hint: LinkedList class needs access to the private fields of the Node class

JAVA declaration (general)

```
class LinkedList {
   private Node head = null;
   public void insert (Node p) { ... ...}
   public void remove (Node p) {... ...}
   public Node search(DataType value) {... ...}
   public void traversal() {... ...}
   ... ...
}

// Version 1
```

JAVA declaration (inner class version)

```
class LinkedList {
   private Node head = null;
   public void insert (Node p) { … …}
   public void remove (Node p) {… …}
   public Node search(DataType value) {… …}
   public void traversal() {… …}
   … …
   private class Node {
     private DataType data;
     private Node next;
     public Node(DataType _d) {
             data = _d;
             next = null
     }
     … …
   }                     // Version 2
}
```

Traversing a linked list

```
public void traversal() {

        Node p = head;

        while (p != null) {
            process (p.data);
            p = p.next;
        }
}
```

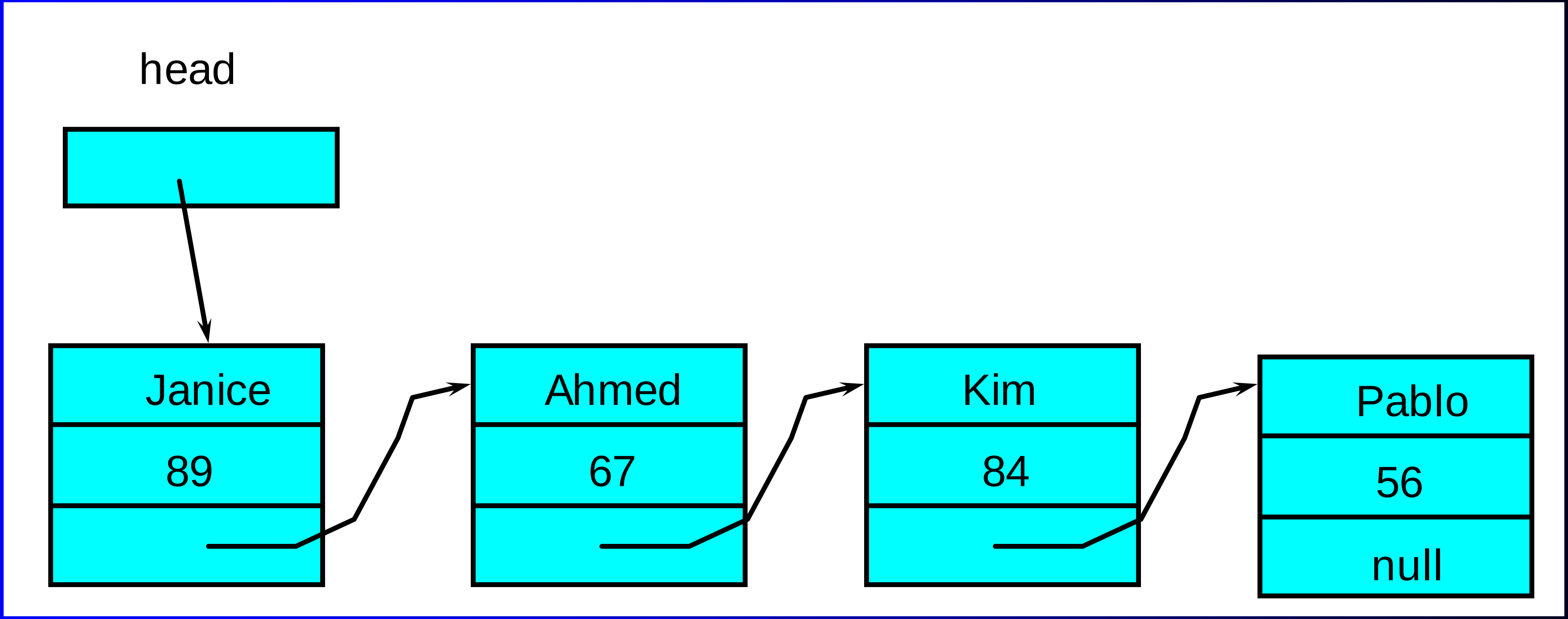
Using version 1 or 2 code?

Searching a linked list

- A common activity is to search a linked list to find a particular node
- Traversal, but not necessarily of the whole list

Searching a linked list (ctd)

- Example
  - –in our student marks list, find Kim’s mark



Searching a linked list (ctd)

- Design
  - search until “Kim” is found or the end of the list is reached
  - if “Kim” was found, print her mark

Searching a linked list (ctd)

- What is wrong with the following solution?

```
StudentNode p = head;

while ((p. name).compareTo(“Kim”) != 0) && (p != null))
  p = p.next;

if (p == null)
  System.out.println(“Kim: not found in list”);
else
  System.out.println(“Kim’s mark is ” + p.mark);
```

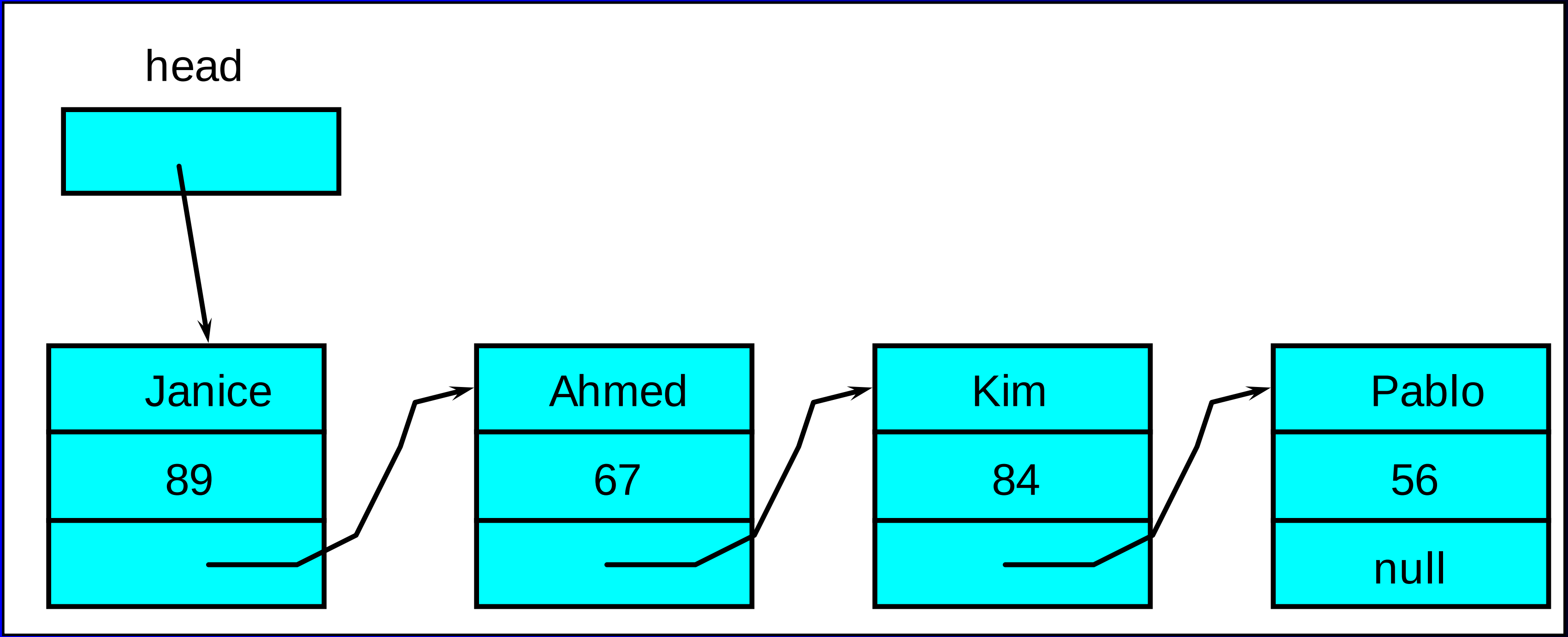
Searching a linked list (ctd)

- the correct code

```
StudentNode p = head;

while ((p != null) && ((p. name).compareTo(“Kim”) != 0)
  p = p.next;

if (p == null)
  System.out.println(“Kim: not found in list”);
else
  System.out.println(“Kim’s mark is ” + p.mark);
```

