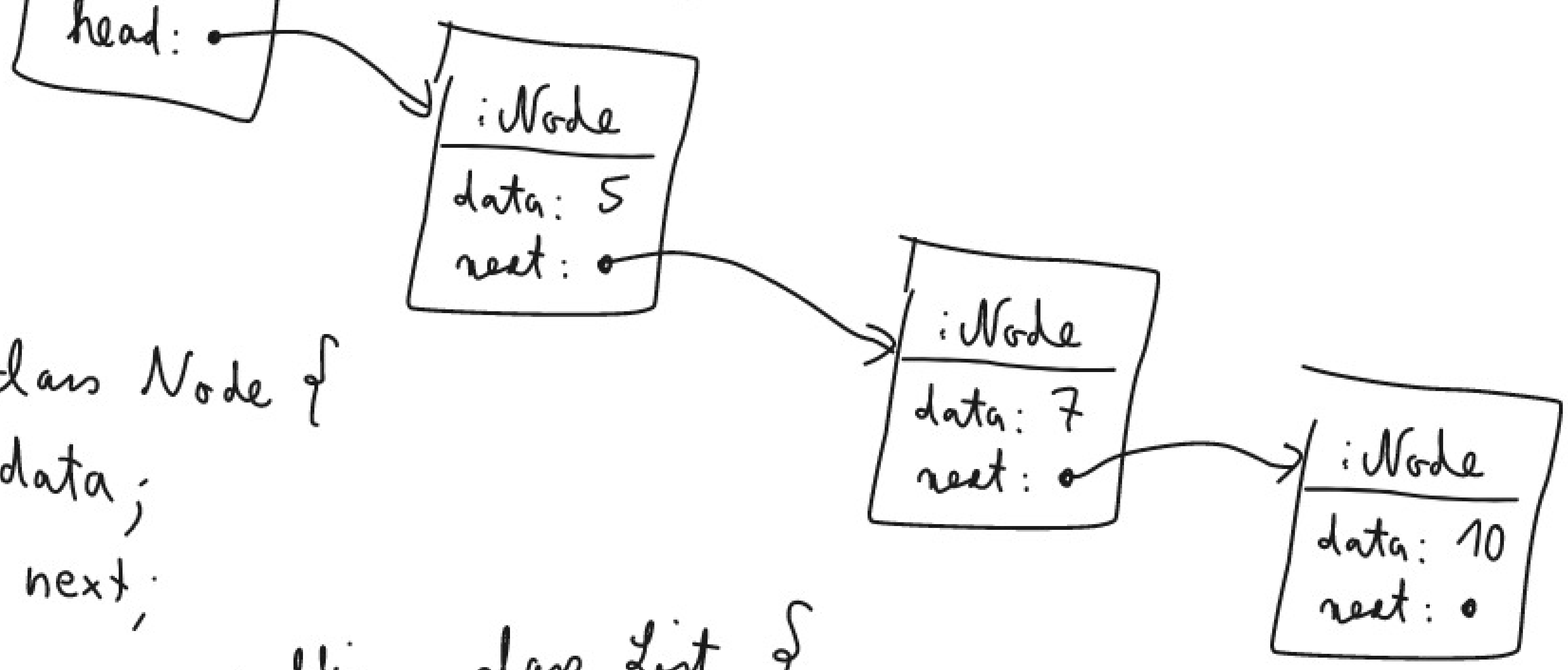
Linked List

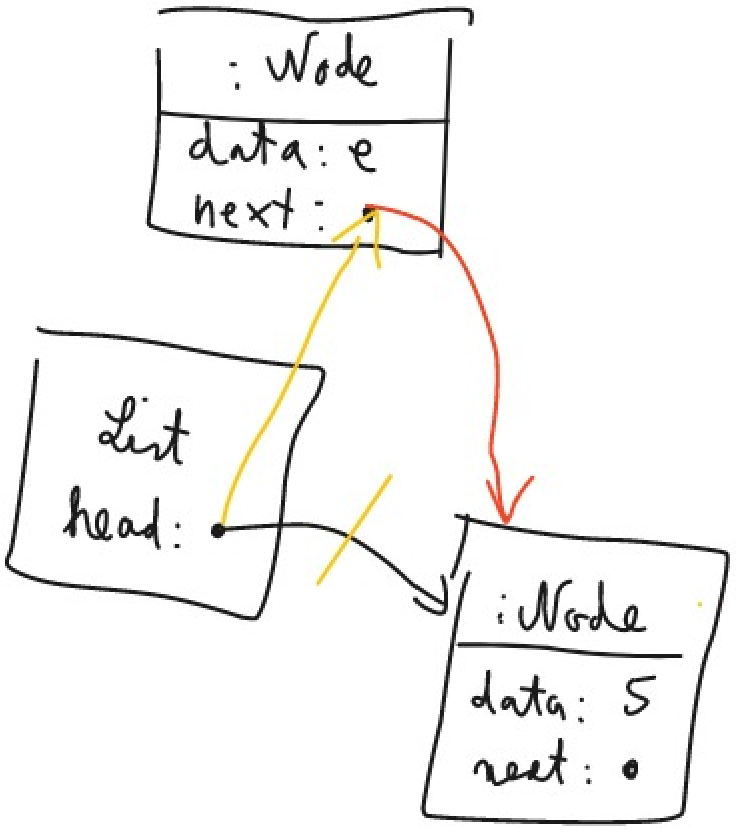


```
  public class Node {
public int data;
public Node next;
   }
```

```
public class List {
private Node head;
private int size = 0;

public void addFirst(int e){
      Node n = new Node();
       n.data = e;  size++;
       if(head == null){
           head= n;  return;
