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, ...

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.

Section 4.3
Stack API

```java
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
}
```

![Diagram of stack operations]

**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
Singly-Linked Data Structures

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

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;
}
```
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>
Stack Push: Linked List Implementation

first
   best → the → was → it

first    second
       best → the → was → it
       Node second = first;

first    second
       first = new Node();

first    second
       first.item = "of";
       first.next = second;

Section 4.3
Stack Pop: Linked List Implementation

```java
String item = first.item;
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;
    }
    "inner class"

    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;
    }
}
```

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

Section 4.3
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);
}
```

```
first

Alice  →  Bob  →  Carol  →  null

item  next
```
List Processing Challenge 2

What does the following code fragment do?

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;
}

Section 4.3