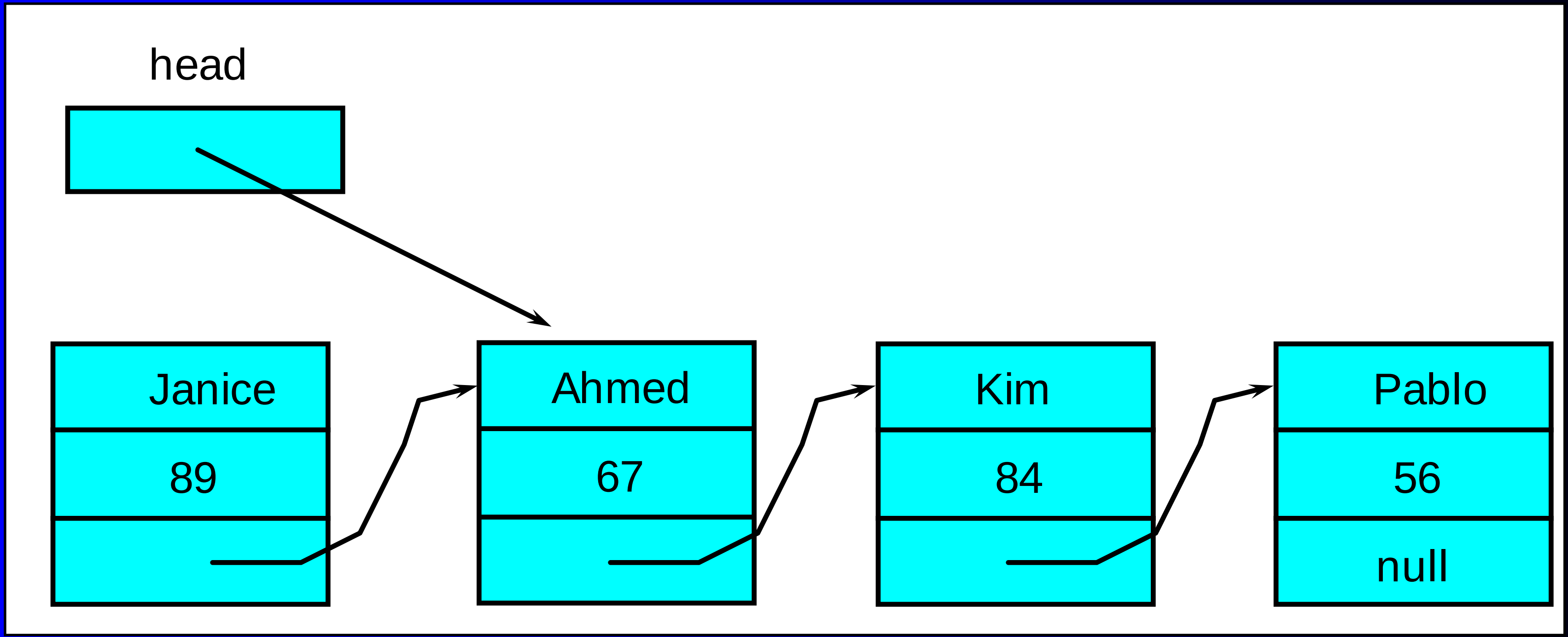
Losing nodes

- What happens below when the statement head = head.next; is executed?



Losing nodes (ctd)

- What happens below when the statement head = head.next; is executed?
- The Janice node is now an orphan and is lost.



Deletion

- Deleting the first node
  - easy
- Deleting after a specified node
  - method prototype

```
void remove(StudentNode after_me);
```

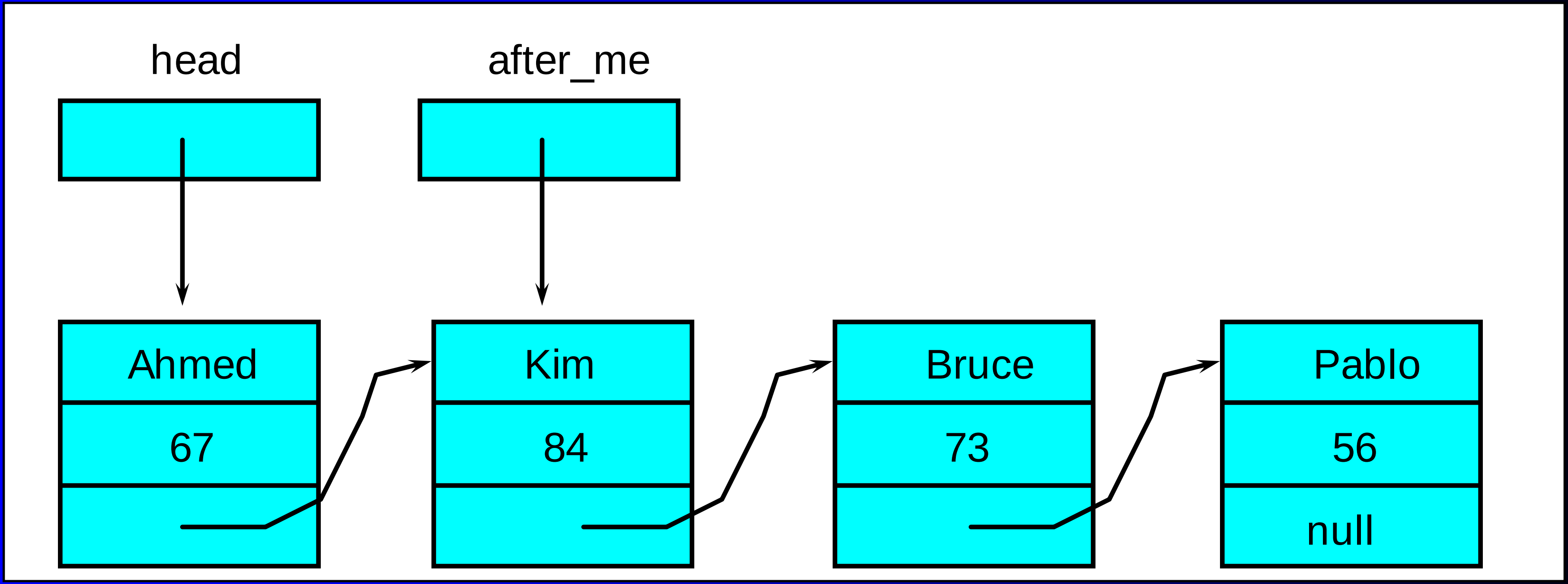
Deletion (ctd)

- JAVA source code

```
public void remove(StudentNode after_me) {
  StudentNode p;

  p = after_me.next;
  after_me.next = p.next;
}
```

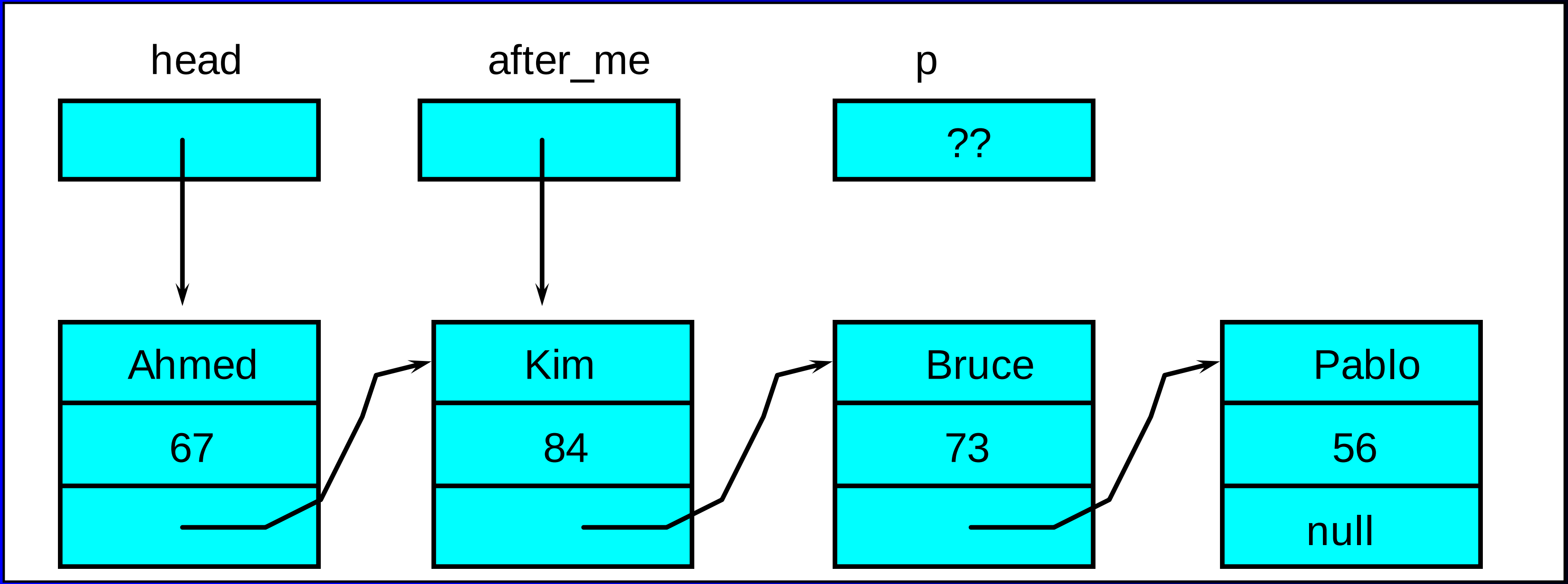
Deletion (ctc

```
public void remove(StudentNode after_me) {
  StudentNode p;

  p = after_me.next;
  after_me.next = p.next;
}
```



