4.3 Stacks, Queues, and Linked Lists
Data Types and Data Structures

**Data types.** Set of values and operations on those values.
- Some are built into the Java language: `int`, `double[]`, `String`, ...
- Most are not: `Complex`, `Picture`, `Stack`, `Queue`, `ST`, `Graph`, ...

**Data structures.**
- Represent data or relationships among data.
- Some are built into Java language: arrays.
- Most are not: linked list, circular list, tree, sparse array, graph, ...

this lecture
Collections

Fundamental data types.

- Set of operations (add, remove, test if empty) on generic data.
- Intent is clear when we insert.
- Which item do we remove?

Stack. [LIFO = last in first out]

- Remove the item most recently added.
- Ex: Pez, cafeteria trays, Web surfing.

Queue. [FIFO = first in, first out]

- Remove the item least recently added.
- Ex: Guitar Hero (RingBuffer)

Symbol table.

- Remove the item with a given key.
- Ex: Phone book.
Linked Lists

Official Florida Presidential Ballot
Follow the arrow and Punch the appropriate dot.

Bush
Buchanan
Gore
Nader

(c) 2000 Mike Collins, Taterbrains.com
Sequential vs. Linked Allocation

Sequential allocation. Put items one after another.
- TOY: consecutive memory cells.
- Java: array of objects.

Linked allocation. Include in each object a link to the next one.
- TOY: link is memory address of next item.
- Java: link is reference to next item.

Key distinctions.
- Array: random access, fixed size.
- Linked list: sequential access, variable size.

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>B0</td>
<td>&quot;Alice&quot;</td>
</tr>
<tr>
<td>B1</td>
<td>&quot;Bob&quot;</td>
</tr>
<tr>
<td>B2</td>
<td>&quot;Carol&quot;</td>
</tr>
<tr>
<td>B3</td>
<td>-</td>
</tr>
<tr>
<td>B4</td>
<td>-</td>
</tr>
<tr>
<td>B5</td>
<td>-</td>
</tr>
<tr>
<td>B6</td>
<td>-</td>
</tr>
<tr>
<td>B7</td>
<td>-</td>
</tr>
<tr>
<td>B8</td>
<td>-</td>
</tr>
<tr>
<td>B9</td>
<td>-</td>
</tr>
<tr>
<td>BA</td>
<td>-</td>
</tr>
<tr>
<td>BB</td>
<td>-</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>addr</th>
<th>value</th>
</tr>
</thead>
<tbody>
<tr>
<td>C0</td>
<td>&quot;Carol&quot;</td>
</tr>
<tr>
<td>C1</td>
<td>null</td>
</tr>
<tr>
<td>C2</td>
<td>-</td>
</tr>
<tr>
<td>C3</td>
<td>-</td>
</tr>
<tr>
<td>C4</td>
<td>&quot;Alice&quot;</td>
</tr>
<tr>
<td>C5</td>
<td>CA</td>
</tr>
<tr>
<td>C6</td>
<td>-</td>
</tr>
<tr>
<td>C7</td>
<td>-</td>
</tr>
<tr>
<td>C8</td>
<td>-</td>
</tr>
<tr>
<td>CA</td>
<td>&quot;Bob&quot;</td>
</tr>
<tr>
<td>CB</td>
<td>C0</td>
</tr>
</tbody>
</table>
Singly-Linked Data Structures

From the point of view of a particular object:
all of these structures look the same.

Multiply-linked data structures. Many more possibilities.
Linked Lists

Linked list.
- A recursive data structure.
- An item plus a pointer to another linked list (or empty list).
- Unwind recursion: linked list is a sequence of items.

Node data type.
- A reference to a String.
- A reference to another Node.

```java
public class Node {
    public String item;
    public Node next;
}
```

first

Alice → Bob → Carol → null

item next

special pointer value null terminates list
Building a Linked List

Node third = new Node();
third.item = "Carol";
third.next = null;

Node second = new Node();
second.item = "Bob";
second.next = third;

Node first = new Node();
first.item = "Alice";
first.next = second;
Stack Push (Insert at Front): Linked List Implementation

```
Node second = first;
first = new Node();
first.item = "of";
first.next = second;
```
Stack Pop (Remove from Front): Linked List Implementation

```
String item = first.item;
first = first.next;
return item;
```
public class LinkedStackOfStrings {
    private Node first = null;

    private class Node {
        private String item;
        private Node next;
    }

    public boolean isEmpty() { return first == null; }

    public void push(String item) {
        Node second = first;
        first = new Node();
        first.item = item;
        first.next = second;
    }

    public String pop() {
        String item = first.item;
        first = first.next;
        return item;
    }
}
Two data structures to implement stack data type.

Array.
- Every push/pop operation take constant time.
- But... must fix maximum capacity of stack ahead of time.

Linked list.
- Every push/pop operation takes constant time.
- Memory is proportional to number of items on stack.
- But... uses extra space and time to deal with references.
Q. What does the following code fragment do?

```java
for (Node x = first; x != null; x = x.next) {
    StdOut.println(x.item);
}
```
Q. What does the following code fragment do?

```java
Node last = new Node();
last.item = StdIn.readString();
last.next = null;
Node first = last;
while (!StdIn.isEmpty()) {
    last.next = new Node();
    last = last.next;
    last.item = StdIn.readString();
    last.next = null;
}
```