FUNCTION	BEST-CASE	WORST-CASE
LoadOrders	Because we must go through the entire list to load the orders, we will always have a complexity of **O(n)**	Because we must go through the entire list to load the orders, we will always have a complexity of **O(n)**
LoadDsMembers	If the list is already sorted, we are going to have a complexity of **O(n),** because we are going to make n comparisons.	If the list is not yet sorted, we would have to sort it, and because we have a for loop that has while loop we will have complexity of **O(n^2)**
showDsMembers	Because we must go through the entire list to print the dsmembers, we will always have a complexity of **O(n)**	Because we must go through the entire list to print the dsmembers, we will always have a complexity of **O(n)**
assignDistributors	The best case will be when there are no orders, so we don't have to assign anything. The complexity will be **O(1)**	In the worst case we must go through the list of dsmember and orders and all have equal size. we also have a contain method with a complexity of n. So, we have a complexity of **O(n^3)**
deliverPackages	In this function there is no best case or worst case because we must always go through the dismembers list to find the dismember So the complexity will be **O(n)**(even if it has packages)	In this function there is no best case or worst case because we must always go through the dismembers list to find the dismember So the complexity will be **O(n)**(even if it has packages)
deliver	As this method calls the previous method (deliverpackages) n times, we will have a complexity of **O(n^2)**	As this method calls the previous method (deliverpackages) n times, we will have a complexity of **O(n^2)**

removedsmemeber	The best case will be when the dsmember is not in the list of dsmember, but we must iterate over the list. So, we would have a complexity of **O(n)**	In the worst case when we find the dsmember, we would have to distribute its packages, where we must go through its orders. thereby ending up with a complexity of **O(n^3)**