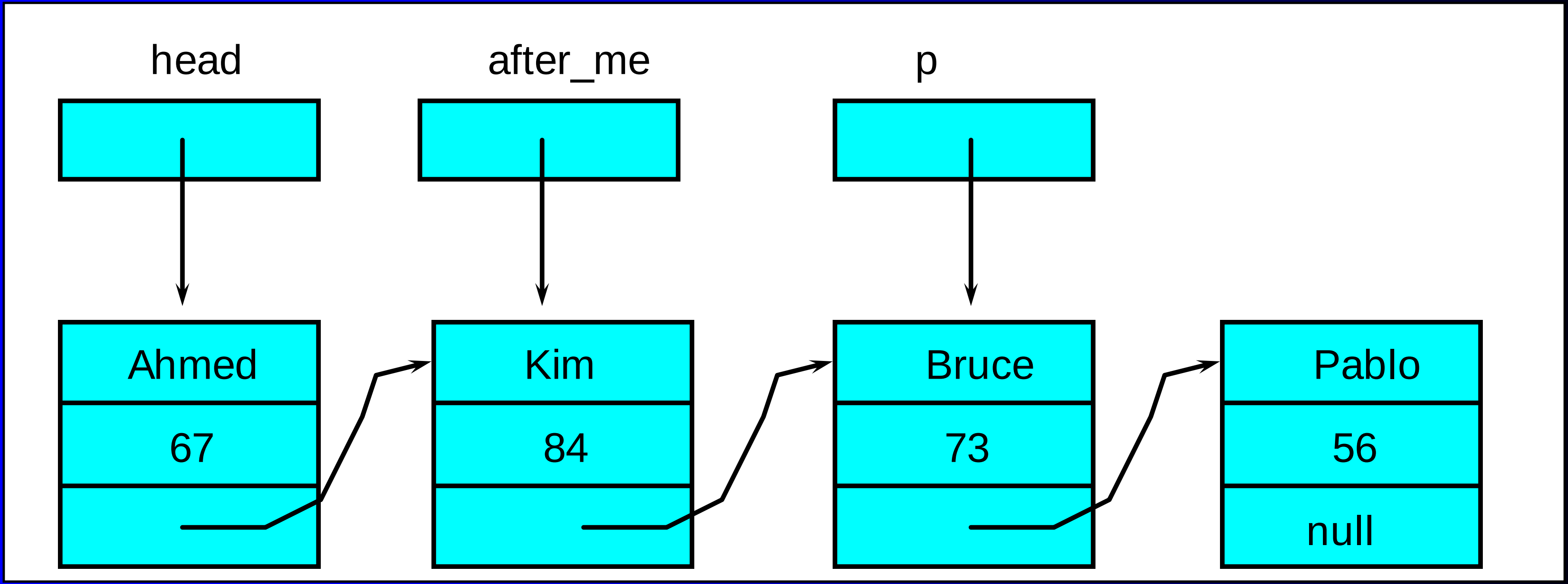
Deletion (ctd)

```
public void remove(StudentNode after_me) {
  StudentNode p;

  p = after_me.next;
  after_me.next = p.next;
}
```



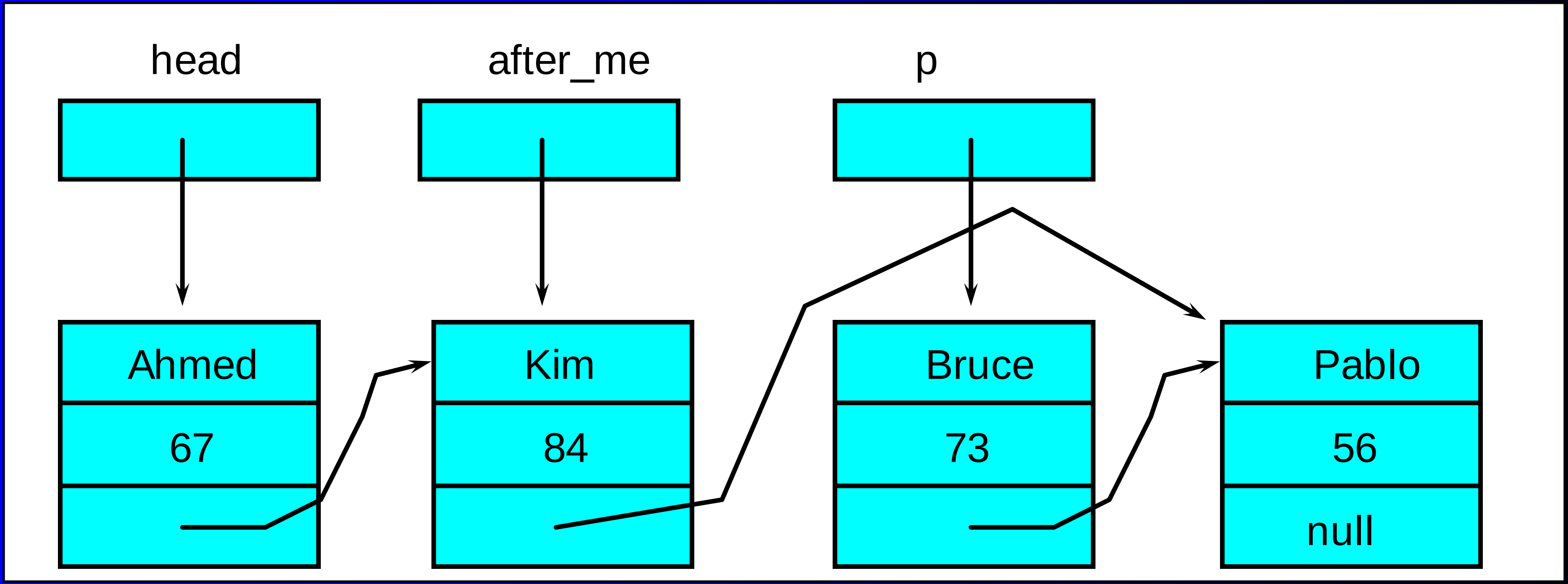
Deletion (ctd)

```
public void remove(StudentNode after_me) {
  StudentNode p;

  p = after_me.next;
  after_me.next = p.next;
}
```



