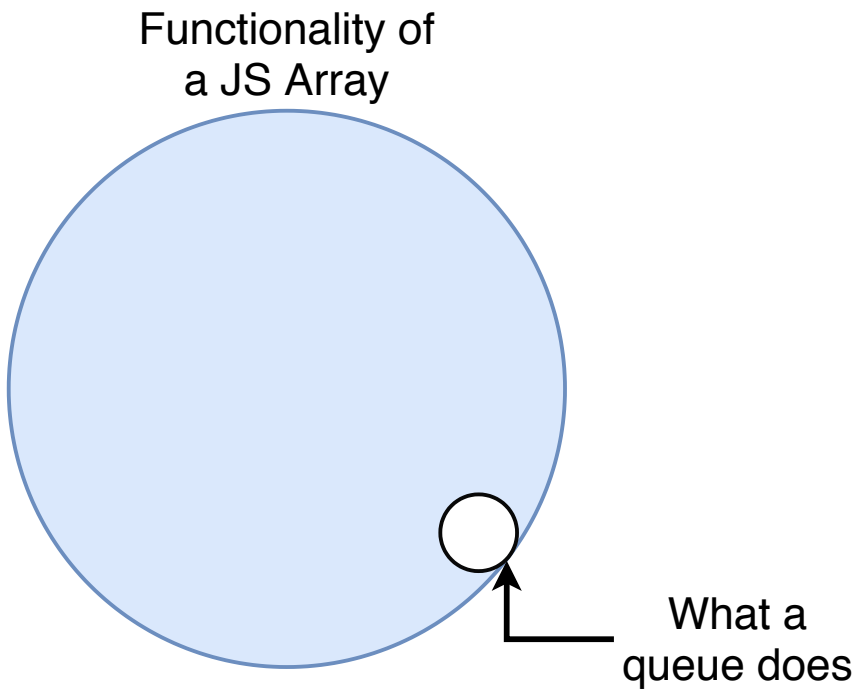
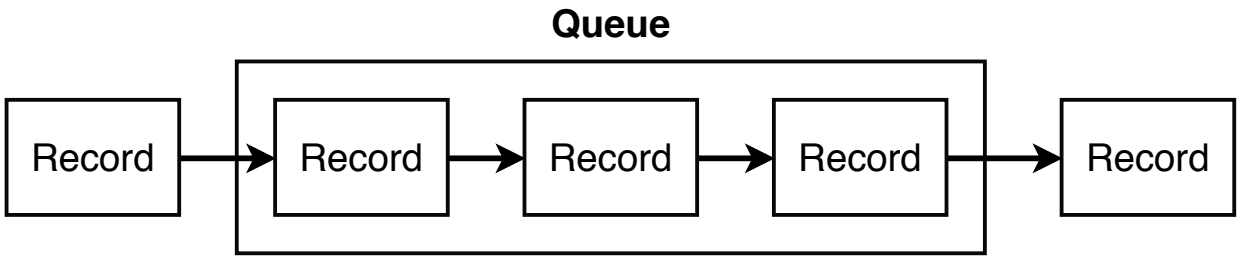
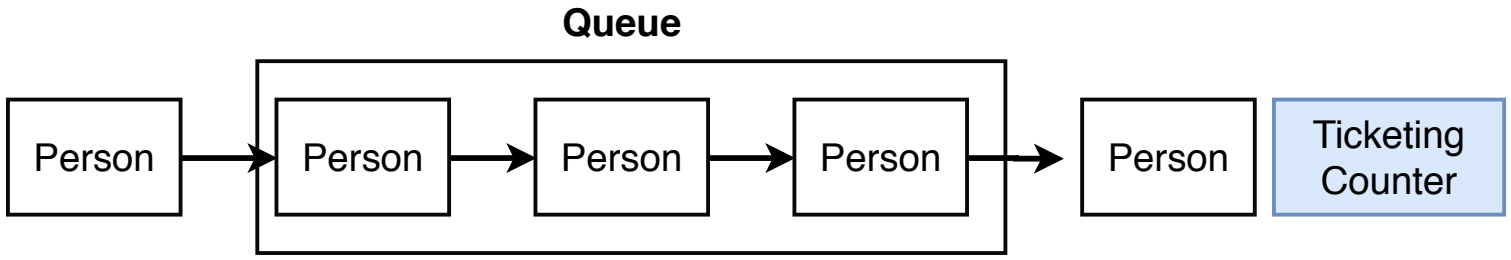
Data Structures

