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 a stack and a queue](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:
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:

```
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>
```

• Into this?

```
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td></td>
</tr>
<tr>
<td>20</td>
<td></td>
</tr>
<tr>
<td>30</td>
<td></td>
</tr>
</tbody>
</table>
```
Linked node problem 2

- What set of statements turns this picture:

```
list []
```

```
data 10
next
```

```
data 20
```

- Into this?

```
list []
```

```
data 30
next
```

```
data 10
```

```
data 20
```
• Suppose we had the following `ListNodes`:

```
<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
</tbody>
</table>
```

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

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

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

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

![Diagram showing data and next values before and after reassignment]
References vs. objects

variable = value;

a variable (left side of = ) 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 3

- What set of statements turns this picture:

```
   list1[ ]  ⇒  data  next  ⇒  list2[ ]
       10        20
       30        40
```

- Into this?

```
   list1[ ]  ⇒  data  next  ⇒  list2[ ]
       10        20        30
       40
```
Linked node problem 3

- How many ListNode variables?

- Which variables change?
Linked node problem 3

- How many ListNode variables?

- Which variables change?

list1\[A\] → data \(10\) \(\rightarrow\) data \(20\)

list2\[D\] → data \(30\) \(\rightarrow\) data \(40\)

list1.next.next = list2

list1\[A\] → data \(10\) \(\rightarrow\) data \(20\) \(\rightarrow\) data \(30\)

list2\[D\] → data \(40\) \(\rightarrow\)
Linked node problem 3

- How many ListNode variables?

- Which variables change?

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

- How many ListNode variables?

- Which variables change?

```
list1.next.next = list2
list2 = list2.next
list1.next.next.next = null
```
Linked node problem 4

- What set of statements turns this picture:

```
list1

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

list2

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

- Into this?

```
list1

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

list2

<table>
<thead>
<tr>
<th>data</th>
<th>next</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
</tr>
</tbody>
</table>
```