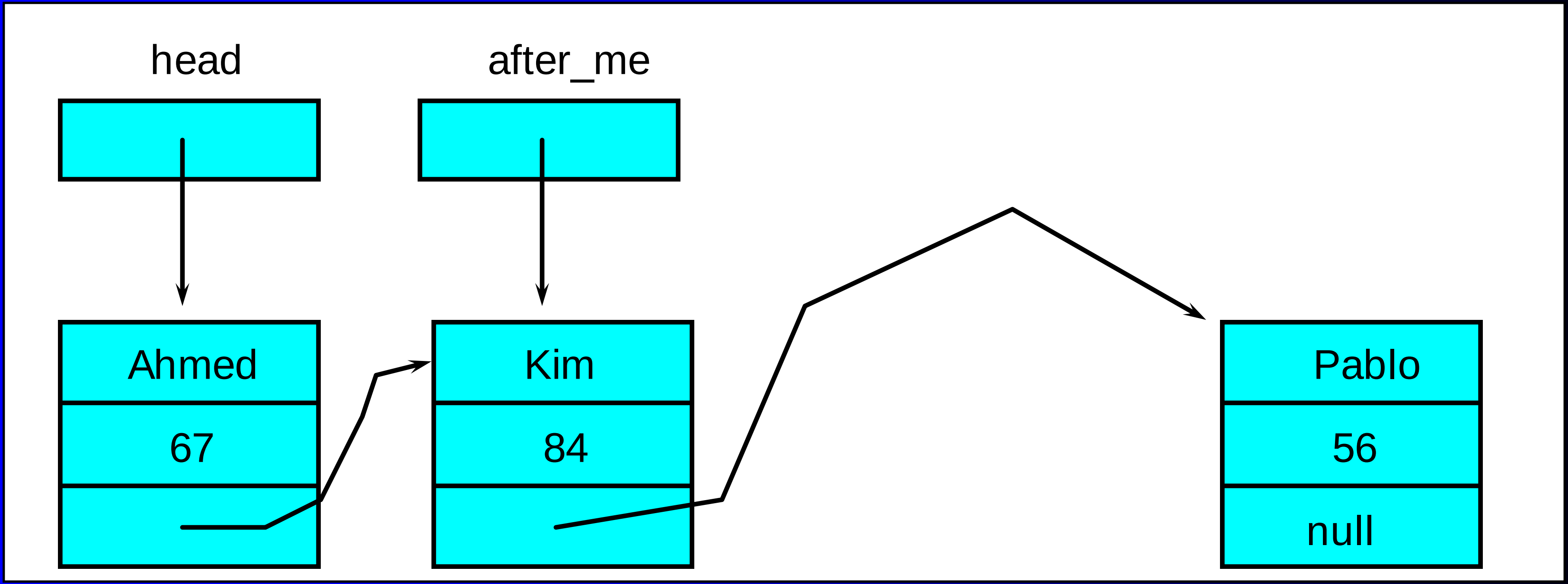
Deletion (ctd)

```
public void remove(StudentNode after_me) {
  StudentNode p;

  p = after_me.next;
  after_me.next = p.next;
}
```



Deletion (ctd)

```
public void  remove(StudentNode after_me) {
  StudentNode p;

  p = after_me.next;
  after_me.next = p.next;
}
```



Class exercise: deletion

- Write JAVA code to remove the first element in a linked list

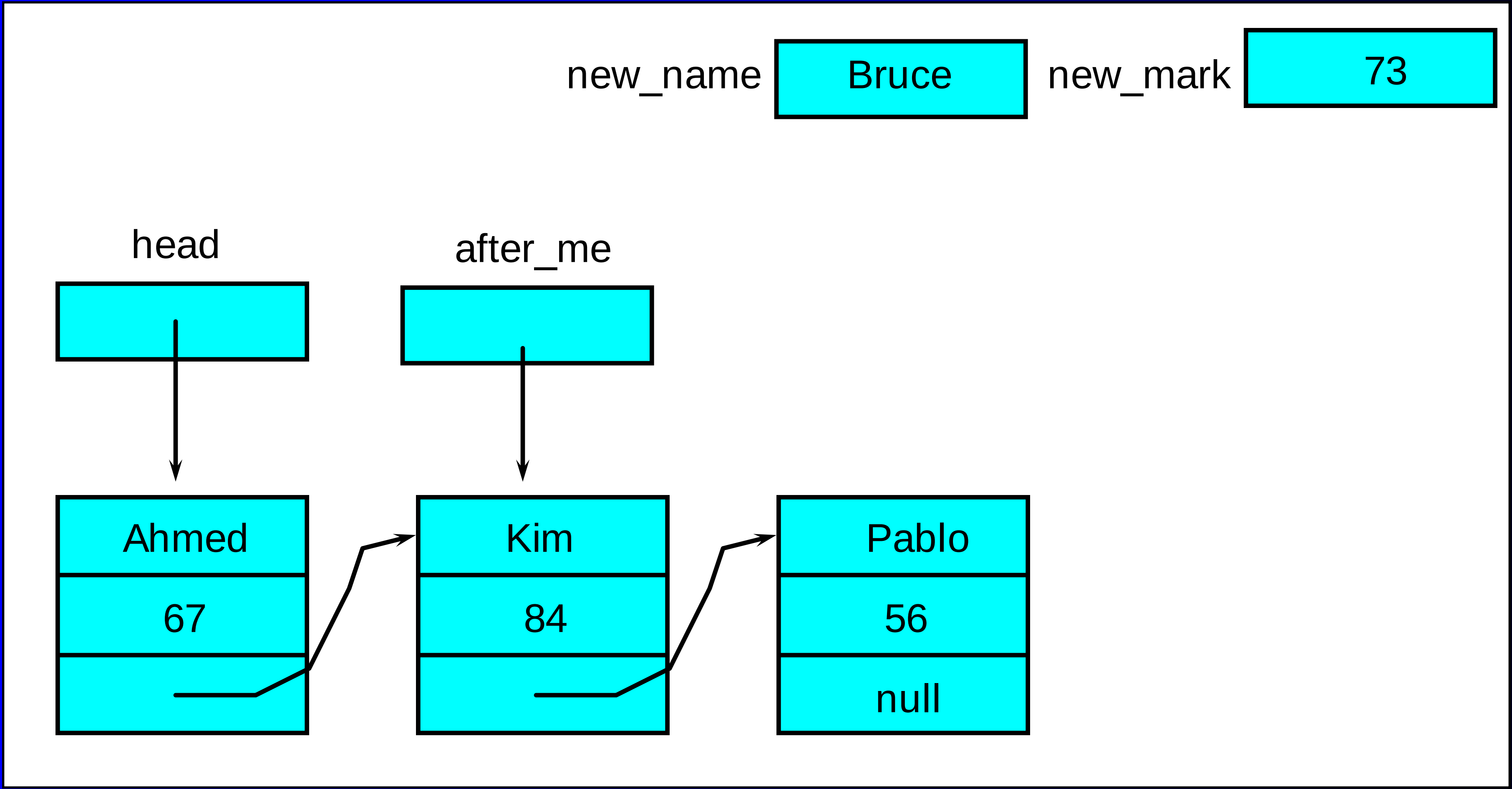
Insertion

- Last lecture
  - insertion at front of list
- This lecture
  - insertion after a specified node
  - method prototype

```
void insert(StudentNode after_me,
                 String new_name, int new_mark);
```

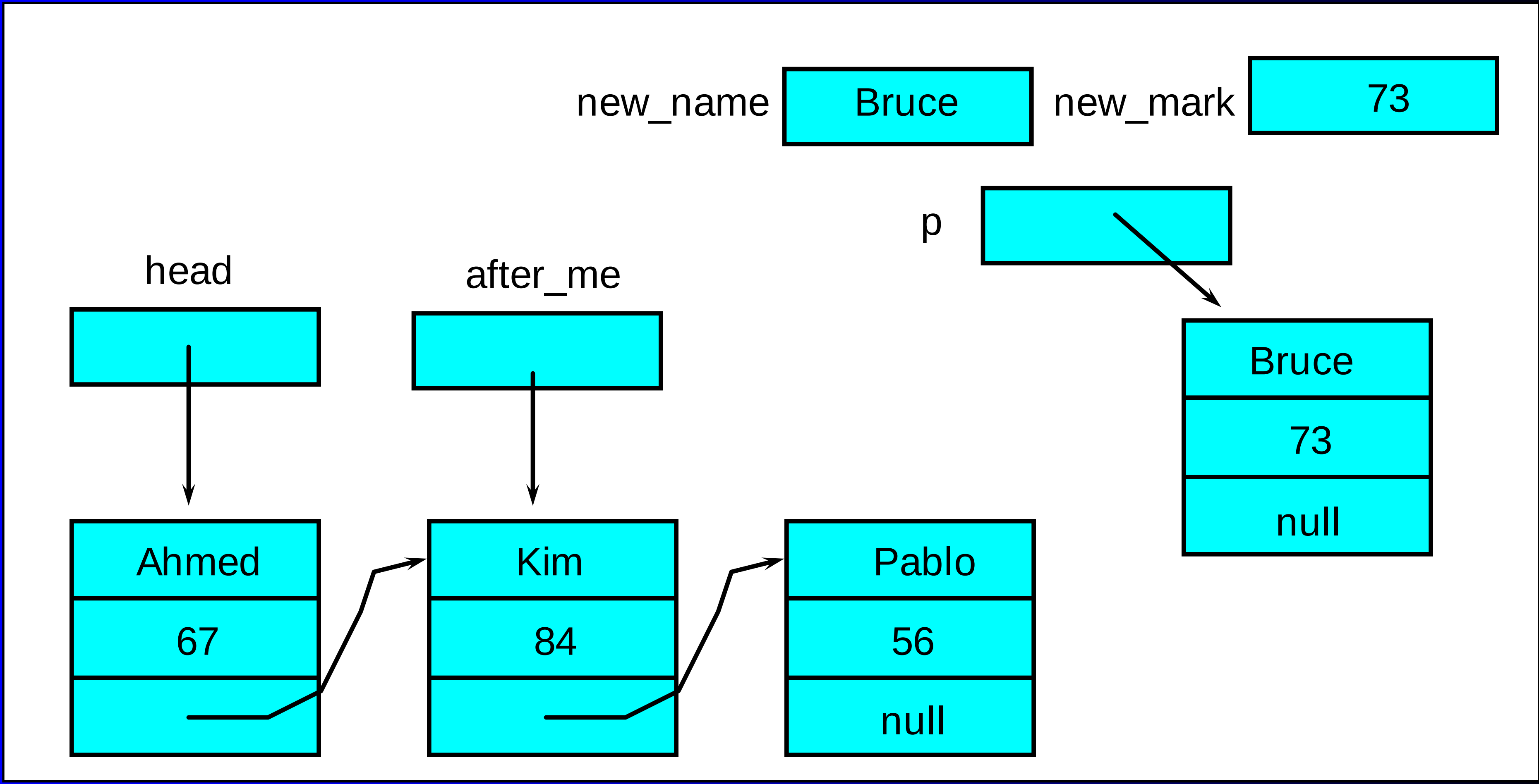
Insertion (ctd)

```
void insert(StudentNode after_me, String new_name, int new_mark);
```



