# Shallow vs Deep Copying

Copying Linked Lists

# Poll Everywhere Question

What will the output of the following code be, assuming `ListNode` and `ReadThis` are defined as seen in class?

```
public class PollQ
{
    public static void main(String[] args)
    {
        ListNode n1 =
           new ListNode(new ReadThis("URL-1"));

        n1.next =
           new ListNode(new ReadThis("URL-2"));

        n1.next.next =
           new ListNode(new ReadThis("URL-3"));

        ListNode newList = new ListNode(n1.data);
        newList.next = new ListNode(n1.data);

        newList.next.data.url += "-!!";

        System.out.println(n1.data.url);
    }
}
```

**Text 37607**

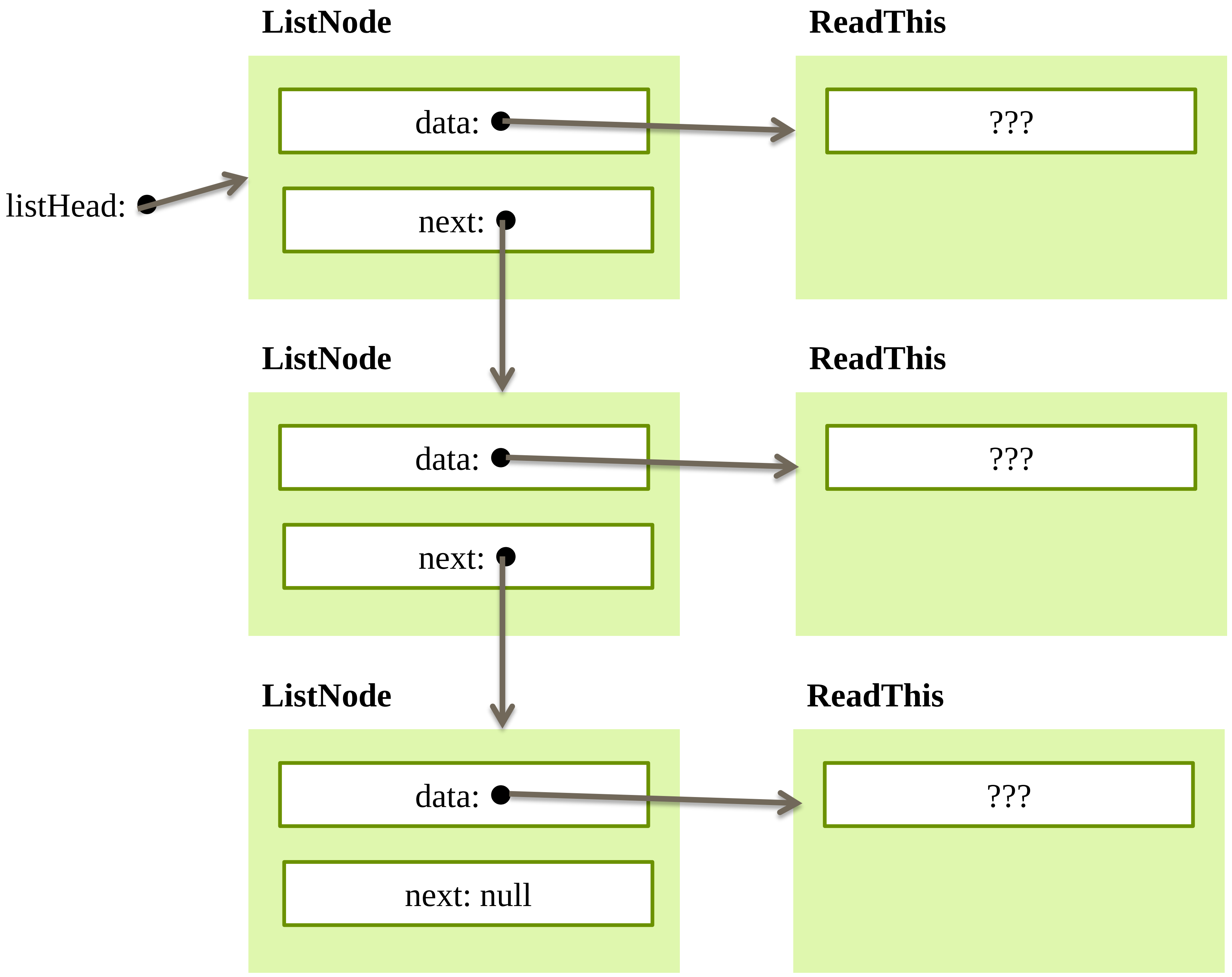
**234538**
`URL-1`

**234539**
`URL-1-!!`

**276202**
It will not compile

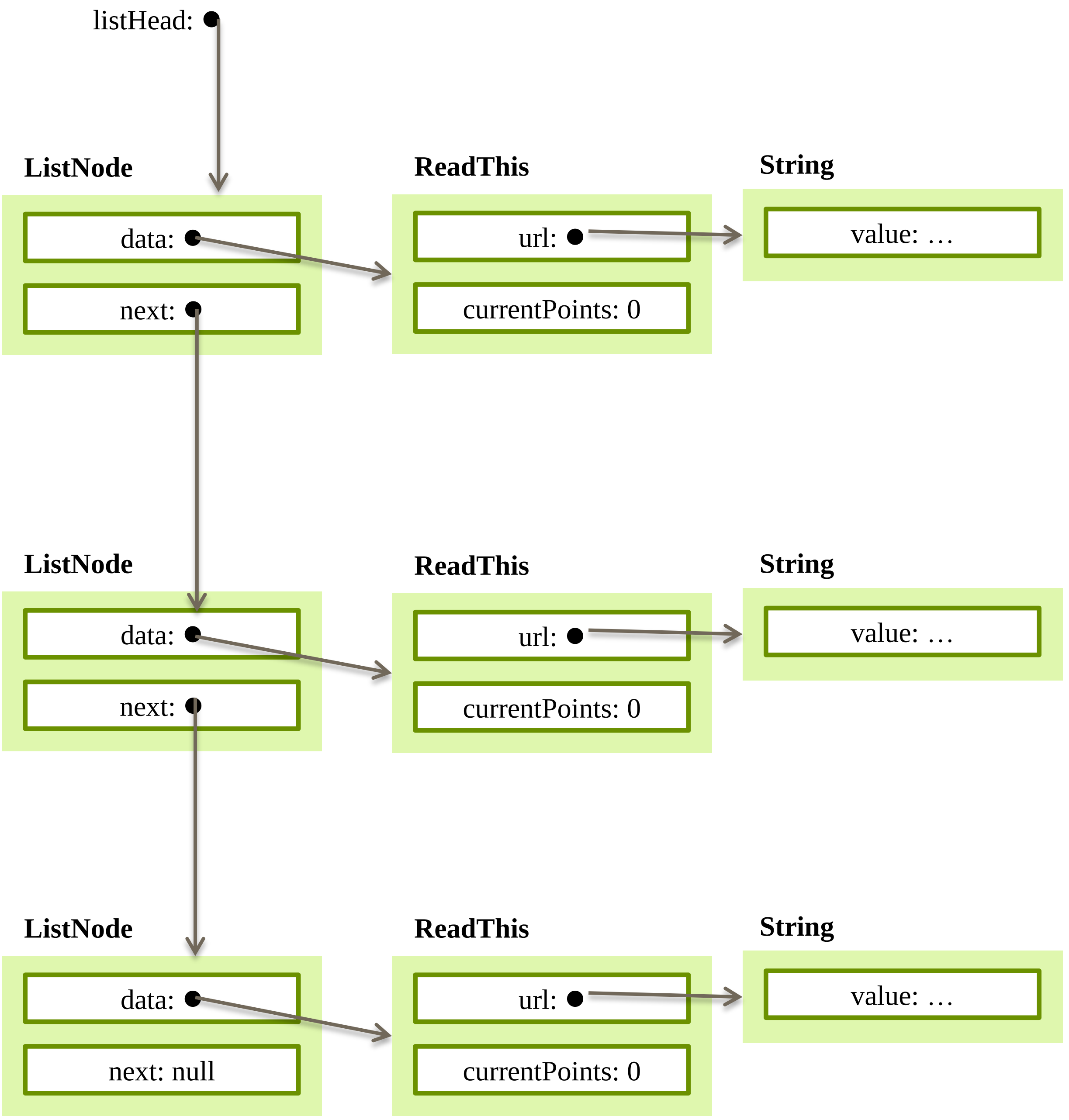
**276203**
There will be a runtime error

