Questions From Last Time

- Can we include implementation details in the inside comments? Yes, but not in the method headers.
- When do I use static methods? If you want to write a method that doesn't use a particular instance of the class, it should be static.
  
  ```java
  // This method doesn't use a particular instance; it is a property
  public static int numberOFArraysCreated() {}  
  ```

- What if I'm completely lost in lecture? Come to office hours; I'm happy to explain the entire lecture again. Also, raise your hand for clarifications!
- I didn't have enough space to answer all the questions that were asked. Feel free to come up afterwards/at office hours to get the other questions answered.

Stacks

**Stack**

A stack is a collection which orders the elements last-in-first-out ("LIFO"). Note that, unlike lists, stacks do not have indices.

- Elements are stored internally in order of insertion.
- Clients can ask for the top element (pop/peek).
- Clients can ask for the size.
- Clients can add to the top of the stack (push).
- Clients may only see the top element of the stack.

<table>
<thead>
<tr>
<th>Client: Impl</th>
<th>Impls:</th>
<th>Client: Impl</th>
<th>Impls:</th>
<th>Client: Impl</th>
</tr>
</thead>
<tbody>
<tr>
<td>↓↑</td>
<td>↓↑</td>
<td>↓↑</td>
<td>↓↑</td>
<td>↓↑</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>1</td>
<td></td>
<td>9</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
</tbody>
</table>

Okay; Wait; Why?

A stack seems like what you get if you take a list and remove methods.

Well...yes...

- This prevents the client from doing something they shouldn't.
- This ensures that all valid operations are fast.
  
  - `add(idx, val)`: $O(n)$
  - `remove(idx)`: $O(n)$
  - `push(val)`: $O(1)$
  - `pop()`: $O(1)$

- Having Fewer operations makes stacks easy to reason about.
**Abstract Data Types (ADT)**

An abstract data type is a description of what a collection of data can do. We usually specify these with interfaces.

### List ADT

In Java, a List can add, remove, size, get, set.

### List Implementations

An ArrayList is a particular type of List. Because it is a list, we promise it can do everything a List can. A LinkedList is another type of List.

Even though we don’t know how it works, we know it can do everything a List can, because it’s a List.
Queue Reference

Queue is an interface. So, you create a new Queue with:

```java
Queue<Integer> queue = new LinkedList<Integer>();
```

<table>
<thead>
<tr>
<th>Method</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>add(val)</td>
<td>Adds val to the back of the queue</td>
</tr>
<tr>
<td>remove()</td>
<td>Removes the first value from the queue; throws a NoSuchElementException if the queue is empty</td>
</tr>
<tr>
<td>peek()</td>
<td>Returns the first value in the queue without removing it; returns null if the queue is empty</td>
</tr>
<tr>
<td>size()</td>
<td>Returns the number of elements in the queue</td>
</tr>
<tr>
<td>isEmpty()</td>
<td>Returns true if the queue has no elements</td>
</tr>
</tbody>
</table>

Applications Of Queues

- Queue of print jobs to send to the printer
- Queue of programs / processes to be run
- Queue of keys pressed and not yet handled
- Queue of network data packets to send
- Queue of button/keyboard/etc. events in Java
- Modeling any sort of line
- Queuing Theory (subfield of CS about complex behavior of queues)

War (the card game)

War is played with a standard 52 card deck.

1. The deck is shuffled.
2. The deck is completely dealt out among players.
3. Both players place down a card.
4. If the cards have equal value, go back to step 3. Otherwise, the player with the higher card appends all the cards to her deck.
5. Play continues until someone runs out of cards.

Let's Write Code for War!