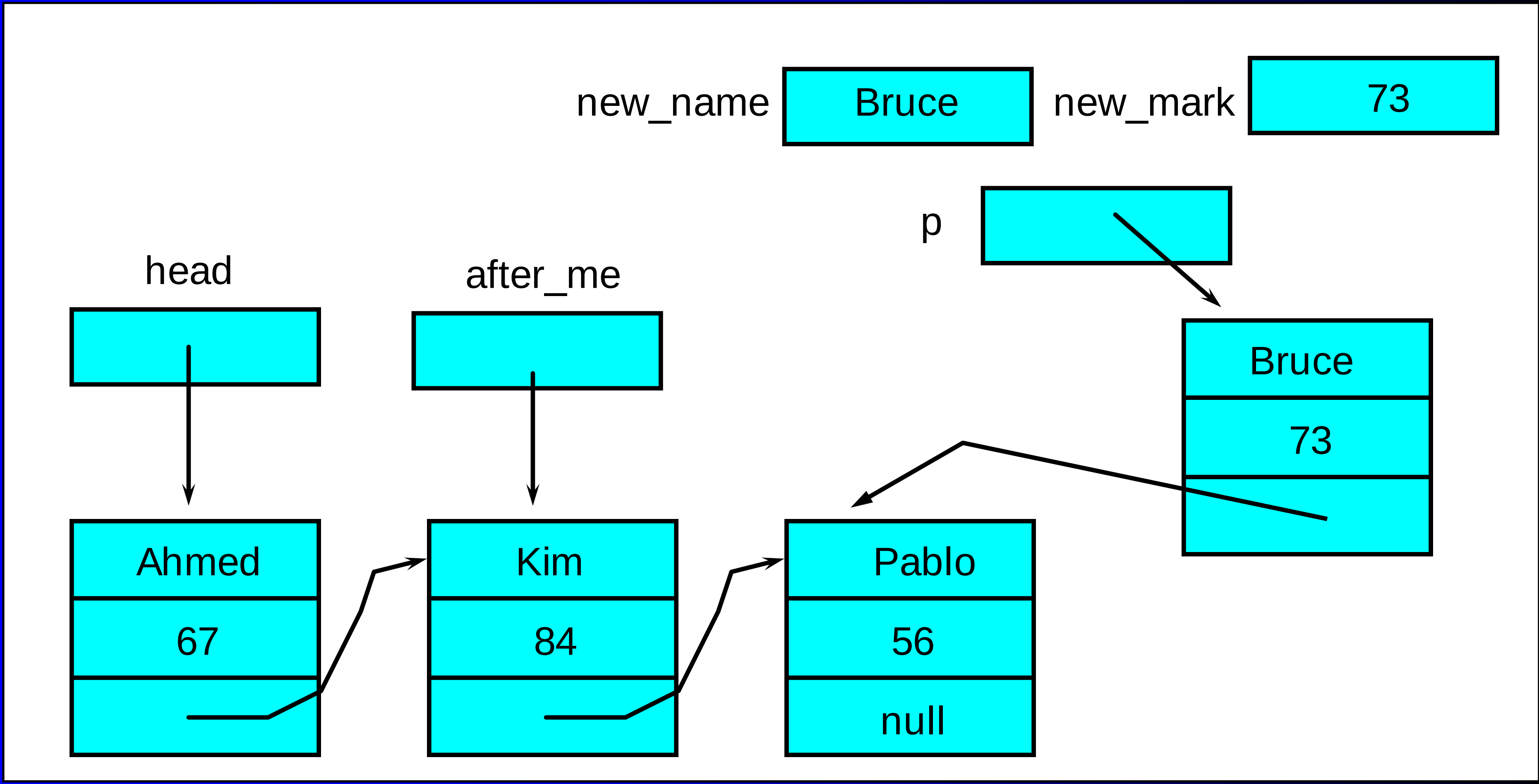
Insertion (ctd)

```
void insert(StudentNode after_me, String new_name, int new_mark);
```



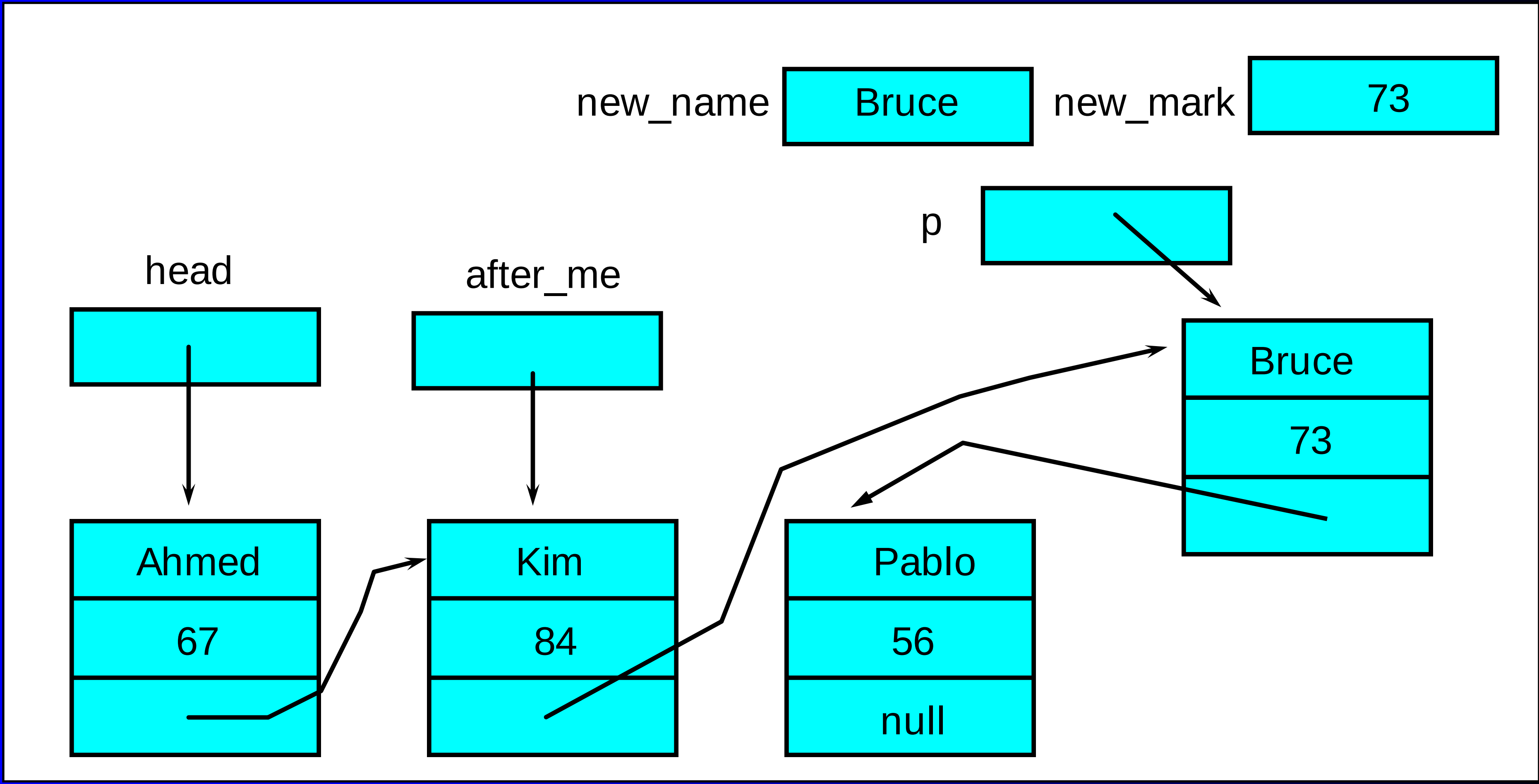
Insertion (ctd)

```
void insert(StudentNode after_me, String new_name, int new_mark);
```



