6-29: Stacks and Queues

Back to being a client! We'll be going back to using data structures that are in the Java language.

Abstract Data Type: A specification of a collection of data and the operations that can be performed on it. Describes **what** a collection does, not **how** it does it

What do you want a phone to be able to do?

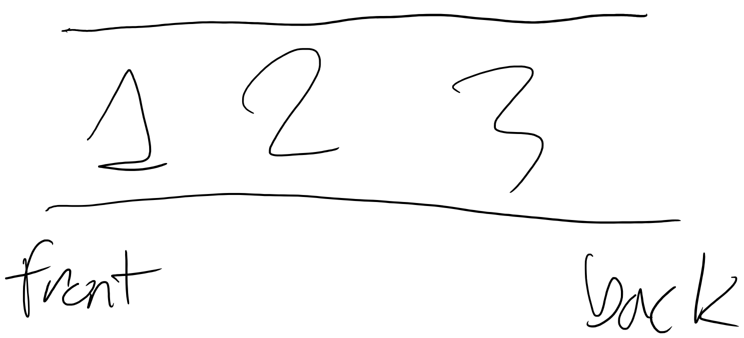
- Call people
- Receive calls
- Text people
- Turn on and off
- Change volume
- Save phone numbers

Java Interfaces create these Abstract Data Types for us!

"You should favor the use of interfaces over classes to refer to objects.
If appropriate interface types exist, then parameters, return values,
variables and fields should all be declared using interface types.
The only time you really need to refer to an object's class is when you're
creating it with a constructor."
-Joshua Bloch

Queue:
FIFO (first in, first out) data structure

Queue<E> q = new LinkedList<E>()	- makes an empty queue
public void add(E value)	- adds to the back of the queue
public E remove()	- removes from the front of the queue
public boolean isEmpty()	- true if empty, false otherwise
public int size()	- number of elements in the queue



Stack:
LIFO (last in, first out) data structure

Stack<E> s = new Stack<E>()	- makes an empty stack
public void push(E value)	- adds to the top of the stack
public E pop()	- removes from the top of the stack
public boolean isEmpty()	- true if empty, false otherwise
public int size()	- number of elements in the stack