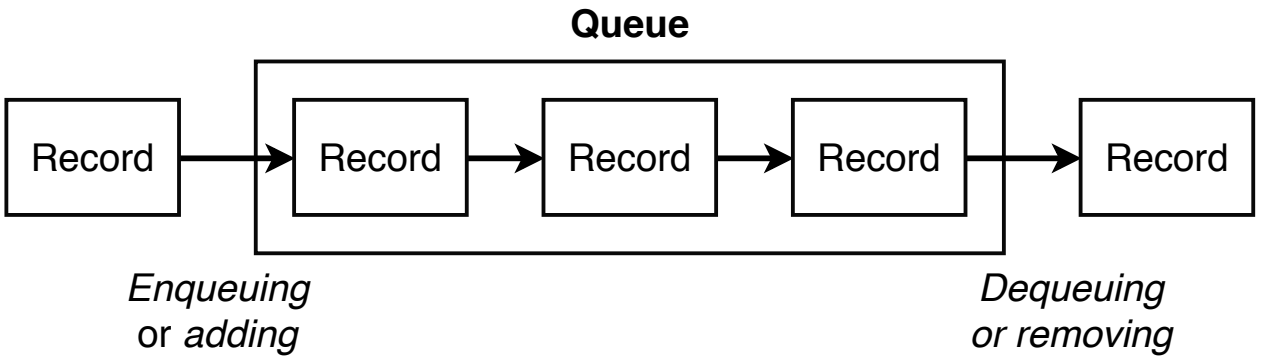
Ways of organizing information with optimal 'runtime complexity' for adding or removing records

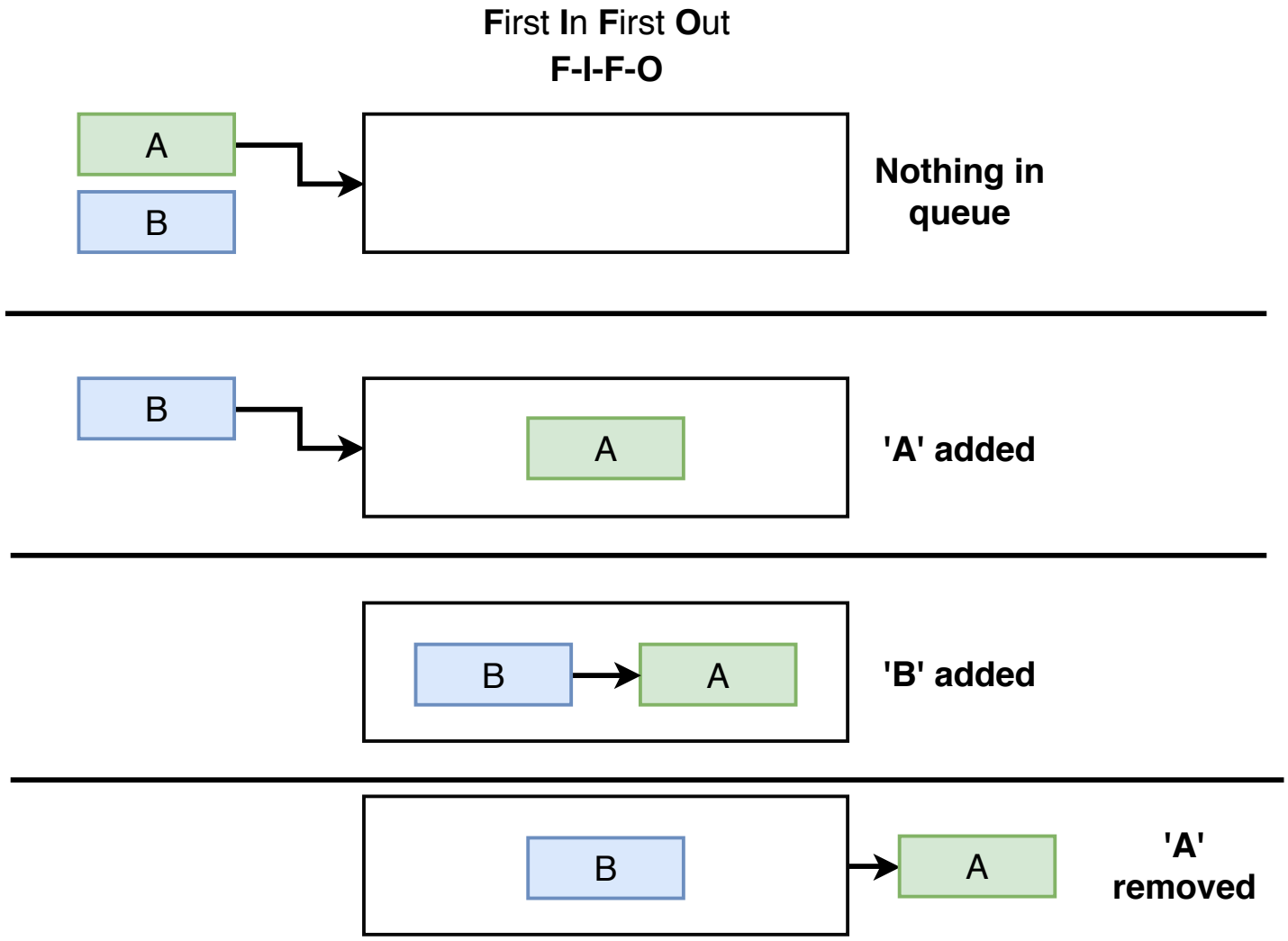
Javascript natively implements several data structures. **You will still be asked about 'inferior' data structures.**