       }
       Node z = head;
       head = n;
       n.next = z;
```

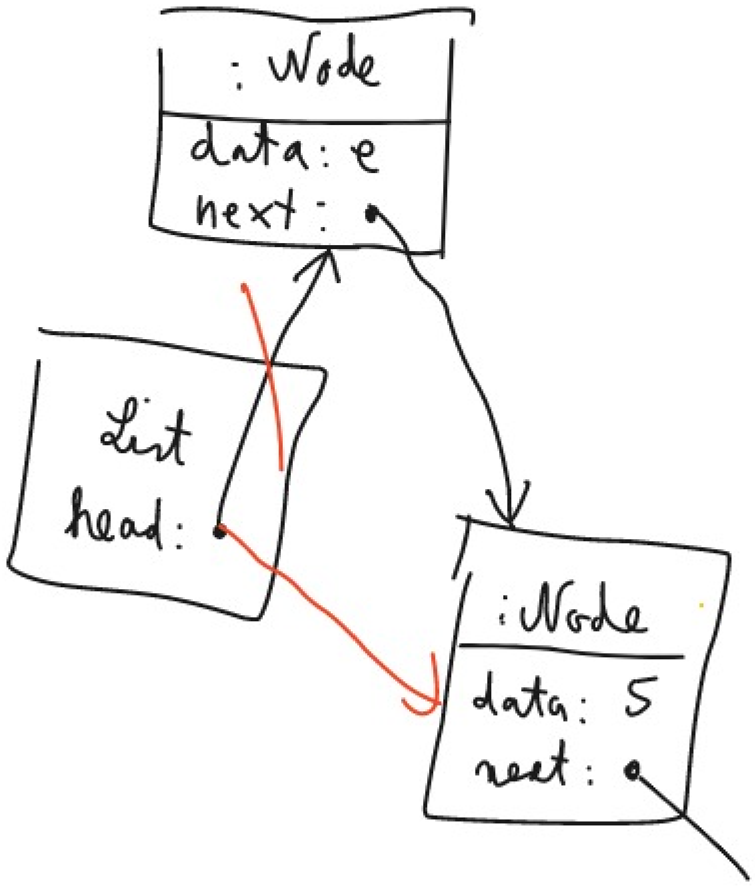
oder n.next=head;
head = n;

```
}
```



```
public int size(){
   ~~if(head==null){~~
     ~~return 0;~~
   ~~}~~
   Node n= head;
   int s = 0
   while (n!= null){
              s++;
              n= n.next;
   }
     return s;
}
```

