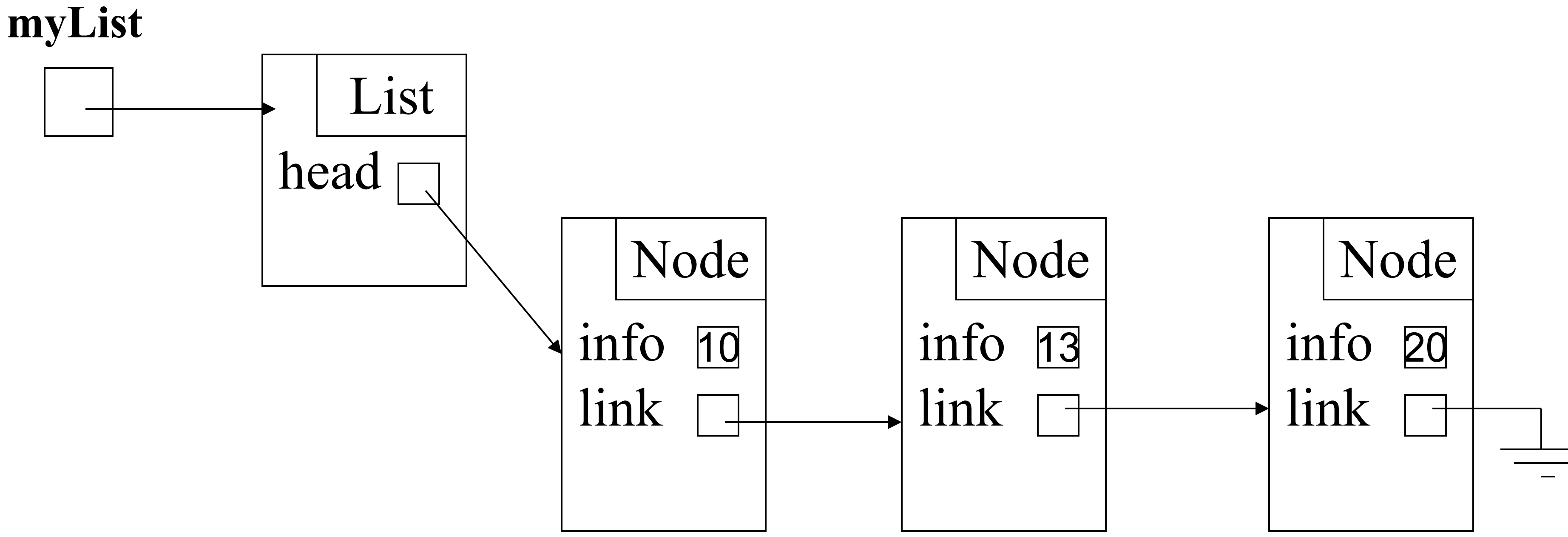
Inserting a Node in a Linked List

Problem #1

- Consider the following linked list where the nodes are sorted in numerical order.



- In your class List,

```
class List
{
    private Node head;
    //**********************
    class Node
    {
          int info;
          Node link;
           //*****************
          Node (int i, Node s)
          {
                info = i;
                link = s;
          }

    }
}
```

Defines the head of a linked list

- Fields can only be accessed through List
-Defines any additional nodes
- This one contains a field called
item and a field called link

