**Quicksort**

**Part 1: Understanding the high level function of the partition method.**

/\*\*

\* partition(): If an element in the array's index is >= start and <= end,

\* then this element is "in bounds". This method first computes the pivot \* which is the element at index (start + end)/2. Then it performs a

\* 'partition' which means that every in bounds element less than the pivot

\* appears before the pivot

\* and every in bounds element greater than the pivot appear after it.

\*/

**public** **static** **int** partition(**int**[] arr, **int** start, **int** end) {

Original: [8, 4, 2, 6, 3, 1, 9, 0]

Going to Perform a Partition with {start, end} = {2, 6}.

Chose Pivot: 3.

After: [8, 4, 2, 1, 3, 6, 9, and 0]

So you look at the execution example, the original array was passed in along with 2 and 6 as the parameters. This resulted is the pivot index being 4 and the element being at that position is the three. So the elements in the original array that are in bounds are {2, 6, 3, 8, and 9}. The only places out of bounds is the first two indexes and the last index. If you look at the array after the partition, particularly the places in bounds {2, 3, 6, 8, 9} you will notice that all elements in bounds larger than 3 appear after three and all elements in bounds smaller than 3 appear before three. Note: the inbounds elements are sorted, they just have to be on the correct side of the pivot.

**Part 2: Understanding the implementation of the partition method.**