Functionality of
a JS Array
What a
queue does

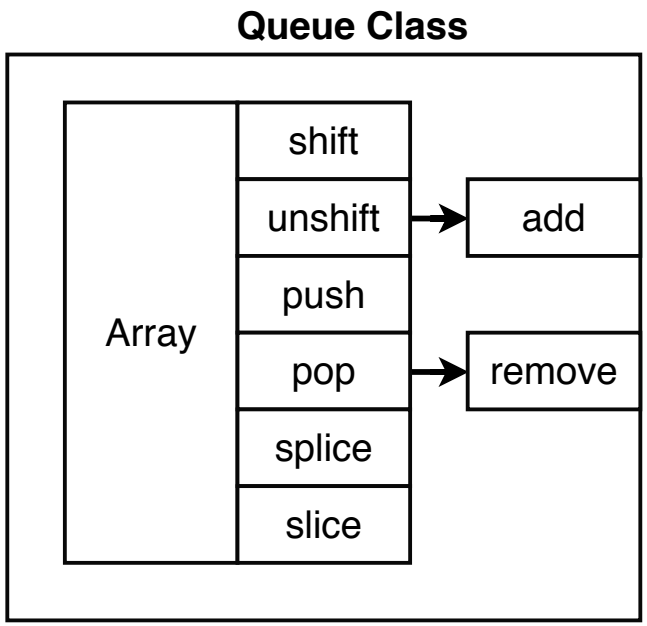

Queue
Record
Record
Record
Record
Record


Queue
Person
Person
Person
Person
Person
Ticketing Counter

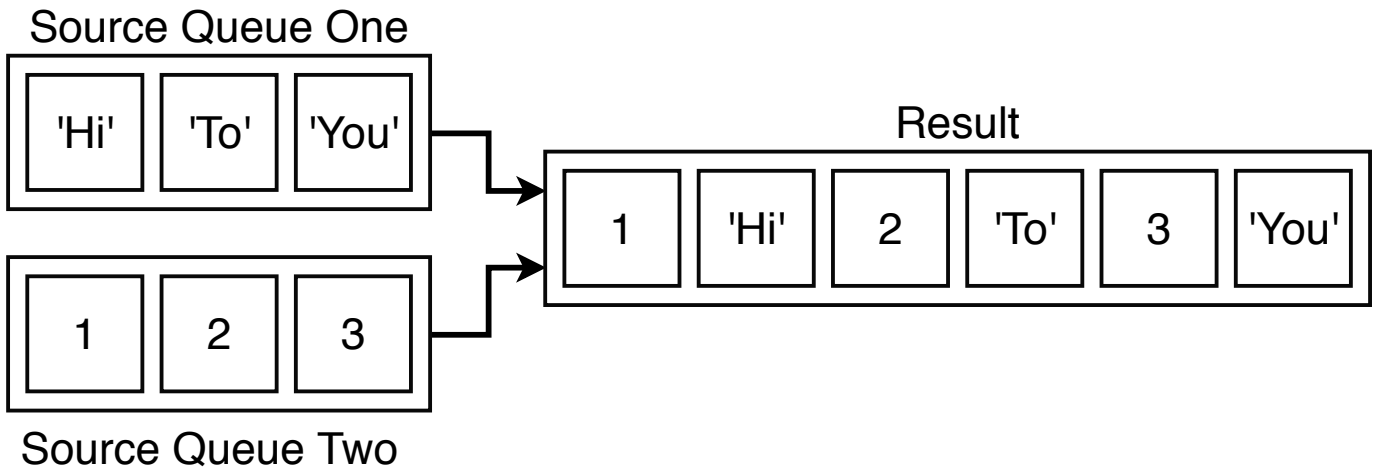

Queue
Record
Record
Record
Record
Record
Enqueuing
or adding
Dequeuing
or removing

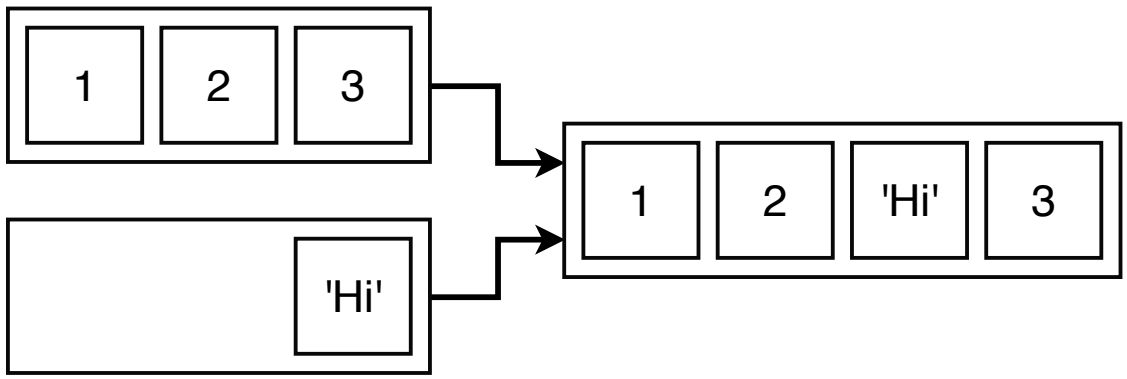

First In First Out
F-I-F-O
A
B
Nothing in queue
B
A
'A' added
B
A
'B' added
B
A
'A' removed

