Stacks and Queues

- Resources available:
  kevin-miao.com

Today

- Extremely Short Review: Collections
- Quiz Review
- Worksheet

Starting Stack: Top → B - D - E - F

1. pop
2. pop
3. push A
4. push G
5. pop
6. push C
7. pop

Q2.1
0.25 Points

What is left in the stack after the following calls have been made?

Please specify your answer using uppercase letters with no spaces in between them ordered from top to bottom. For example if you think the remaining elements in order are A, B, C, D, with A on top and D on the bottom then your answer should be ABCD.

Q2.2
0.25 Points

What are the elements popped off of the stack?

Please specify your answer using uppercase letters with no spaces in between them ordered from the first item popped to the last item popped. For example if you think the elements popped in order were A, B, C, D, with A being popped first and D being popped last then your answer should be ABCD.

First in
Last out

Stack
+ Push
+ Pop

Pancake Stack

First in
First out

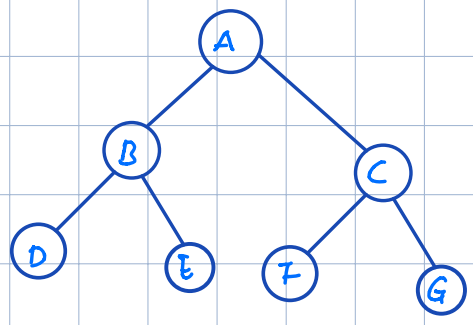
Queue
+ push
+ pop

Queue

- Why are we learning about Stacks and Queue?

Fringe for tree traversals.

(universal term)



+ DFS = Stack
+ ABDE CFG

+ BFS = Queue
+ A B C D E F G

4 When am I Useful Senpai?

Based on the description, choose the data structure which would best suit our purposes. Choose from: **A - arrays, B - linkedlists, C - stacks, D - queues** (excluding dequeue's cause they're too OP).

1. Keeping track of which customer in a line came first.

Queue

2. We will expect many inserts and deletes on some dataset, but not too many searches and lookups.

LinkedList → $\Theta(1)$ → $\Theta(N)$

3. We gather a lot of data of a fixed length that will remain relatively unchanged overtime, but we access its contents very frequently.

Array

4. Maintaining a history of the last actions on Word in case I need to undo something.

Stack

5 Pseudo Stack

Implement a stack's `pop` and `push` methods using two Queues. Assume that we have a `MyIntQueue` class with API :

```
boolean isEmpty() //returns true if the queue is empty
void enqueue(int item) //adds item to the back of the queue
int dequeue() //removes the item at the front of the queue
int peek() //returns but doesn't remove the item at the front of the queue
int size() //returns the size of the queue
```

```
public class MyIntStack {
    MyIntQueue q1 = new MyIntQueue();
    MyIntQueue q2 = new MyIntQueue();

    public boolean isEmpty() {
        //Implementation not shown
    }
    public int size() {
        //Implementation not shown
    }
    public void push(int item) {
```

q1.push(item);

move q1 → q2

remove 1st item from q2

and move everything back

```
    }

    public int pop() {
```

while(q1.size > 1){
q2.enqueue(q1.dequeue());
}
int temp = q1.dequeue();
MyIntQueue tempQ = q1;
q1 = q2;
q2 = q1
return temp;

```
    }

}
```

6 A Balancing Act

Given a string *str*, containing just the characters `(`, `)`, {, }, `[`, and `]`, implement a method `hasValidParens` which determines if the string is valid.

The brackets must close in the correct order so "`()`", "`()`{}", and "`[()]`" are all valid, but "`(`", "`({)`}", and "`[(`" are not.

You may use the `getRightParen` method provided below.

```
private static boolean hasValidParens(String str) {
    Stack s = new Stack();
    for (int i = 0; i < str.length(); i++) {
        char c = str.charAt(i);
        if (c == '(' || c == '[' || c == '{') {
            s.push(c);
        } else {
            if (s.isEmpty()) {
                return false;
            }
            if (c != getRightParen(s.pop())) {
                return false;
            }
        }
    }
    return Stack.isEmpty()
}


/**
    The method getRightParen takes in the left parenthesis
    and returns the corresponding right parenthesis.
**/
private static char getRightParen(char leftParen) {
    if (leftParen == '(') {
        return ')';
    } else if (leftParen == '{') {
        return '}';
    } else if (leftParen == '[') {
        return ']';
    } else {
        //not one of the valid parenthesis characters
        throw new IllegalArgumentException();
    }
}
```