Building Java Programs

Chapter 16
Linked Nodes

reading: 16.1
## Road Map

<table>
<thead>
<tr>
<th>CS Concepts</th>
<th>Java Language</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Client/Implementer</td>
<td>• Exceptions</td>
</tr>
<tr>
<td>• Efficiency</td>
<td>• Interfaces</td>
</tr>
<tr>
<td>• Recursion</td>
<td>• References</td>
</tr>
<tr>
<td>• Regular Expressions</td>
<td>• Comparable</td>
</tr>
<tr>
<td>• Grammars</td>
<td>• Generics</td>
</tr>
<tr>
<td>• Sorting</td>
<td>• Inheritance/Polymorphism</td>
</tr>
<tr>
<td>• Backtracking</td>
<td>• Abstract Classes</td>
</tr>
<tr>
<td>• Hashing</td>
<td></td>
</tr>
<tr>
<td>• Huffman Compression</td>
<td></td>
</tr>
</tbody>
</table>

### Data Structures

- Lists
- Stacks
- Queues
- Sets
- Maps
- Priority Queues

### Java Collections

- Arrays
- ArrayList
- LinkedList
- Stack
- TreeSet / TreeMap
- HashSet / HashMap
- PriorityQueue
Recall: stacks and queues

- **stack**: retrieves elements in reverse order as added
- **queue**: retrieves elements in same order as added

Diagram:
- Stack:
  - Push
  - Pop, Peek
  - Top: 3
  - Bottom: 1

- Queue:
  - Front: 1
  - Back: 3
  - Add
  - Remove, Peek
Array vs. linked structure

- All collections in this course use one of the following:
  - an **array** of all elements
    - examples: ArrayList, Stack, HashSet, HashMap

```
42  -3  17  9
```

- **linked objects** storing a value and references to other(s)
  - examples: LinkedList, TreeSet, TreeMap

```
front → 42 → -3 → 17 → 9 null
```

- First, we will learn how to create a **linked list**.
- To understand linked lists, we must understand **references**.
Memory for a List

- **Array (contiguous in memory)**
  
  | 42 | -3 | 17 | 9 |

- **Spread in memory**

  | 42 | 9 | -3 | 17 |
[42, -3, 17, 9]

```
ListNode
  . data (int)
  . next (ListNode)
```
A list node class

```java
public class ListNode {
    public int data;
    public ListNode next;
}
```

- Each list node object stores:
  - one piece of integer data
  - a reference to another list node

- `ListNode` objects can be "linked" into chains to store a list of values:
References to same type

What would happen if we had a class that declared one of its own type as a field?

```java
public class Strange {
    private String name;
    private Strange other;
}
```

Will this compile?
- If so, what is the behavior of the `other` field? What can it do?
- If not, why not? What is the error and the reasoning behind it?
public class ConstructList1 {
    public static void main(String[] args) {
        ListNode list = new ListNode();
        list.data = 42;
        list.next = new ListNode();
        list.next.data = -3;
        list.next.next = new ListNode();
        list.next.next.data = 17;
        list.next.next.next = null;
        System.out.println(list.data + " "+ list.next.data
                           + " "+ list.next.next.data);
        // 42 -3 17
    }
}
List node w/ constructor

```java
public class ListNode {
    int data;
    ListNode next;

    public ListNode(int data) {
        this(data, null);
    }

    public ListNode(int data, ListNode next) {
        this.data = data;
        this.next = next;
    }
}
```

- Exercise: Modify the previous client to use these constructors.
Linked node problem 1

- What set of statements turns this picture:

```
list  
10   
next

10   
next

20   
next
```

- Into this?

```
list  
10   
next

10   
next

20   
next

30   
next
```
Linked node problem 2

• What set of statements turns this picture:

list →

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

• Into this?

list →

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td>10</td>
</tr>
<tr>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>
Suppose we had the following `ListNode`s:

- `list1`:
  - data: 1
  - next: `list2`

- `list2`:
  - data: 4
  - next: `list1`
  - next next: 5

- `list3`:
  - data: 3
  - next: `list2`

What would the lists look like if we ran the code?

```java
list1.next = list2.next;
```
Reassigning references

- when you say:
  - `a.next = b.next;`

- you are saying:
  - "Make variable `a.next` store the same value as `b.next`."
  - Or, "Make `a.next` refer to the same place as `b.next`."

![Diagram showing reassignment of references](image)
References vs. objects

`variable = value;`

A `variable` (left side of `=`) is a place to put a reference
(where the phone number goes; where the base of the arrow goes)
A `value` (right side of `=`) is the reference itself
(the phone number; the destination of the arrow)

- `adjus`
- For the list at right:
  - `a.next = value;` means to update where `a` points
  - `variable = a.next;` means to make `variable` point at
Linked node problem 3

- What set of statements turns this picture:

  list1

  list2

  list1

- Into this?

  list1

  list2
Linked node problem 3

• How many ListNode variables?

• Which variables change?
Linked node problem 3

- How many ListNode variables?

- Which variables change?

```
list1 10  
  \  
  A

list2 30  
  \  
  D

list1 10  
  \  
  A

list2 40  
  \  
  F

A \-> B \-> C
D \-> E \-> F

list1.next.next = list2
```
Linked node problem 3

• How many ListNode variables?

list1

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

list2

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>

list1.next.next = list2
list2 = list2.next

• Which variables change?

list1

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
</tbody>
</table>

list2

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>

list1

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

list2

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
Linked node problem 3

• How many ListNode variables?

list1

A

data 10

next

list2

D

data 30

next

list1

B

data 10

next

list2

C

data 20

next

list1

next

next

next

null

list1.next.next = list2
list2 = list2.next

list1.next.next.next = null

• Which variables change?

list1

A

data 10

next

list2

D

data 40

next
Linked node problem 4

- What set of statements turns this picture:

```
<table>
<thead>
<tr>
<th>list1</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>list2</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
```

- Into this?

```
<table>
<thead>
<tr>
<th>list1</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th>list2</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
```

```
<table>
<thead>
<tr>
<th></th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>3</td>
<td></td>
</tr>
</tbody>
</table>
```