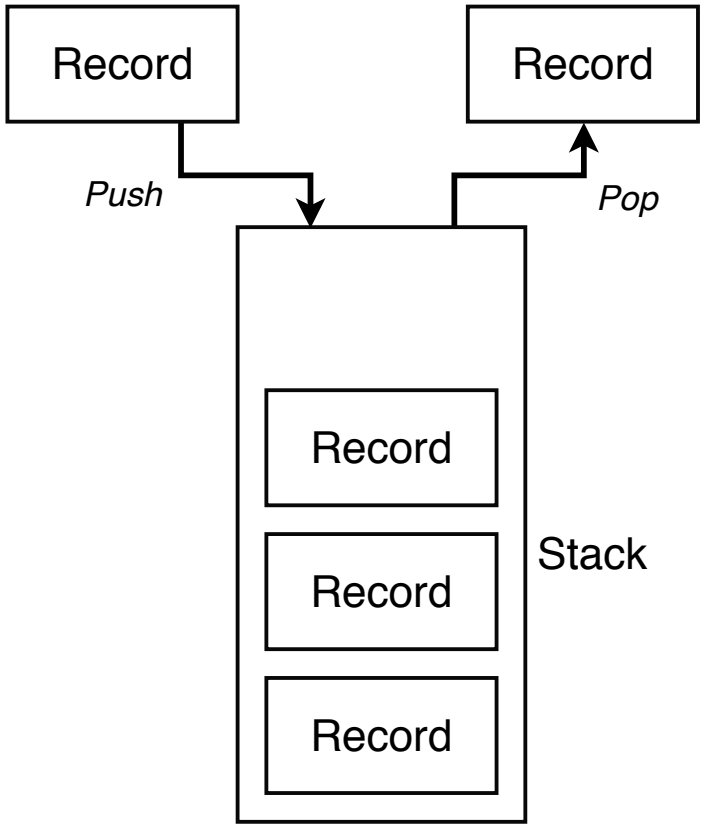
Queue	Array Equivalent
Add to queue	array.unshift();
Remove from queue	array.pop();