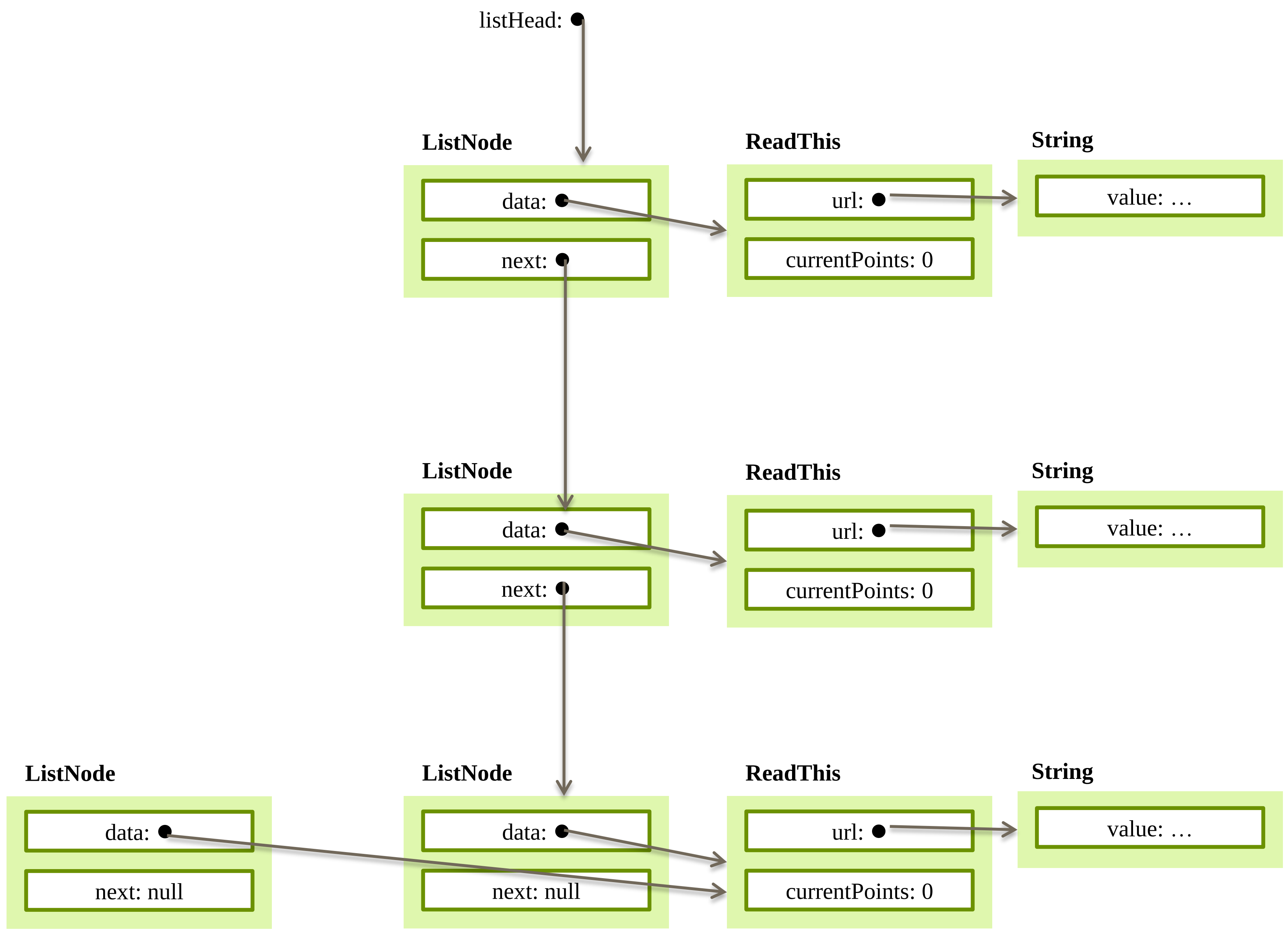
## Copying Linked Lists

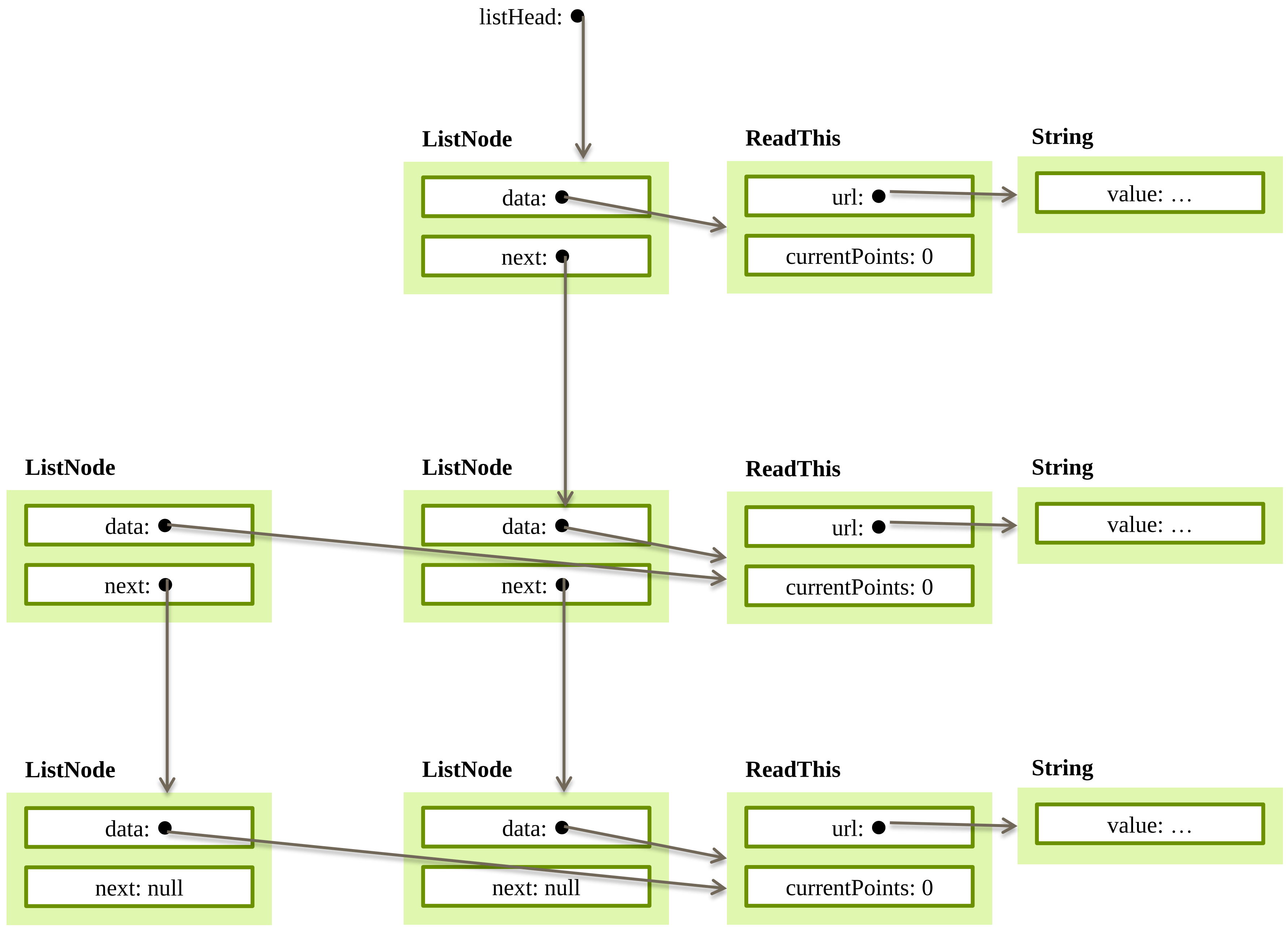

ListNode
ReadThis
data:
???
listHead:
