Angeleeca Jocson
300234878

Natasa Bolic
300341734

In terms of computational speed for algorithms with a time complexity of O(n), the ordinary array is generally the fastest among the options, ArrayList, Vector, and arrays.

ArrayList and Vector are both part of the Java Collections Framework and are implemented using arrays internally. They provide dynamic resizing, which means they can grow or shrink as needed, but this flexibility comes at a cost. When ArrayList or Vector exceeds their current capacity, they need to allocate a new array and copy elements from the old one, which can introduce some overhead. This resizing operation takes O(n) time complexity in the worst case, where n is the number of elements in the list. Therefore, while operations like adding, removing, or resizing elements can have time complexities of O(1) on average, they can occasionally be O(n) due to the resizing overhead.

The ordinary array in Java has a fixed size, which means it doesn't have the resizing overhead of ArrayList or Vector. Accessing an element by its index in an array takes O(1) time complexity since you can directly compute the memory address of the element. Iterating over all elements in an array, which is a common operation for algorithms with O(n) time complexity, is also very efficient because it involves simple pointer arithmetic. This only works if the built in array is the fixed size needed. If the array needs to be resized, then it would take a lot more memory and operations to resize them, since the ordinary array is NOT dynamic.

Number of elements in array	Time to process ArrayList	Time to process VectorArray	Time to process OrdinaryArray
500 000 000	44618ms	104805ms	43917ms
200 000 000	10509ms	23404ms	7519ms
100 000 000	3585ms	7162ms	3080ms