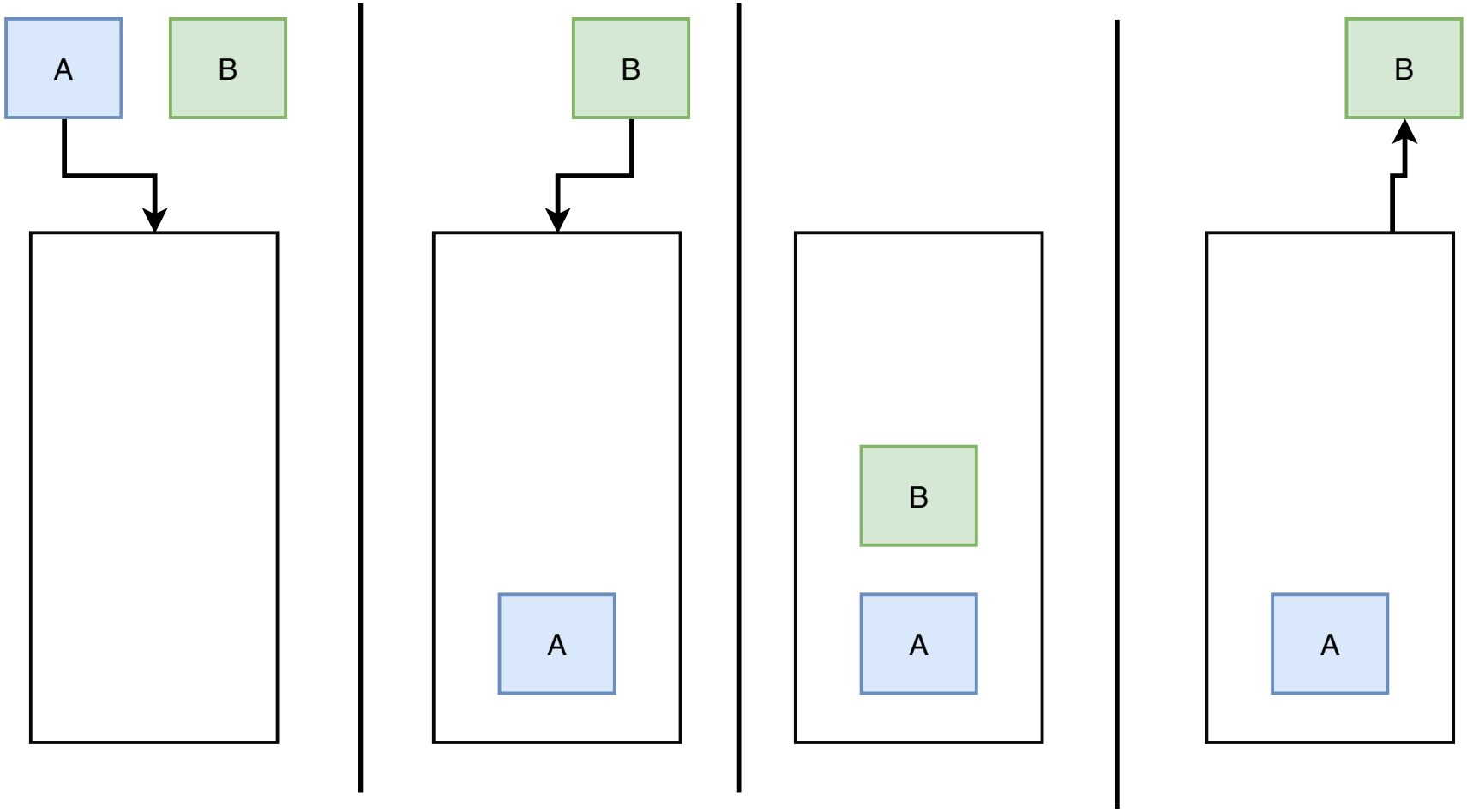

Queue Class
Array
shift
unshift
push
pop
splice
slice
add
remove

To...	Run This
Create a new, empty queue	const q = new Queue();
Add a record to a queue	q.add(1);
Remove record at the end of a queue	q.remove();


Source Queue One
'Hi'
'To'
'You'
1
2
3
Source Queue Two
Result
1
'Hi'
2
'To'
3
'You'

