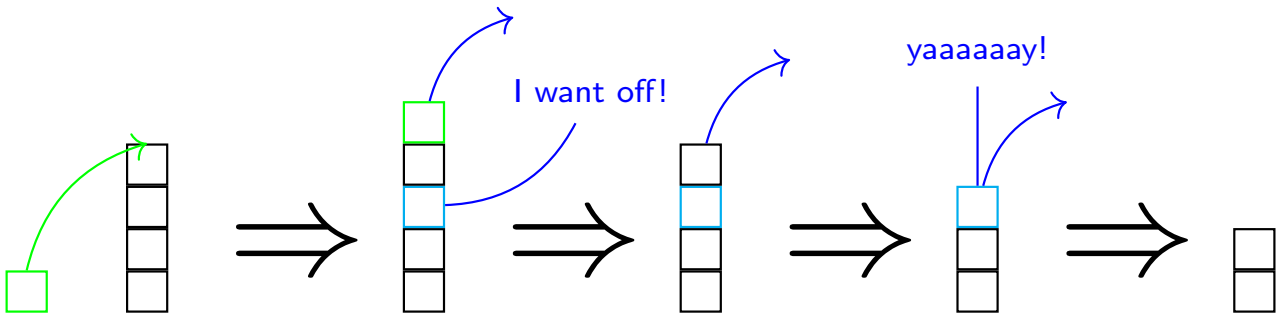
Stacks and Queues

Specialized Lists

- Sometimes, it is useful to restrict the capabilities of an ADT to get certain behavior
- We will look at two specializations of lists that restrict how data enters and leaves the structure

Stacks

- ADT where data can be accessed from only one end
  - Think of it like a stack of books; to get to the second or third book in the stack, you must remove the books on top of it.
- Known as a LIFO or FILO Data Structure
  - Last In First Out
  - First In Last Out



Stack Operations

- `push(Element e)`
  - Puts `e` on top of the stack
- `pop( )`
  - Removes the top element from the stack (and returns it)
- top()
  - Returns the top object on the stack
  - Called `peek( )` in the JCF

Implementation Options

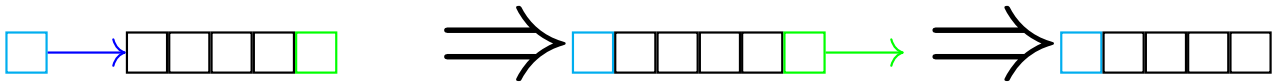
- Because `push` and `pop` modify the *same* end of the list, the choices are:
  - `push` = add at front, `pop` = remove from front
  - `push` = add at back, `pop` = remove from back
- Recall the characteristics of the ArrayList and the LinkedList

	+/- @ Front	+/- @ Back
ArrayList	slow	fast
LinkedList	fast	fast

- Either choice is appropriate!
- Java uses a `Vector` (a thread-safe ArrayList) and extends it (instead of having a `Vector` instance variable)
  - This allows the computer to use *cache* effectively, a speedup you will learn more about in Operating Systems

Queues

- The most English of Data Types
  - Queue is what the English call lines
- Data enters the rear of the queue, and exits the front of the queue
- FIFO
  - First In First Out



Queue Operations

- `enqueue (Element e)`
  - Puts e at the end of the queue
- `dequeue( )`
  - Removes the element from the top of the line
- `front( )` or `peek( )`
  - Returns value at front of the line

Implementation Options

- Because `push` and `pop` modify the *different* ends of the list, the choices are:
  - `push` = add at front, `pop` = remove from back
  - `push` = add at back, `pop` = remove from front
- Recall the characteristics of the ArrayList and the LinkedList

	+ @ Front	- @ Front	+ @ Back	- @ back
ArrayList	slow	slow	fast	fast
Singly LinkedList	fast	fast	fast	slow
Doubly LinkedList	fast	fast	fast	fast

- An ArrayList would be a poor choice; all operations at the front are slow
- Either a Singly or a Doubly LinkedList are appropriate choices.
- Java offers a `Queue` interface that `LinkedIst` extends
  - `add(Element e)`
  - `remove()`
  - `element()`

Applications of Stacks and Queus

- Stacks
  - Reversing input
  - Parenthesis matching
  - Basis for our entire computational model
- Queues
  - Holds data to process in order