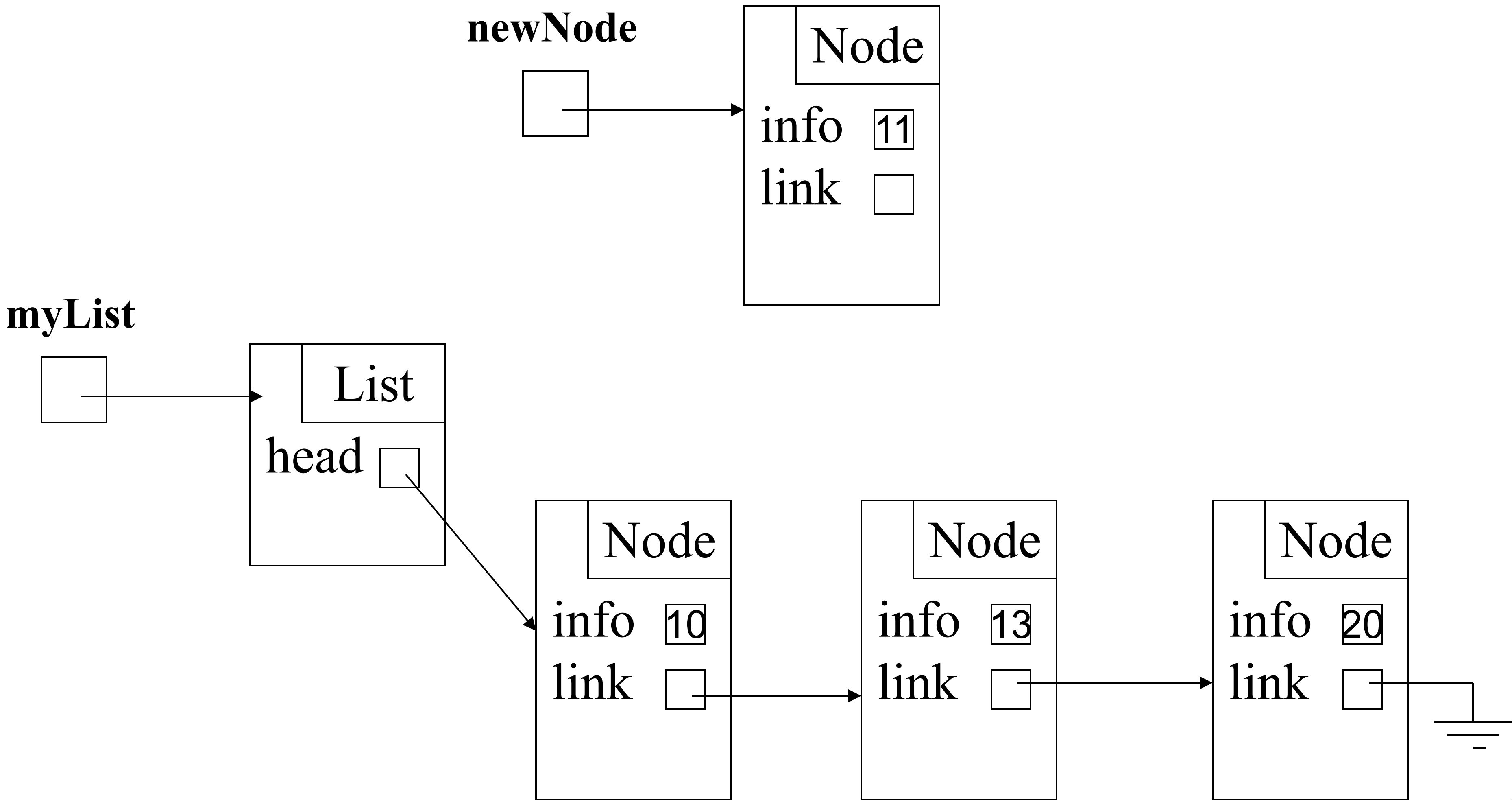
Constructor for a node

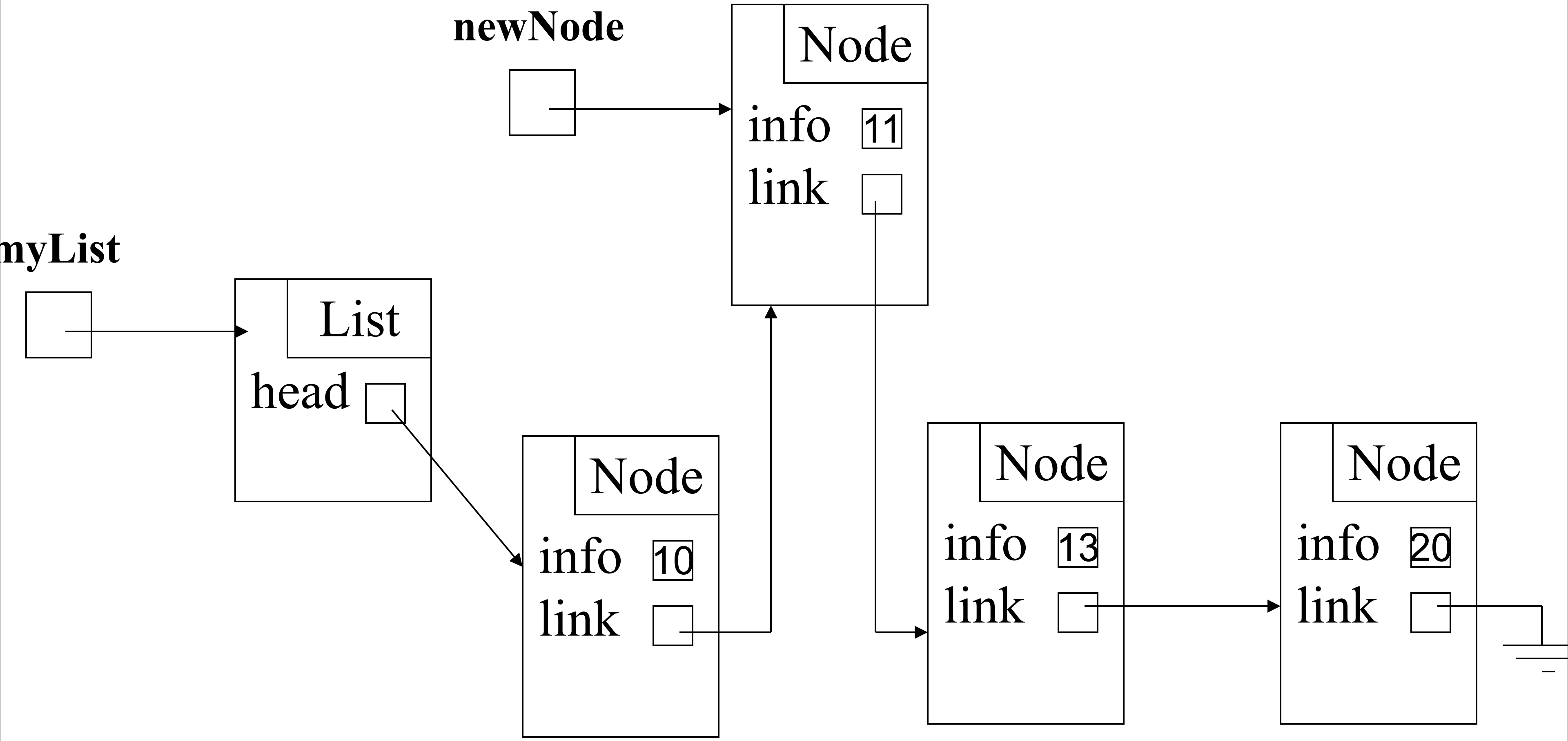
Solution

- Step 1: Find the correct location to insert the node

- Step 2: Adjust the links so that the node is inserted at this point

- A new Node is to be inserted in order of the numbers contained in the field info




newNode
Node
info 11
link
myList
List
head
Node
info 10
link
Node
info 13
link
Node
info 20
link

The Syntax

- The following method *insert* will receive a variable called item of type int.
  - This method will traverse through the linked list and create a Node containing item and insert it in the correct location in the List
- Usually we would create a reference variable to move along the linked list but
  - Because a link points in only one direction, we will need to create two reference variables. One pointing to the current Node and one pointing to the previous one.

```
public void insert (int item)
{
            Node current = head;   // current points to the current Node
            Node previous = null;    // previous points to the previous Node
                                     // The new Node will we inserted between previous and current
            boolean located = false;    // Once the location is found, located will be true
            while (located==false && current != null){
                        if (item < current.info)
                                    located = true;
                        else
                        {
                                    previous = current;    // moves along the list
                                    current = current.link;
                        }
            }
            Node newNode = new Node(item, current);
            // creates a new Node with item as info and points to ‘current’

            if (current == head)
                        head = newNode;  // if new Node was inserted into an empty list
            else
                        previous.link = newNode;
                        // otherwise the ‘previous’  node points to the new node
}
```

```
List myList = new List();
myList.insert(10);
myList.insert(13);
myList.insert(20);
myList.insert(11);
```