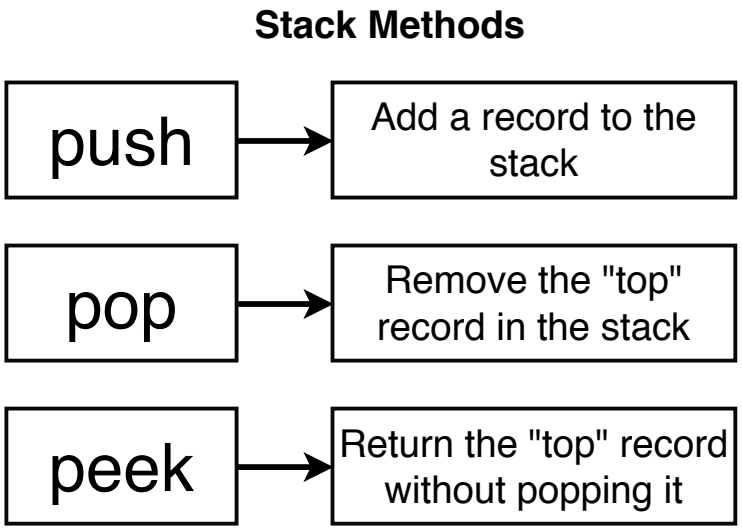

1
2
3
'Hi'
1
2
'Hi'
3

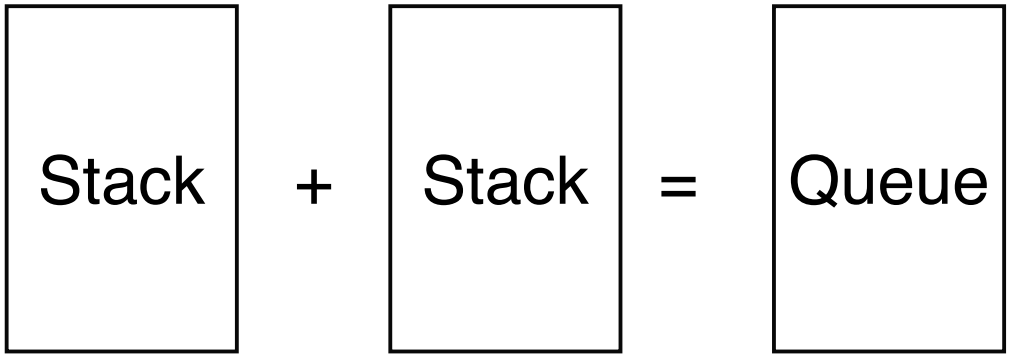

Record
Record
Push
Pop
Record
Record
Record
Stack



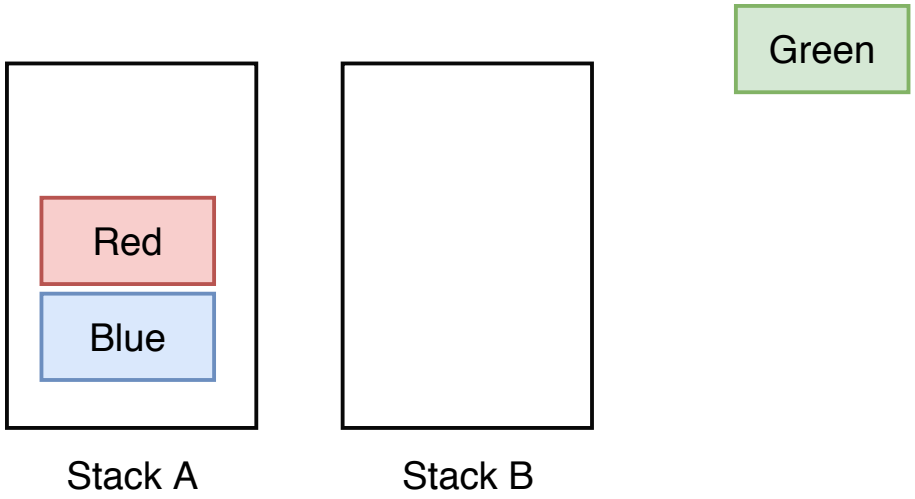
First **I**n **L**ast **O**ut

FILO


Stack Methods
push
Add a record to the stack
pop
Remove the "top" record in the stack
peek