next:
ListNode
ReadThis
data:
???
next:
ListNode
ReadThis
data:
???
next: null

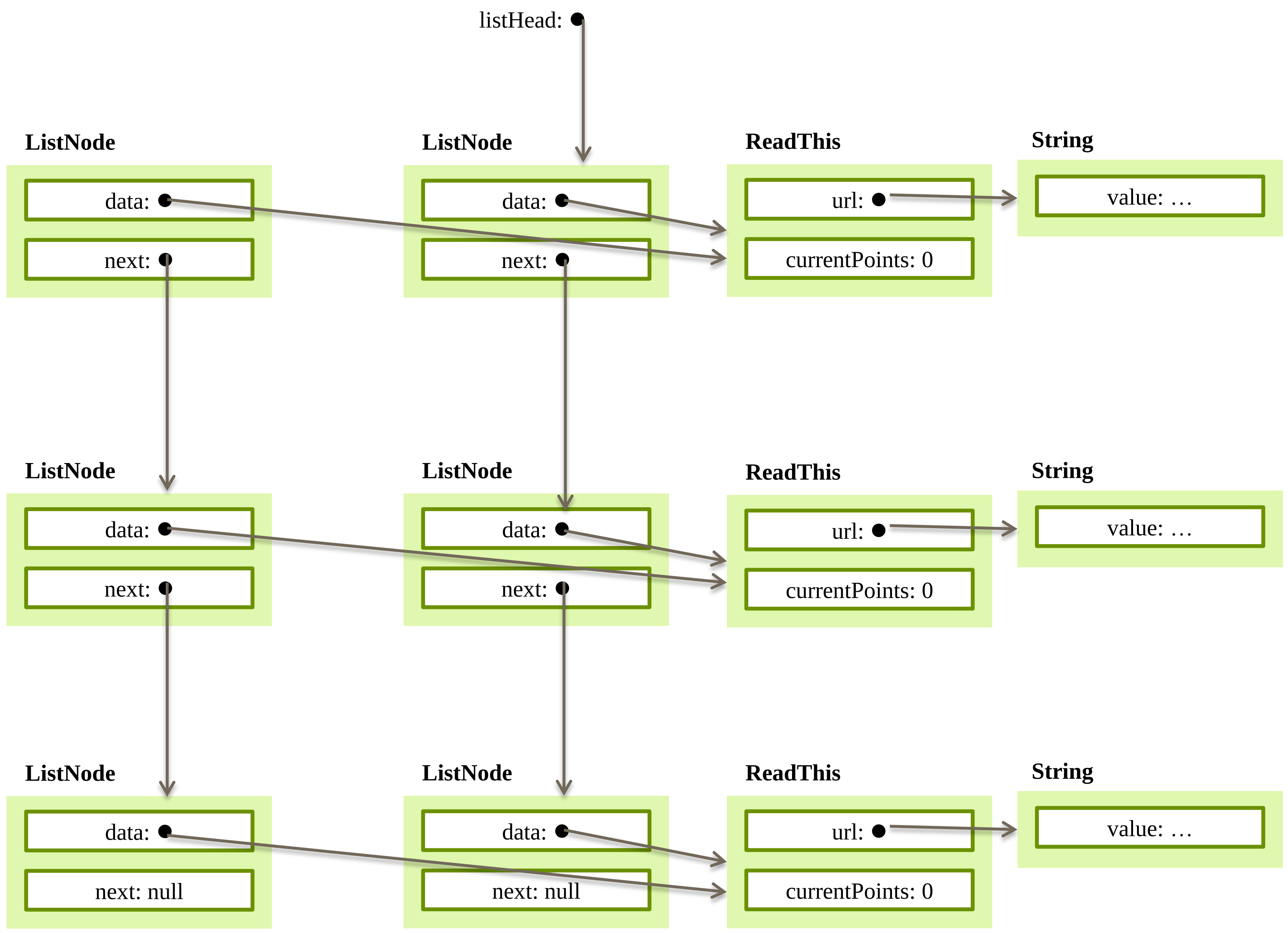
## How do we copy the list?