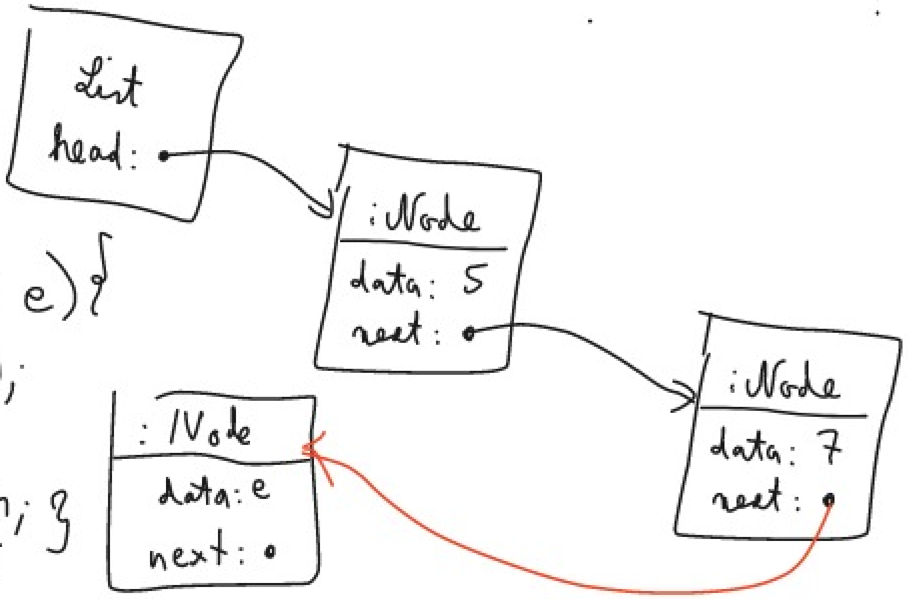
ode



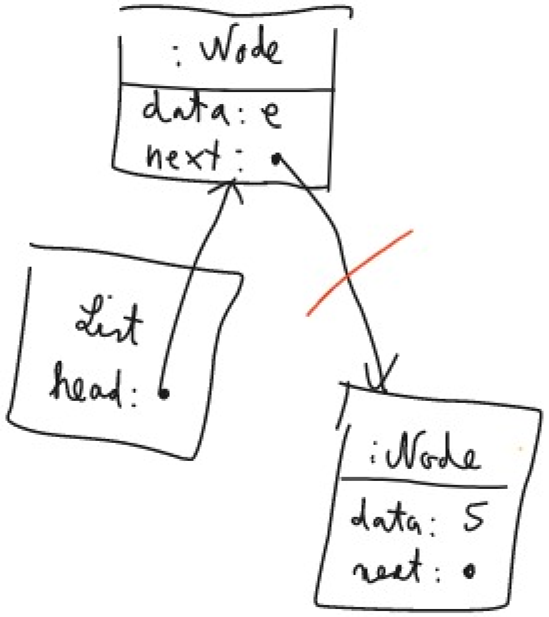
```
public void removeFirst(){
        if(head == null)
           return;

        head=head.next;
        size--;

}
```

```
public int size(){
    return size;
}
```

```
public void addLast(int e){
    Node n= new Node();
    n.data = e;
    size++;
    if(head==null){head=n;
                   return; }
    Node c= head;
    while(c.next!=null){
        c=c.next;
    }
    c.next= n;

}
```

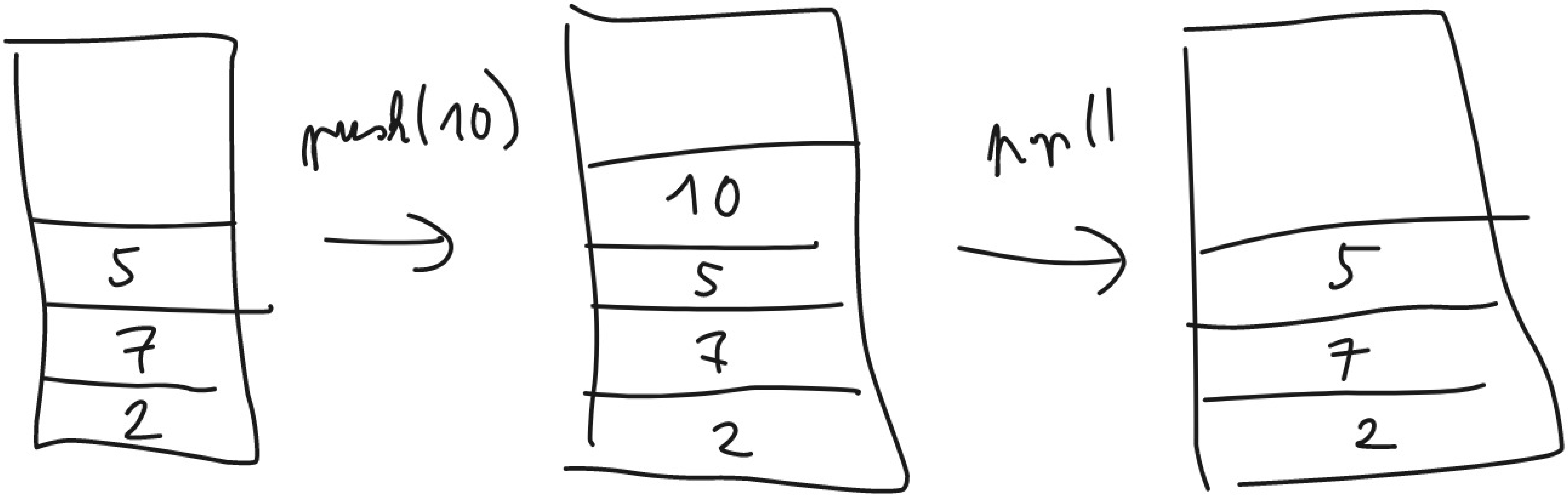


```
public void removeLast(){
    if(head==null){
        return;
    } size--;
    Node c= head;
    if(head.next ==null){
        head=null; return;
    }
    while(c.next.next!=null){
        c=c.next;
    }
    c.next = null
}
```



