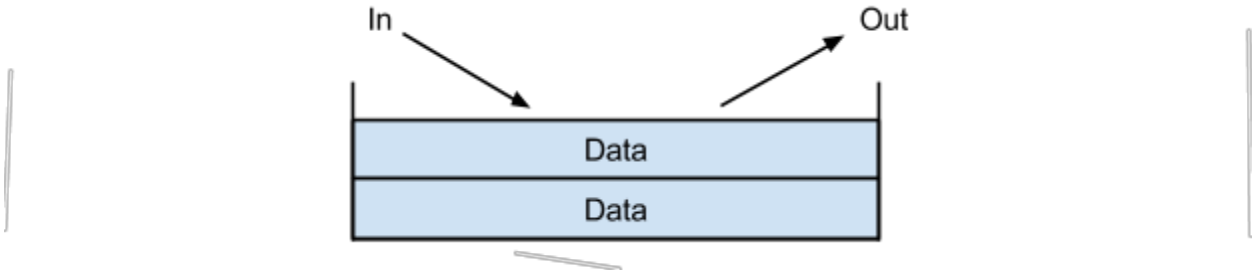
APCS Take Home Quiz #2
Rafi Long

Queue
Superclass: Collection<E>
Meaning: A data structure in which the first element inserted is the first element out. You can append data to one end of the data structure, and then you remove it from the other side.
Draw a picture of a queue.

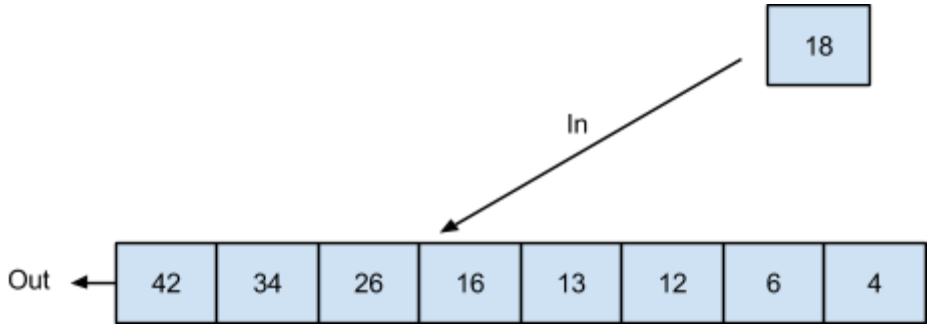

```
Queue todo = new PriorityQueue<String>();
queue.add("10 - Go buy some bread");
queue.add("9001 - Go to robotics");
queue.poll();
```

Stack
Superclass: Vector<E>
Meaning: A data structure in which the last element added is the last element out. You can append data from the data structure, and then you remove it from that side.
Draw a picture of a stack.




```
Stack stack = new Stack();
stack.push(42);
stack.push(7);
stack.pop();
```

Priority Queue
Meaning: A data structure in which the highest value is removed from the data structure each time a value is removed. It could also be the lowest if specified.
Draw a picture of a priority queue.

