Method	ArrayList Runtime	LinkedList Runtime	Explanation
get(int index)	O(1)	O(n)	The ArrayList Big O notation is O(1) because it only has an if else statement and there is no looping involved. The LinkedList Big O notation is O(n) meaning in the worst case scenario is could loop through the whole thing or just loop through the first element. Overall, ArrayList is faster.
rotate(int n)	$O(n^2)$	O(n)	The ArrayList Big O notation is $O(n^2)$ because it contains a nested for loop and each for loop in the worst case iterate n times. The LinkedList Big O notation is also O(n) because the for and while loops aren't nested and will run independently. Overall, LinkedList is faster.
merge(List<T>otherList)	O(n + n)	O(n)	The ArrayList Big O notation is O(n + n) because there are 3 while loops which are not nested. Two of them are O(n) and the first while loop is O(n + n) because it has to loop through this.size() and other.size() making it O(n + n) which dominates over the other two. The LinkedList Big O notation is O(n) since the while loops are running independently and run over the same input data. Overall, LinkedList is faster.
reverse() method.	O(n/2)	O(n)	The ArrayList Big O notation is O(n/2) because it contains a for loop that in the worst case it will iterate n/2 times. The LinkedList Big O notation is O(n) meaning it only has to loop through once. Overall, ArrayList is faster.