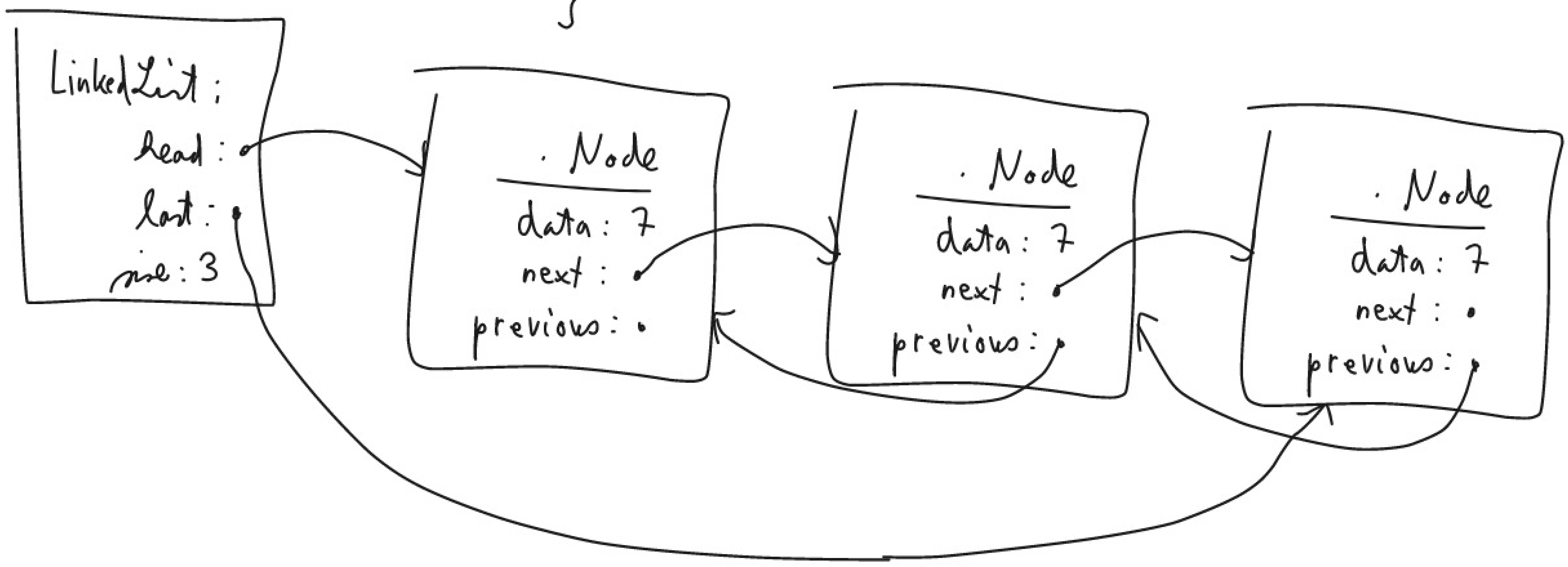
```
public int getFirst(){
    return head.data;

}
```

```
public int getLast(){
                    if(size==0) throw Exception("Liste leer!");
    Node c= head;
    while(c.next!=null){
        c=c.next;
    }
    return c.data;
}
```



```
public class Stack {
    private List l = new List();
    public void push(int e) {
        l.addFirst(e);
    }
    public int pop() {
        int z = l.getFirst();
        l.removeFirst();
    }   return z;


}
```