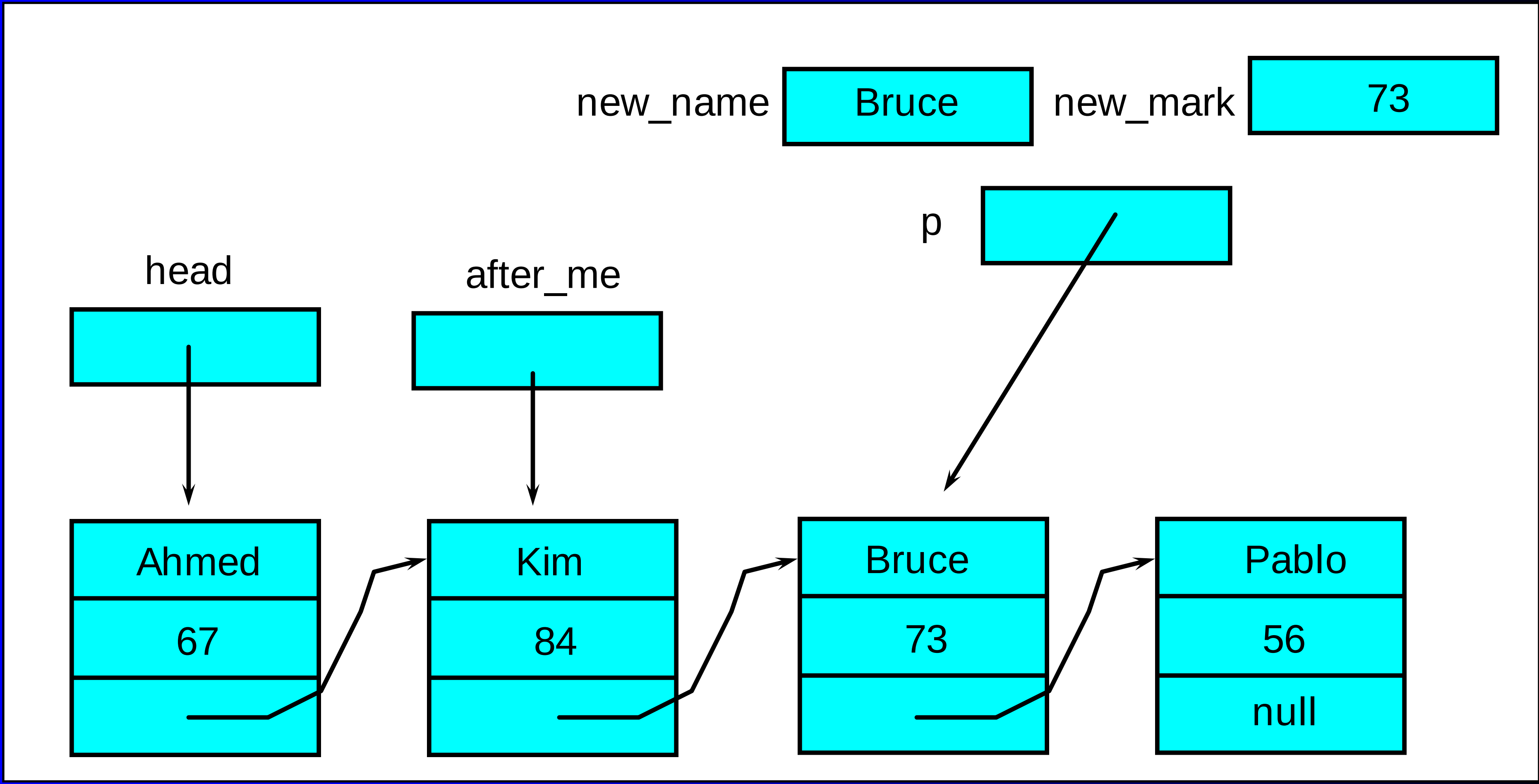
Insertion (ctd)

```
void insert(StudentNode after_me, String new_name, int new_mark);
```



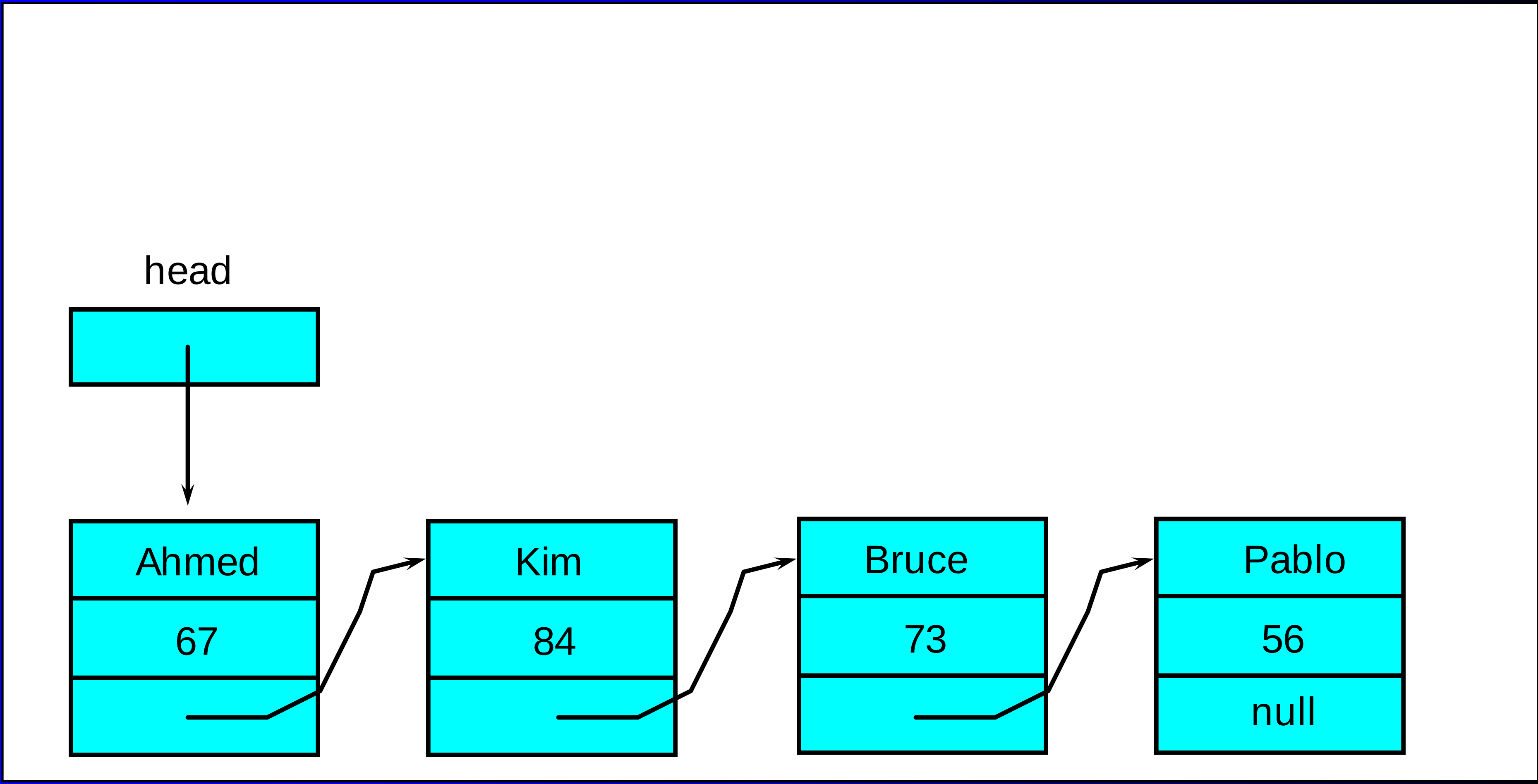
Insertion (ctd)

```
void insert(StudentNode after_me, String new_name, int new_mark);
```