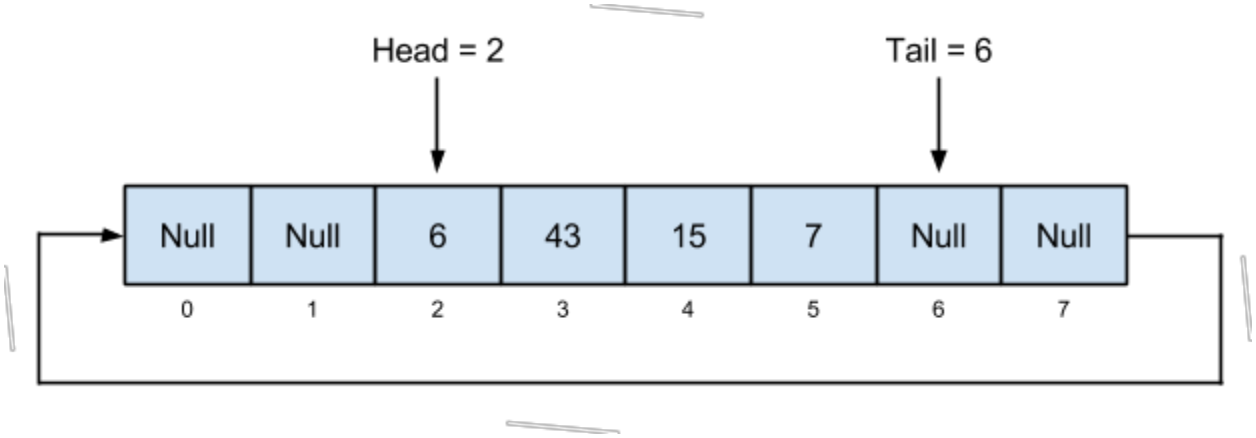



```
PriorityQueue queue = new PriorityQueue<Patient>();
queue.add(joe);
queue.add(fred);
queue.poll();
```

Circular Queue

Meaning: A circular queue is a data structure in which the last index is linked to the first. When elements are added to the queue, then get inserted at the end, or the tail value. When elements are removed, the are removed from the front, or the head value, which is set to null and then incremented. This makes removal and insertion times O(1), as the index they are in is predetermined. Once all data elements are used, the tail value is set to the front of the data structure.

Draw a picture of a circular queue.



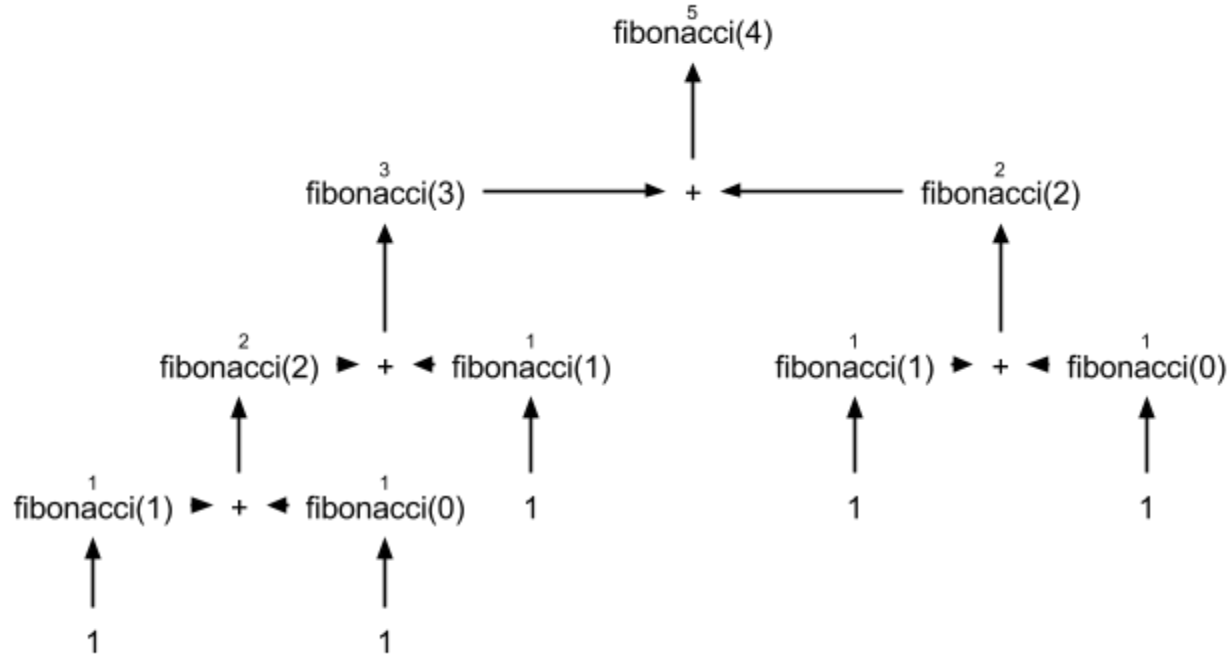

```
// note that java does not support this class, and I am assuming separate implementation
CircularQueue operations = new CircularQueue<Operation>();
operations.add(compile);
operations.add(run);
operations.poll();
```

Recursion

Definition: Recursion is calling a method from inside the method itself. This can be used when the operation builds off of itself, such as fibonacci, so the return can be the return of a return, and so on until the value is given.

