Building Java Programs

Chapter 16
Linked Nodes

reading: 16.1
## Road Map

### CS Concepts
- Client/Implementer
- Efficiency
- Recursion
- Regular Expressions
- Grammars
- Sorting
- Backtracking
- Hashing
- Huffman Compression

### Java Language
- Exceptions
- Interfaces
- References
- Comparable
- Generics
- Inheritance/Polymorphism
- Abstract Classes

### 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 the same order as added

![Diagram of a stack and queue](image)
Array vs. linked structure

- All collections in this course use one of the following:
  - an **array** of all elements
    - examples: ArrayList, Stack, HashSet, HashMap
  - **linked objects** storing a value and references to other(s)
    - examples: LinkedList, TreeSet, TreeMap

- 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
A list node class

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\(_s\) can be "linked" into chains to store a list of values:
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?
List node client example

```java
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
   data  next
      42   |
            |
   data  next
      -3   |
            |
   data  next
      17   null
```
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
  \[
  \begin{array}{c|c}
  \text{data} & \text{next} \\
  \hline
  10 & \phantom{0} \\
  \end{array}
  \rightarrow
  \begin{array}{c|c}
  \text{data} & \text{next} \\
  \hline
  20 & \phantom{0} \\
  \end{array}
  \]

- Into this?

  list
  \[
  \begin{array}{c|c}
  \text{data} & \text{next} \\
  \hline
  10 & \phantom{0} \\
  \end{array}
  \rightarrow
  \begin{array}{c|c}
  \text{data} & \text{next} \\
  \hline
  20 & \phantom{0} \\
  \end{array}
  \rightarrow
  \begin{array}{c|c}
  \text{data} & \text{next} \\
  \hline
  30 & \phantom{0} \\
  \end{array}
  \]
Linked node problem 2

- What set of statements turns this picture:

```
list   data   next
   10       —   data   next
    —       20
```

- Into this?

```
list   data   next
   30       —   data   next
    —       10       —   data   next
                                 —       20
```
Suppose we had the following `ListNode`s:

<table>
<thead>
<tr>
<th>list1</th>
<th>list2</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="data=1,next=2" alt="Node 1" /></td>
<td><img src="data=3,next=4" alt="Node 3" /></td>
</tr>
<tr>
<td><img src="data=2,next=3" alt="Node 2" /></td>
<td><img src="data=4,next=5" alt="Node 4" /></td>
</tr>
</tbody>
</table>

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

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

- What set of statements turns this picture:

  list1
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  10 & \\
  \end{array}\]
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  20 & \\
  \end{array}\]

  list2
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  30 & \\
  \end{array}\]
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  40 & \\
  \end{array}\]

- Into this?

  list1
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  10 & \\
  \end{array}\]
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  20 & \\
  \end{array}\]
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  30 & \\
  \end{array}\]

  list2
  \[\begin{array}{cc}
  \text{data} & \text{next} \\
  40 & \\
  \end{array}\]
Linked node problem 3

How many ListNode variables?

- list1
  - A
  - data: 10
  - next: B

- list2
  - D
  - data: 30
  - next: E

Which variables change?

- list1
  - data: 10
  - next: B

- list2
  - data: 40
  - next: D

- list1
  - data: 20
  - next: C

- list2
  - data: 30
  - next: E
Linked node problem 3

- How many ListNode variables?

- Which variables change?

list1.next.next = list2

```
<table>
<thead>
<tr>
<th></th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>20</td>
</tr>
<tr>
<td>C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td>40</td>
</tr>
<tr>
<td>F</td>
<td>40</td>
<td></td>
</tr>
</tbody>
</table>
```
Linked node problem 3

- How many ListNode variables?
- 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>30</td>
<td></td>
</tr>
</tbody>
</table>

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

D
```
Linked node problem 3

- How many ListNode variables?

- Which variables change?

```
list1
  A

list2
  D

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

list1.next.next.next = null
```
Reassigning references

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

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

- Adjust
- For the list at right:
  - `a.next = value;`
    means to `t` where `points`
  - `variable = a.next;`
    means to make `variable` point at
Linked node problem 4

• What set of statements turns this picture:

```
list1  □

1 ▶ next

list2 □

2 ▶ next
```

• Into this?

```
list1  □

1 ▶ next

list2 □

2 ▶ next
```