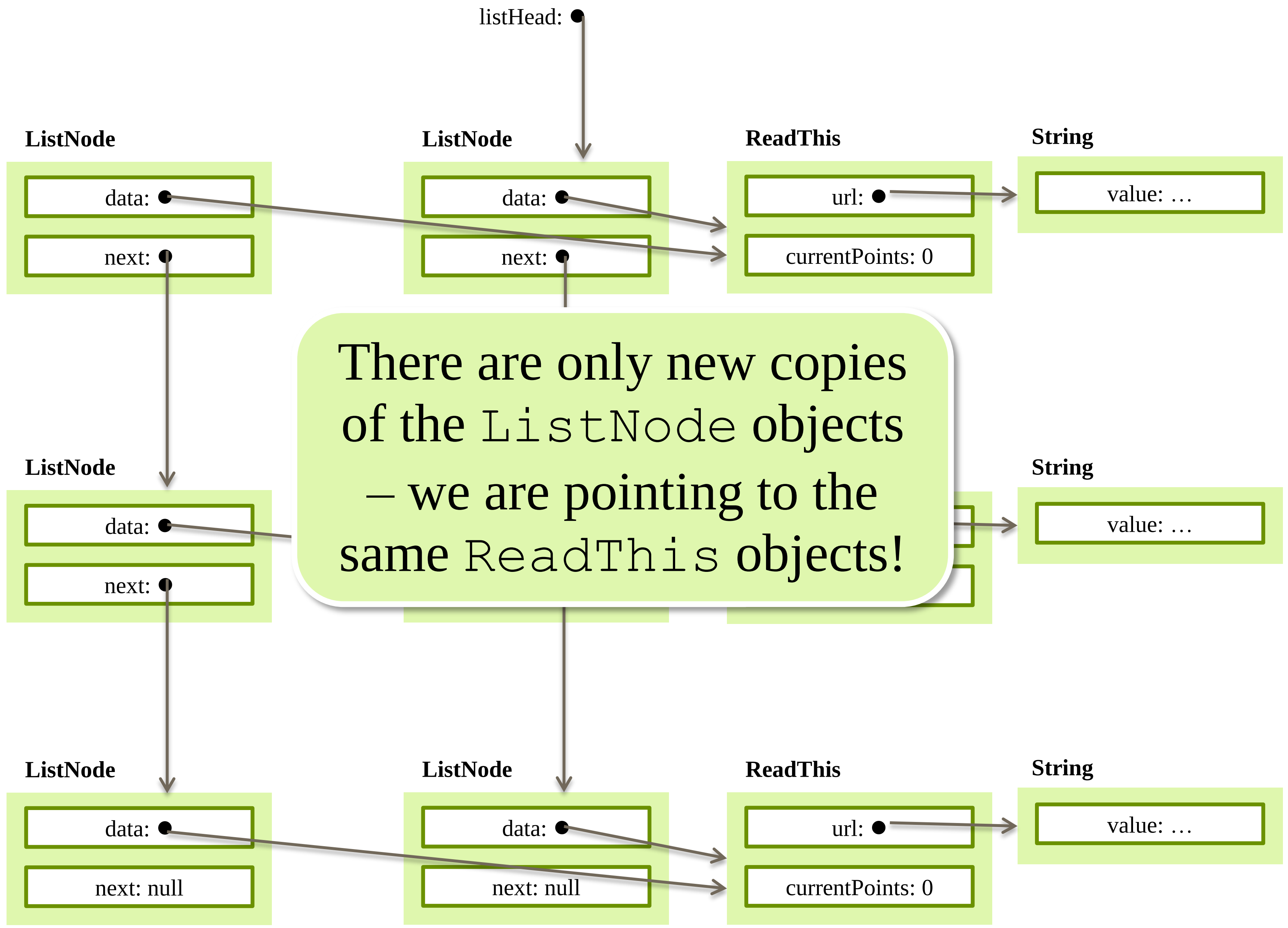
Walk through the list backwards, make a new node to replace each of the old nodes, copy over the information.


listHead: ●
ListNode
data: ●
next: ●
ReadThis
url: ●
currentPoints: 0
String
value: …
ListNode
data: ●
next: ●
ReadThis
url: ●
currentPoints: 0
String
value: …
ListNode
data: ●
next: null
ReadThis
url: ●
currentPoints: 0
String
value: …


listHead:
ListNode
data:
next:
ReadThis
url:
currentPoints: 0
String
value: …
ListNode
data:
next:
ReadThis
url:
currentPoints: 0
String
value: …
ListNode
data:
next: null
ListNode
data:
next: null
ReadThis
url:
currentPoints: 0
String
value: …


listHead: ●
ListNode
data: ●
next: ●
ReadThis
url: ●
currentPoints: 0
String
value: …
ListNode
data: ●
next: ●
ListNode
data: ●
next: ●
ReadThis
url: ●
currentPoints: 0
String
value: …
ListNode
data: ●
next: null
ListNode
data: ●
next: null
ReadThis
url: ●
currentPoints: 0
String
value: …


listHead:
ListNode
data:
next:
ListNode
data:
next:
ReadThis
url:
currentPoints: 0
String
value: …
ListNode
data:
next:
ListNode
data:
next:
ReadThis
url:
currentPoints: 0
String
value: …
ListNode
data:
next: null
ListNode
data:
next: null
ReadThis
url:
currentPoints: 0
String
value: …



There are only new copies of the `ListNode` objects – we are pointing to the same `ReadThis` objects!

## Shallow Copy

When a **shallow copy** is made of a class, the references are copied but not the objects being referred to.

