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`, ...

this lecture

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

Section 4.3
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: Line for help in TA office hours.

Symbol table:
• Remove the item with a given key.
• Ex: Phone book.

this lecture

Guitar Hero Assignment

Section 4.3
Stack API

```
public class StackOfStrings

    StackOfStrings()  // create an empty stack

    boolean isEmpty()  // is the stack empty?

    void push(String item)  // push a string onto the stack

    String pop()  // pop the stack
```

Section 4.3
Stack Client Example 1: Reverse

```java
public class Reverse {
   public static void main(String[] args) {
      StackOfStrings stack = new StackOfStrings();
      while (!StdIn.isEmpty()) {
         String s = StdIn.readString();
         stack.push(s);
      }
      while (!stack.isEmpty()) {
         String s = stack.pop();
         StdOut.println(s);
      }
   }
}
```

% more tiny.txt
it was the best of times

% java Reverse < tiny.txt
times of best the was it

stack contents when standard input is empty
Linked Lists

Section 4.3

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

Singly-Linked Data Structures
sequential (this lecture)
parent-link tree
rho
circular
general case

Multiply-linked data structures: Many more possibilities.

Section 4.3
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 item next)

special pointer value `null` terminates list
Building a Linked List

```java
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;
```

```
<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>C9</td>
<td>-</td>
</tr>
<tr>
<td>CA</td>
<td>&quot;Bob&quot;</td>
</tr>
<tr>
<td>CB</td>
<td>C0</td>
</tr>
<tr>
<td>CC</td>
<td>-</td>
</tr>
<tr>
<td>CD</td>
<td>-</td>
</tr>
<tr>
<td>CE</td>
<td>-</td>
</tr>
<tr>
<td>CF</td>
<td>-</td>
</tr>
</tbody>
</table>
```

The diagram illustrates the nodes and their connections in a linked list, with the values stored in each node and their memory addresses in the main memory.
Stack Push: Linked List Implementation

```java
Node second = first;
first.item = "of";
first.next = second;
first = new Node();
```

Section 4.3
Stack Pop: Linked List Implementation

```
first = first.next;
return item;
```
Stack: Linked List Implementation

```java
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;
    }
}
```

"inner class"

stack and linked list contents after 4th push operation
Linked List Stack: Test Client Trace

push

be
or
not
to

be
or
not
to

not
or
be
to
null

to
no

be
not
or
be
to
null

be
not
or
be
to
null

that
or
be
to
null

that
or
be
to
null

or
be
to

be
to

is
to
Stack Data Structures: Tradeoffs

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.
List Processing Challenge 1

What does the following code fragment do?

```java
for (Node x = first; x != null; x = x.next) {
    StdOut.println(x.item);
}
```

Alice — Bob — Carol — null
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;
}
```

First, last is set to a new Node with an empty string as its item. Then, while input is not empty, a new Node is created, followed by setting last to the new Node, and then reading a new string and setting it as the item of the last node. This continues until there is no input left.

Diagram:
- First node is Alice
- Next node is Bob
- Next node is Carol
- Next node is null

Section 4.3