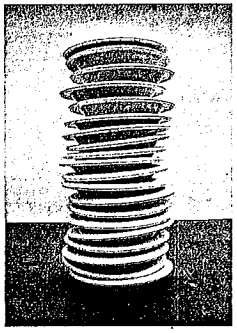
CompSci 100e
Program Design and Analysis II

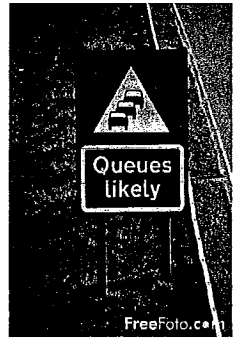


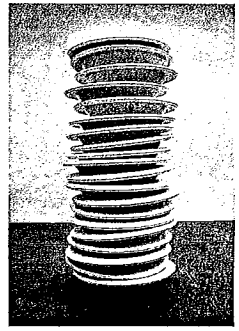
March 15, 2011

Prof. Rodger

Announcements

- APTs extended til 3/17 (Typing Job, Rat route hints)
- New APT set out today due March 22
- Today Stacks and Queues
  - Linear structures used in problem domains and algorithms





Why don't we just use arrays?

- Stacks used in implementing recursion, postscript language, Java language, graph algorithms
  - Stacks implemented using array/ArrayList
- Queues used in simulation, graph algorithms, scheduling
  - Queues implemented using array/LinkedList

Simple stack example

- `Stack` is part of java.util.Collections hierarchy
  - As an ADT it's a LIFO data structure (last in, first out)
  - what does pop do? What does push do?

```
Stack<String> s = new Stack<String>();
s.push("panda");
s.push("grizzly");
s.push("brown");
System.out.println("size = "+s.size());
System.out.println(s.peek());
String str = s.pop();
System.out.println(s.peek());
System.out.println(s.pop());
```

Postfix, prefix, and infix notation

- Postfix notation used in some HP calculators
  - No parentheses needed, precedence rules still respected

```
3 5 +          4 2 * 7 + 3 -          9 7 +
  *
```

  - Read expression
    - For number/operand: push
    - For operator: pop, pop, operate, push
- See *Postfix.java* for example code, key ideas:
  - Use StringTokenizer, handy tool for parsing
  - Note: Exceptions thrown, what are these?
- What about prefix and infix notations, advantages?

Interlude: Exceptions

- Exceptions are *raised* or *thrown* in exceptional cases
  - Bad indexes, null pointers, illegal arguments, ...
  - File not found, URL malformed, ...
- Runtime exceptions aren't handled or *caught*
  - Bad index in array, don't try to handle this in code
  - Null pointer stops your program, don't code that way!
- Some exceptions are caught or rethrown
  - `FileNotFoundException` and `IOException`
- RuntimeException extends Exception
  - catch not required

Simple queue example

- `Queue` is part of java.util.Collections hierarchy
  - As an ADT it's a FIFO data structure (first in, first out)
  - what does add do? What does remove do?

```
Queue<String> s = new Queue<String>();
s.add("panda");
s.add("grizzly");
s.add("brown");
System.out.println("size = "+s.size());
System.out.println(s.peek());
String str = s.remove();
System.out.println(s.peek());
System.out.println(s.remove());
```


Analysis

- Assume a Stack is implemented with an ArrayList. How are inserts and removes done?
  - Inserting N elements into a stack takes how long?
  - Removing N removes from a stack takes how long?
- Assume a Queue is implemented with an ArrayList. How are inserts and removes done?
  - Inserting N elements into a queue takes how long?
  - Removing N elements from a queue takes how long?