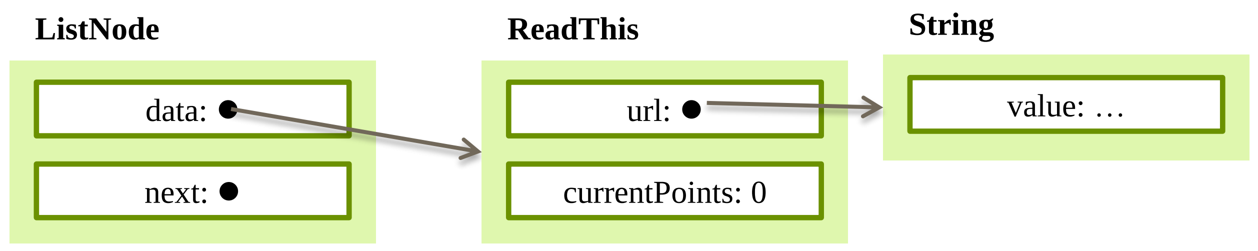
## What if we need a more complete copy of the list?

The procedure is the same; we just have to make sure to copy the objects stored in the nodes as well.

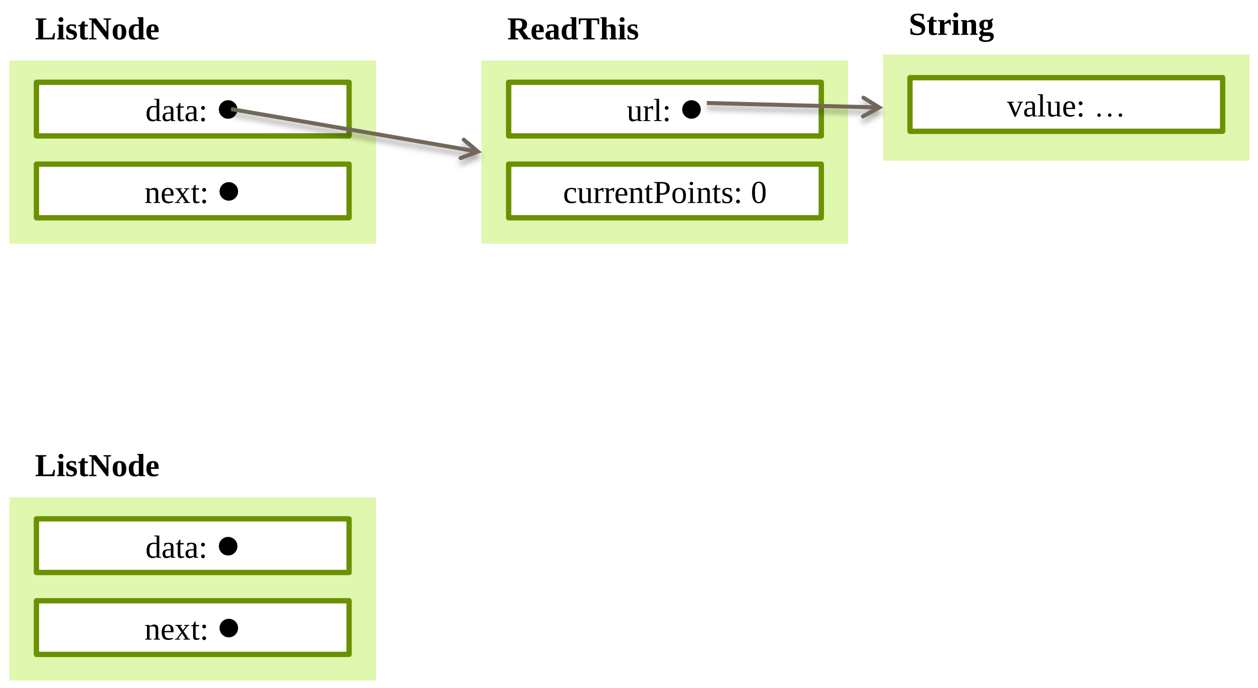
## Deep Copy

When a **deep copy** is made of a class, a copy of every object the class refers to will be created as well.