Insertion (ctd)

```
void insert(StudentNode after_me, String new_name, int new_mark);
```



Insertion (ctd)

- Pseudocode

```
procedure insert (after_me, new_name, new_mark)
    create the new node with new_name and new_mark
    set the next field of the new node to the next
        field of the node referenced by after_me
    set the next field of the after_me node so that
        it points to the new node
```

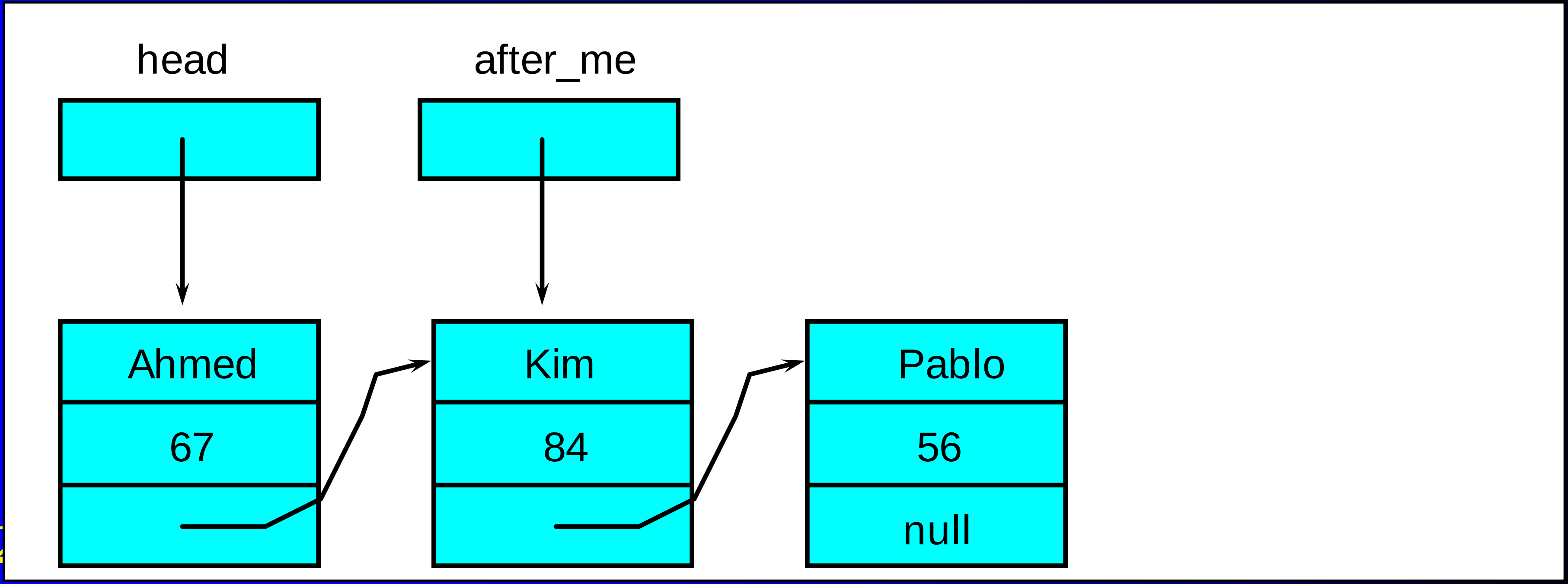
Insertion (ctd)

- JAVA source code

```
public void insert(StudentNode after_me, String
new_name, int new_mark) {
  StudentNode p = new StudentNode(new_name,
new_mark);

  p.next = after_me.next;
  after_me.next = p;
}
```

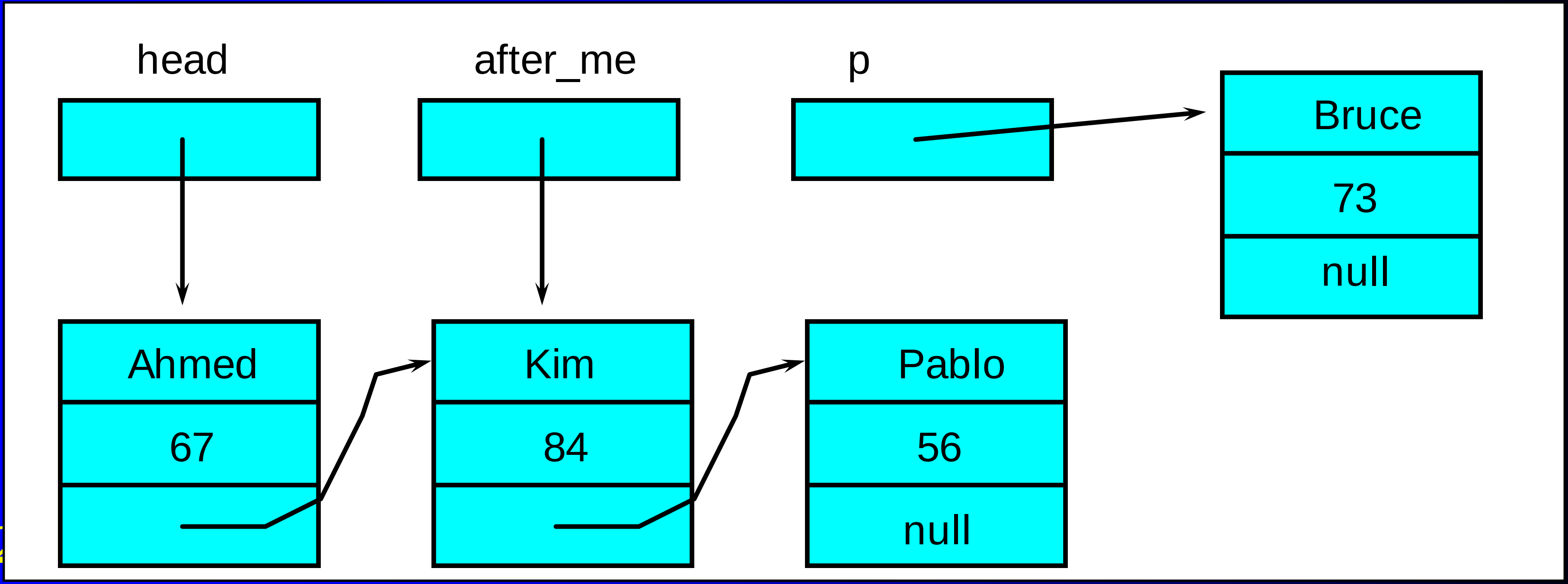
Insertion (ctd)

```
public void insert(StudentNode after_me, String new_name,
               int new_mark) {
  StudentNode p = new StudentNode(new_name,
new_mark);

  p.next = after_me.next;
  after_me.next = p;
}
```



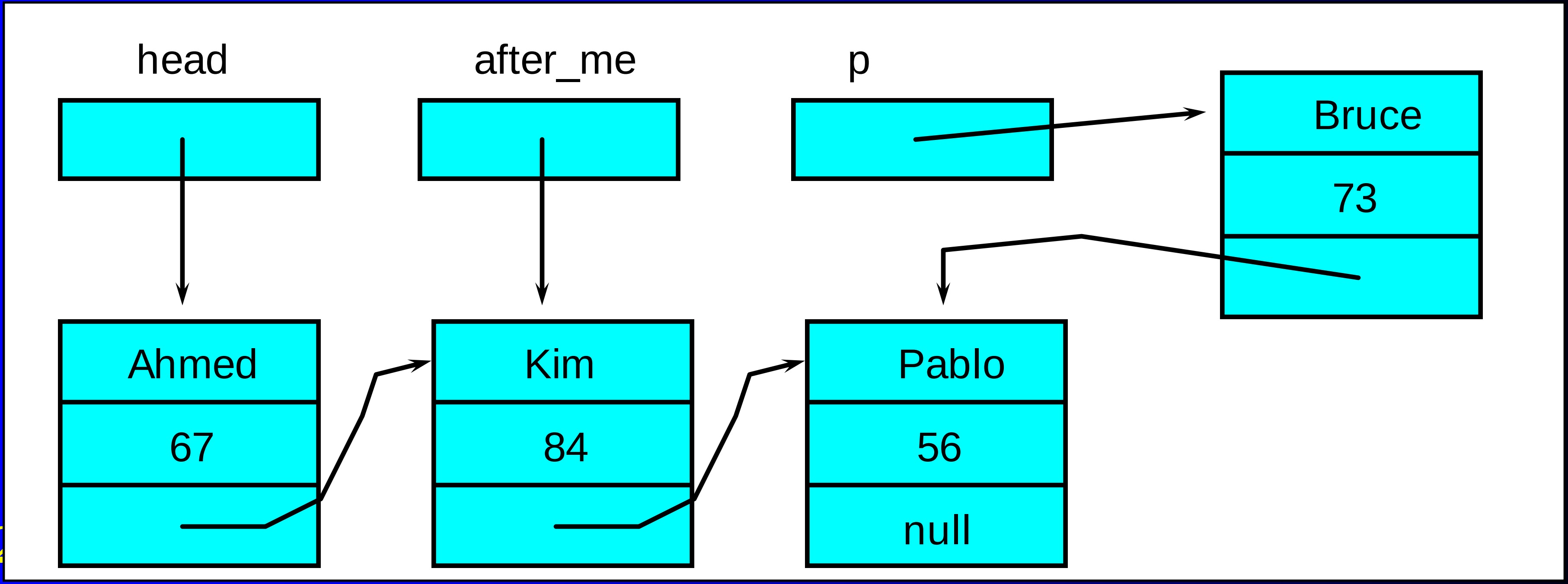
Insertion (ctd)

```
public void insert(StudentNode after_me, String new_name,
               int new_mark) {
  StudentNode p = new StudentNode(new_name,
new_mark);

  p.next = after_me.next;
  after_me.next = p;
}
```