Return the "top" record without popping it

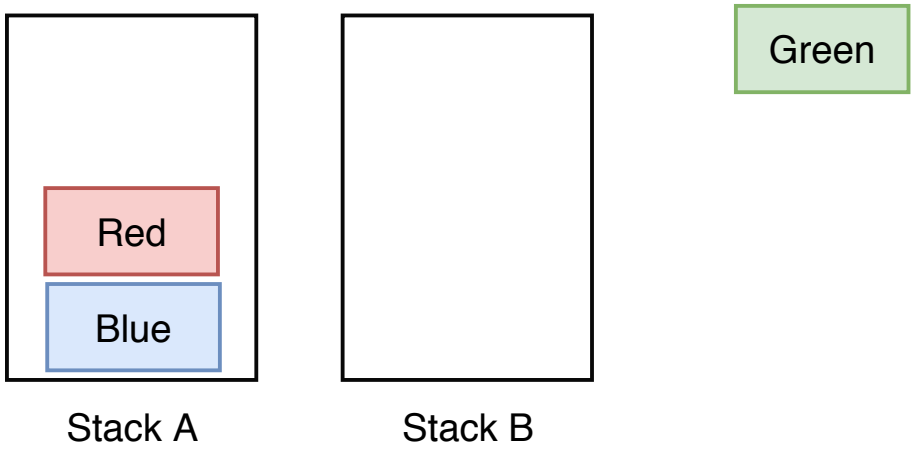

Stack
+
Stack
=
Queue

Add



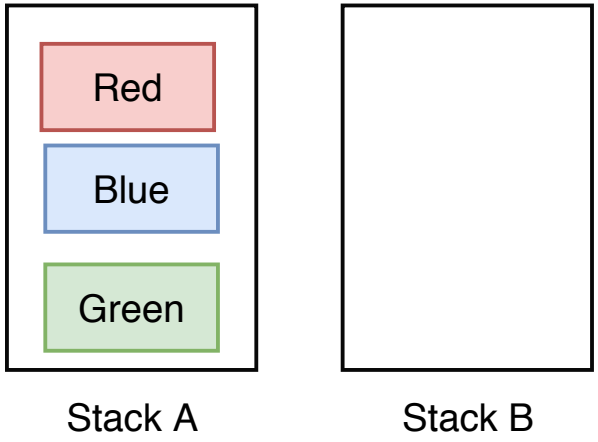
Remember, we want a Queue, so follow 'First In First Out'

Remove



Remember, we want a Queue, so follow 'First In First Out'

Peek



Remember, we want a Queue, so follow 'First In First Out'