}

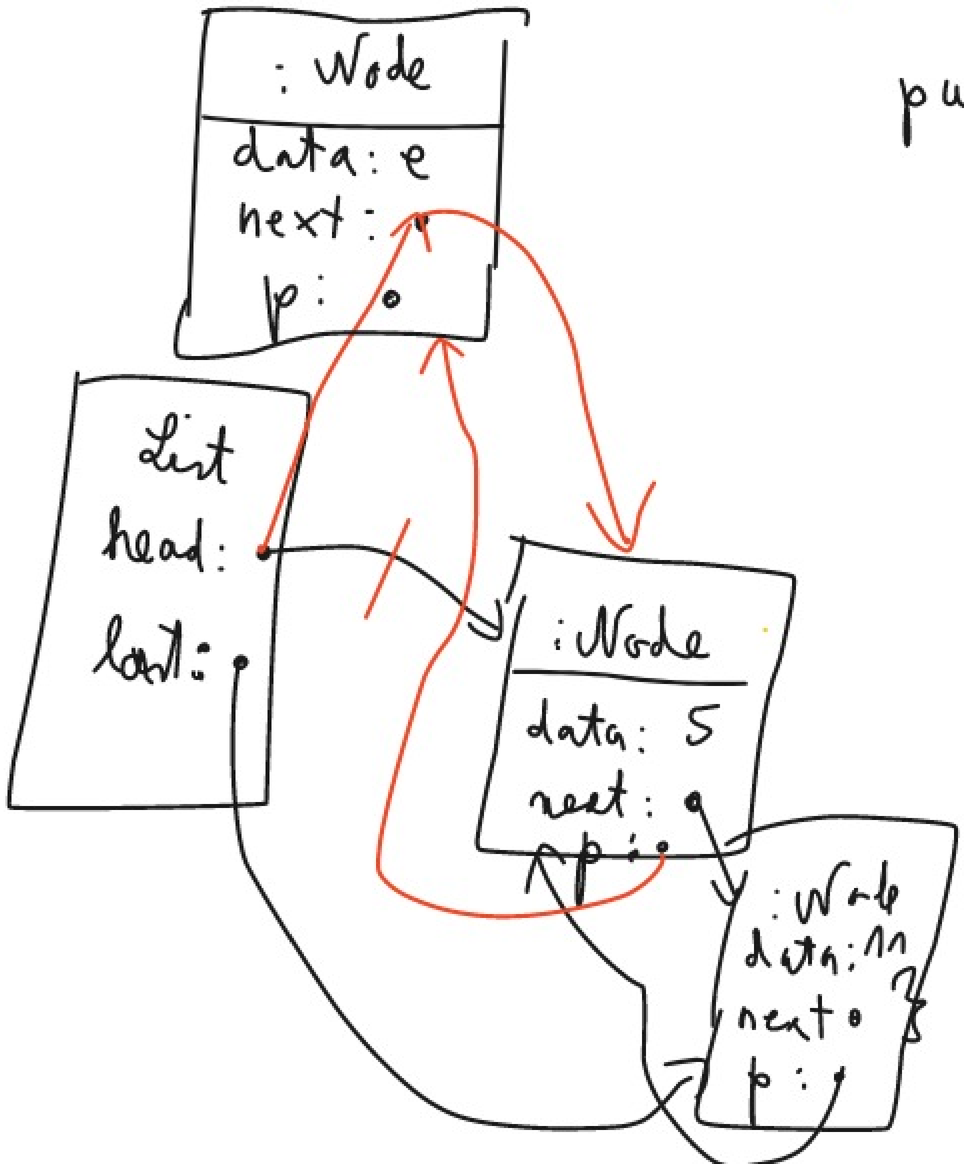


```
public class Node {
public int data;
public Node next;
public Node
    previous;
}
```

```
public class List {
private Node head; private Node last;
private int size = 0;

public void addFirst(int e){
    Node n = new Node();
    n.data = e; size++;
    if(head == null){
        head = n; Last = n; return;
    }
    Node z = head;
    head = n;
    n.next = z;
    z.previous = n;
}
```

oder

```
n.next = head;
head = n;
```



miro