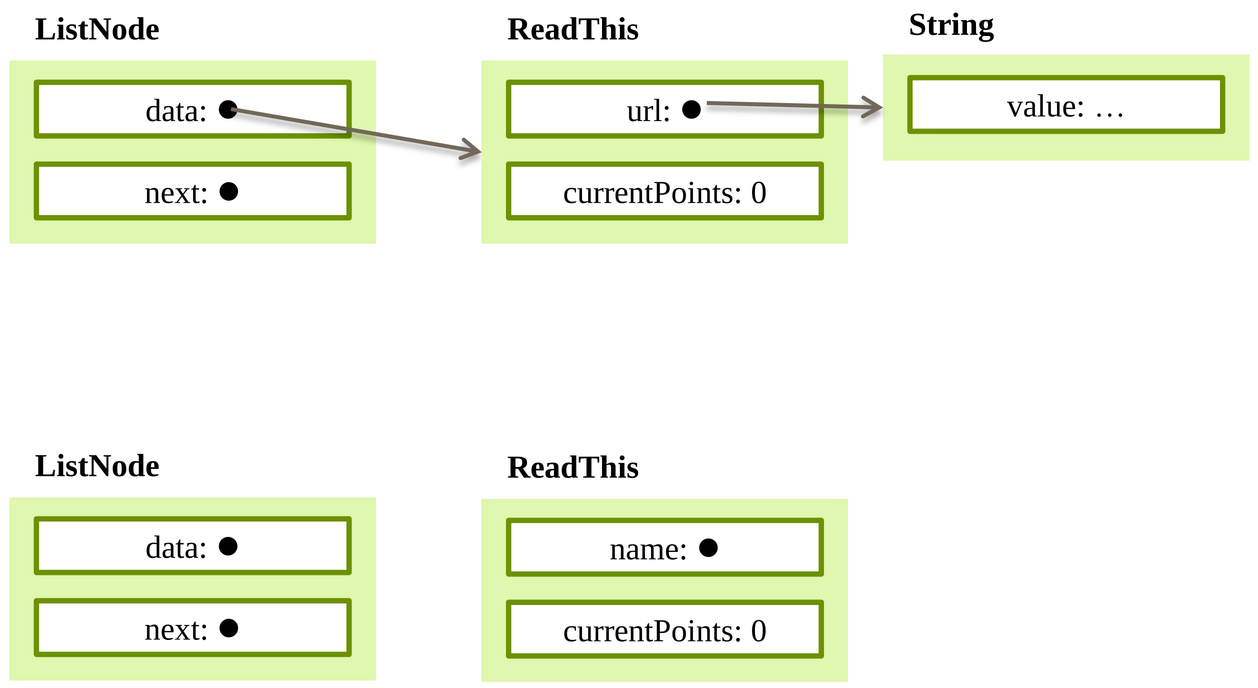
Suppose we want to deep copy a list node, but not the `ReadThis` object…



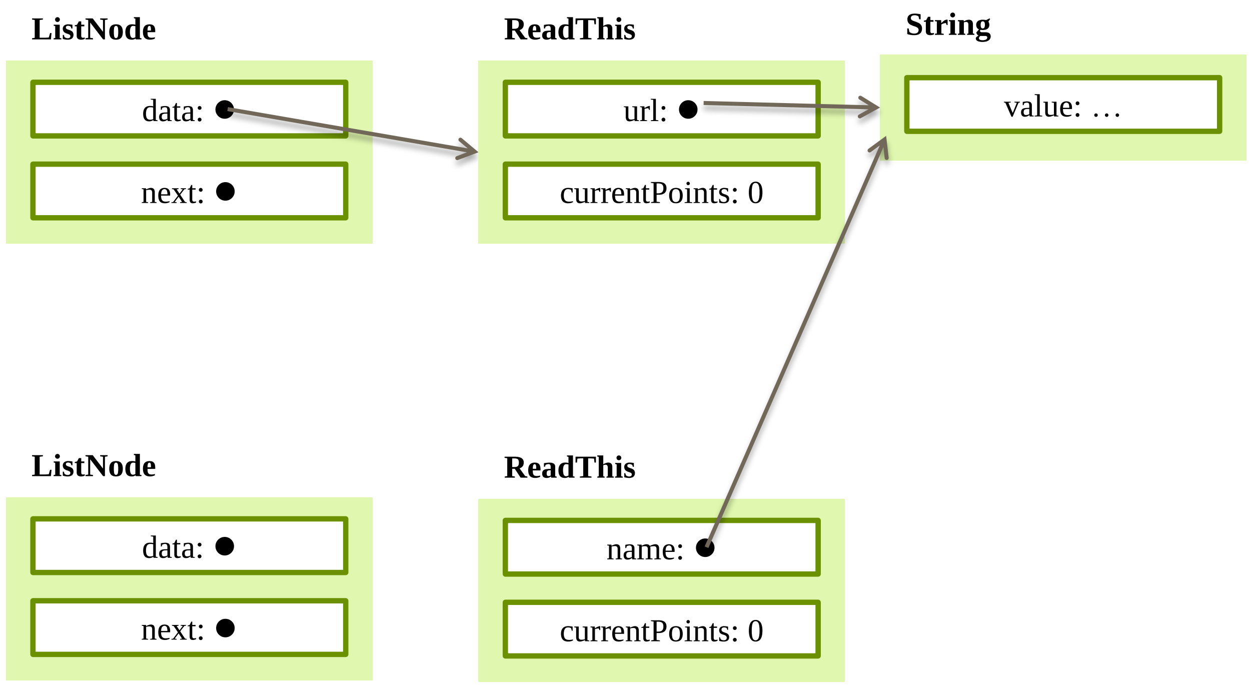
Make a new `ListNode`:



