CS310
Data Structures

K. Raven Russell
krusselc@gmu.edu
George Mason University

Today

- **Last Lecture**
  - Stacks and Queues
- **Today**
  - More Stacks and Queues

What will these methods do?
Try tracing with linked list:
"A" → "B" → "C"

```
GIVEN:

class Node<T> {
    T data;
    Node<T> next;
}

class Stack<T> {
    void push(T data) {…}
    T pop() {…}
    int size() {…}
}
```

<drawing area>

PROBLEM 1:

```
void method1(Node<?> c) {
    if(c == null) return;
    System.out.println(c.data);
    method1(c.next);
}
```

PROBLEM 2:

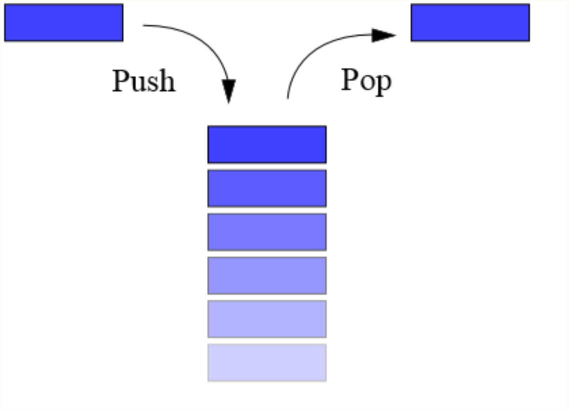
```
void method2(Node<?> c) {
    if(c == null) return;
    method2(c.next);
    System.out.println(c.data);
}
```

PROBLEM 3:

```
void method3(Node<?> c) {
    Stack<Node<?>> stack
        = new Stack<Node<?>>();
    while(c != null) {
        stack.push(c);
        c = c.next;
    }
    while(stack.size() > 0) {
        System.out.println(stack.pop().data);
    }
}
```

Stacks and Queues

- Stack
  - Data structure that works like a... stack
  - e.g. a stack of paper
- Queue
  - Data structure that works like a... queue
  - or a “line” if you aren't British



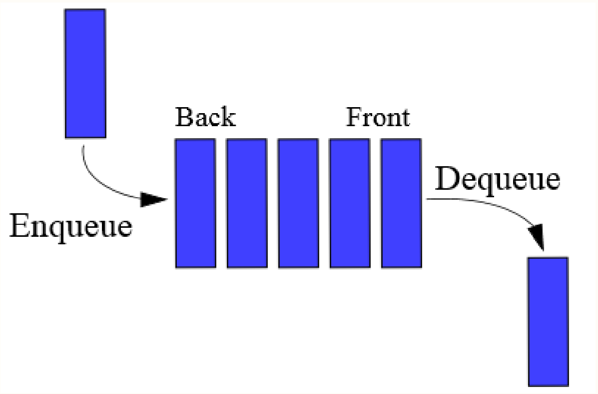


Image Source: http://en.wikipedia.org/wiki/File:Data_stack.svg and http://en.wikipedia.org/wiki/File:Data_Queue.svg

Stacks and Queues Summary

– Typical to implement Stack using an Array or Dynamic List (good constraints)

– Typical to implement Queue using Singly Linked List

Stack

Operation Implementation	push	pop	peek	isEmpty	size
Dynamic Array	1*	1	1	1	1
Linked List	1	1	1	1	1

Queue

Operation Implementation	enqueue	dequeue	peek	isEmpty	size
Dynamic Array	1*	1	1	1	1
Linked List	1	1	1	1	1

* Amortized

Questions?

New Material

Data Structure: Priority Queue

- Queue
  - first item in = first item out (FIFO)
- Priority Queue
  - highest priority item = first item out
- Options
  - unsorted list
  - sorted list
  - multiple queues
  - heap (not covered today… we’ll get to this later in the semester)

Whiteboard

Unsorted List

- Enqueue
  - add to one end of the list
  - choose which end carefully!
    - *We want an O(1) add for our underlying data structure*
- Dequeue
  - search list for highest priority item and remove
  - shift later items in the list down (if array)

- Underlying structures:
  - array
  - linked list

Sorted List

- Enqueue
  - add to end of the list and swap until in place
  - can be done like one step of an insertion sort
- Dequeue
  - remove the front/back of the list
  - choose which end carefully!
    - *We want an O(1) remove for our underlying data structure*

- Underlying structures:
  - array (circular or highest priority at back end)
  - linked list

Multiple Queues

- Have one queue per priority level
  - fixed number of priorities! (e.g. high/medium/low)
- Enqueue
  - enqueuer in correct priority's queue
- Dequeue
  - dequeue from highest priority (non-empty) queue

- Easy implementations:
  - array of (circular) array queues
  - array of linked list queues

Priority Queue Summary

– n = number of items put in the queue

– m = number of priorities

Operation Implementation	enqueue	dequeue (the highest priority item)	peek (at the highest priority item)
Unordered List	?	?	?
Ordered List	?	?	?
Multiple Queues	?	?	?
Binary Heap**	?	?	?

* Covered later this semester

Priority Queue Summary

– n = number of items put in the queue

– m = number of priorities

Operation Implementation	enqueue	dequeue (the highest priority item)	peek (at the highest priority item)
Unordered List	O(1)	O(n)	O(n)
Ordered List	O(n)	O(1)	O(1)
Multiple Queues	O(1)	O(m)	O(m)
Binary Heap**	O(lg n)	O(lg n)	O(1)

* Covered later this semester

Interview Questions from Data Structures

Interview Question 1:
Build a Queue and a Stack using only a List (Linked or Array-based)

Interview Question 2:
Build a Queue using only Stacks

Interview Question 3:
Design a special stack which has the following O(1) operations:
push(), pop(), **min()**

Interview Question 4:
Describe an algorithm to **sort a stack** in ascending order using only a second stack and a temporary variable. Do not make any assumptions about how the stack is implemented. The only functions available for the stack are: push(), pop(), peek(), and isEmpty()

Project 2