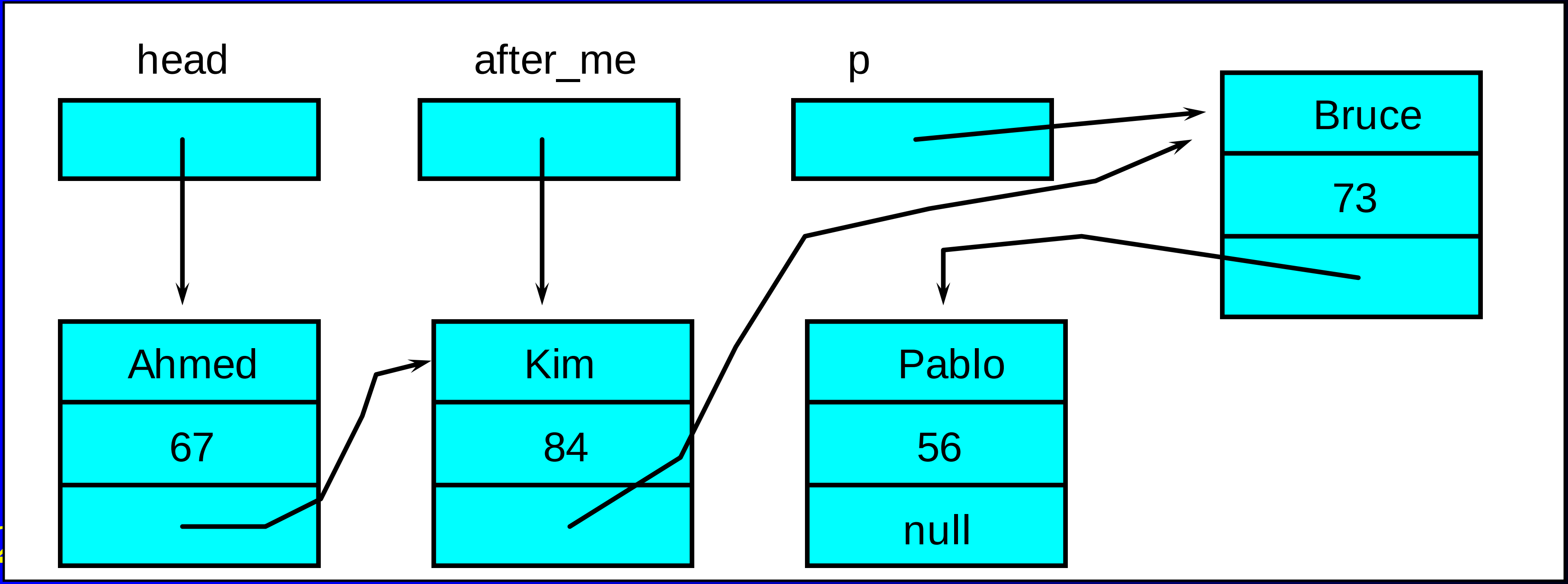
Insertion (ctd)

```
public void insert(StudentNode after_me, String new_name,
               int new_mark) {
  StudentNode p = new StudentNode(new_name,
new_mark);

  p.next = after_me.next;
  after_me.next = p;
}
```



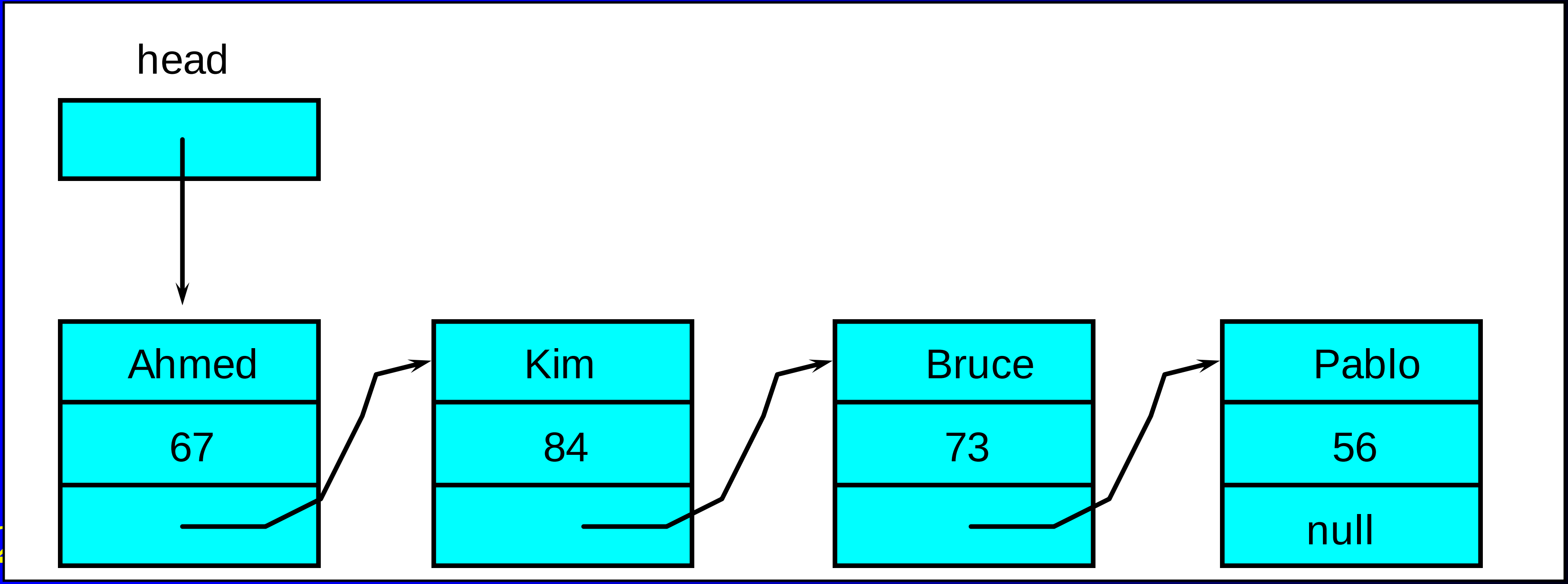
Insertion (ctd)

```
public void insert(StudentNode after_me, String new_name,
               int new_mark) {
  StudentNode p = new StudentNode(new_name,
new_mark);

  p.next = after_me.next;
  after_me.next = p;
}
```



Insertion (ctd)

```
public void insert(StudentNode after_me, String new_name,
              int new_mark) {
  StudentNode p = new StudentNode(new_name,
new_mark);

  p.next = after_me.next;
  after_me.next = p;
}
```



Insertion (ctd)

- Problem:

  Using the insert method, insert Tran, with a mark of 62, after Bruce
- Assume Bruce is in the list

Insertion (ctd)

- Problem:

  Using the insert procedure, insert Tran, with a mark of 62, after Bruce

- Assume Bruce is in the list

```
StudentNode p = head;
while (!(p.name.equals(“Bruce”)) )
{
   p = p.next;
}
insert  (p, “Tran”, 62);
```