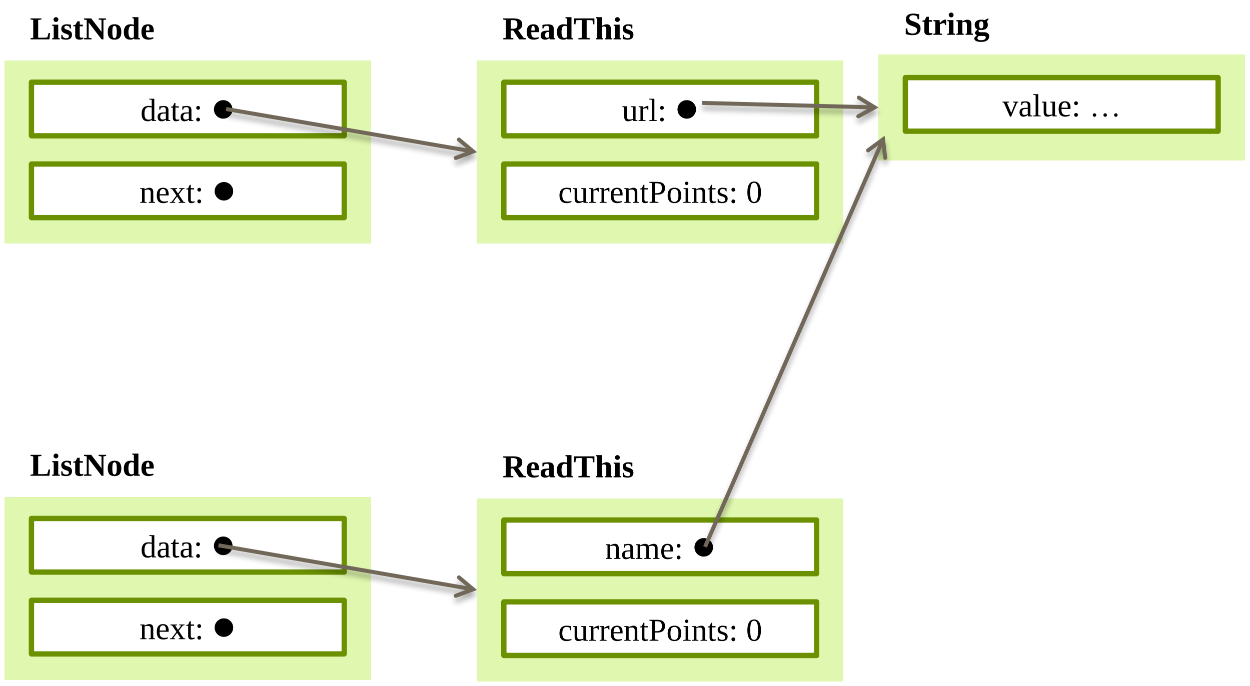
# Make a new `ReadThis`:



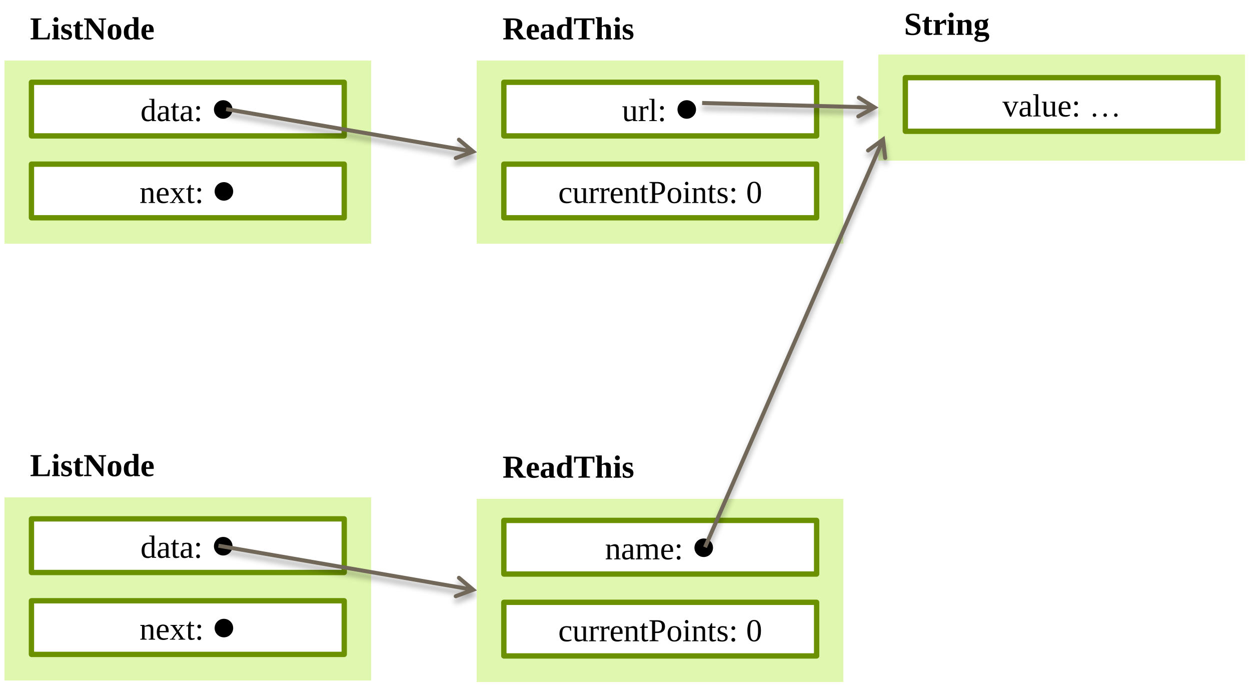
## Copy the `ReadThis`’s `String` reference and `currentPoints` value:



Point the new list node's data to the new `ReadThis`:



Now we have a **deep copy** of the `ListNode`.


ListNode
data:
next:
ReadThis
url:
currentPoints: 0
String
value: …
ListNode
data:
next:
ReadThis
name:
currentPoints: 0