Necessary elements: A return that is not void, a conditional, and a call to the same method.

Write the fibonacci method (use long for all data types) using recursion. Draw a diagram of fibonacci(4).




```
public static long fibonacci(long value) {
  if (value > 1) {
    return 1;
  } else {
    return fibonacci(value - 2) + fibonacci(value - 1);
  }
}
```

Tree

Meaning: A tree is a data structure made of nodes, where each node is connected to another in a one way connection called edges. All these edges point away from the root.

```
Node root = new Node();
Node child1 = new Node();
Node child2 = new Node();
Node child3 = new Node();
root.addEdge(new Edge(root, child1));
root.addEdge(new Edge(root, child2));
root.addEdge(new Edge(root, child3));
// Now you have a tree! Not a tree class, though.
```

Binary Tree

Meaning: A binary tree is a tree in which each node has only two children.

```
Node root = new Node();
Node child1 = new Node();
```

```
Node child2 = new Node();
root.edgeOne = new Edge(root, child1);
root.edgeTwo = new Edge(root, child2);
// Now you have a binary tree! The programmer who wrote this didn't yet know of the importance
// of getters and setters
```

Traversal

Meaning: Traversal is the process of visiting every element in a data structure.
Three types: Depth first, breadth first, and other are the three types of traversal in trees.

```
ArrayList array = new ArrayList<Integer>();
Iterator<Integer> = array.iterator();
```

Binary Search Tree

Meaning: A binary search tree is a binary tree in which the children on the left are smaller than the parent, and the children on the right are larger. With a balanced tree this allows you to find the element you are looking for in O(log n) time, as well as with insertion and deletion. In an unbalanced tree the search time is O(n).
Write a program that finds the average depth of a binary search tree in which the numbers 0 to 999 are entered in random order. Predict the average depth.
See attached files.

Complete Binary Tree

Meaning: A complete binary tree is a balanced binary tree, in which all leaves, or nodes with no children, are on the same level. Only in a complete binary search tree with search times be optimal. A complete binary tree can also be referred to as a full binary tree.

```
// This is a complete binary search tree as the elements are added in an order to remain
// balanced. However, a tree can be designed to automatically balanced itself.
BST tree = new BST<Integer>();
tree.add(15);
tree.add(10);
tree.add(20);
```

Full Binary Tree

Meaning: A full binary tree is a balanced binary tree, in which all leaves, or nodes with no children, are on the same level. Only in a full binary search tree with search times be optimal. A full binary tree can also be referred to as a complete binary tree.

```
// This is a full binary search tree as the elements are added in an order to remain
// balanced. However, a tree can be designed to automatically balanced itself.
BST tree = new BST<Integer>();
tree.add(15);
tree.add(10);
tree.add(20);
```

TreeSet

Meaning: A TreeSet is a set based off of a TreeMap. It uses a tree in order to allow log(n) time for add, remove, and contains. There is no index, and all elements are sorted. You can also retrieve the first and last elements.

```
TreeSet set = new TreeSet<Integer>();
set.add(14);
set.add(1);
set.add(52);
set.pollFirst();
// 14 and 52 will remain
set.pollLast();
// 14 will remain
```

TreeMap

Meaning: A TreeMap is sorted by the keys in the order that they are entered or by a comparator. This is dependent on the constructor used on creation of the map.

```
TreeMap map = new TreeMap<String, Integer>();
map.put("I am a key", 15);
map.put("I am also a key", 42);
```