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 same order as added

![Diagram of stack and queue operations](image-url)
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 objects can be "linked" into chains to store a list of values:

```
data  next
42    -->  -3    -->  17    -->  9    -->  end
```
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

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
    }
}
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;
    }

    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
```

into this?

```
list
10
20
30
```

- Into this?
Linked node problem 2

- What set of statements turns this picture:

```
list  | data  | next |
10    |       |      |
   |       |      |
20    |       |      |
```

- Into this?

```
list  | data  | next |
30    |       |      |
   |       |      |
10    |       |      |
   |       |      |
20    |       |      |
```
Linked node problem 3

- What set of statements turns this picture:

  list1 →
  | data | next |
  | 10   |     |

  list2 →
  | data | next |
  | 30   |     |

  list1 →
  | data | next |
  | 10   |     |

  list2 →
  | data | next |
  | 40   |     |

- Into this?

  list1 →
  | data | next |
  | 10   |     |

  list2 →
  | data | next |
  | 40   |     |

  list1 →
  | data | next |
  | 20   |     |

  list2 →
  | data | next |
  | 30   |     |
Linked node problem 3

- **How many ListNode variables?**

<table>
<thead>
<tr>
<th>list1</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>list2</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Which variables change?**

<table>
<thead>
<tr>
<th>list1</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>10</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>20</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>30</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>list2</th>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>40</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>E</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Linked node problem 3

- How many ListNode variables?

- Which variables change?

```
list1
  A

  B
  10
  next

list2
  D

  E
  30
  next

list1.next.next = list2

list1
  C
  20
  next

list2
  F
  40
  next

list1
  data  next
  10

list2
  data  next
  30

list1
  data  next
  10

list2
  data  next
  40
```
Linked node problem 3

- How many ListNode variables?

- Which variables change?
Linked node problem 3

- How many ListNode variables?

- Which variables change?

```java
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

\[ \text{variable} = \text{value}; \]

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

- For the list at right:
  - \texttt{a.next = value;}  
    means to adjust where \texttt{a} points
  - \texttt{variable = a.next;}  
    means to make \texttt{variable} point at

```
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>1</td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
```
Linked node problem 4

- What set of statements turns this picture:

```
list1  data  next
  1      .
```

```
list2  data  next
  2     /
```

- Into this?

```
list1  data  next
  1      .
```

```
list2  data  next
  2     /
```

```data  next
  3      \n```