**Exercise 3: Sorting Customer Orders**

**Scenario:**

You are tasked with sorting customer orders by their total price on an e-commerce platform. This helps in prioritising high-value orders.

***Steps:***

1. Understand Sorting Algorithms:Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).

***Ans:*** *Bubble Sort* is a simple comparison-based algorithm where each pair of adjacent elements is compared, and the elements are swapped if they are in the wrong order. This process is repeated until the list is sorted. O(n^2) in the average and worst case. O(n) in the best case (already sorted array).

*Insertion Sort* is a type of sorting that builds the final sorted array one item at a time. It takes each element from the input and finds the correct place in the sorted part of the array.O(n^2) in the average and worst case. O(n) in the best case (already sorted array).

*Quick Sort*is a divide-and-conquer algorithm that picks a pivot element and partitions the array into two halves, ensuring that elements on the left are less than the pivot and elements on the right are greater. It then recursively sorts the two halves.O(n log n) on average, but O(n^2) in the worst case. The worst case occurs when the smallest or largest element is always chosen as the pivot.

*Merge Sort* is another divide-and-conquer algorithm that divides the array into two halves, recursively sorts them, and then merges the two sorted halves.O(n log n) in all cases (best, average, and worst).

***Setup:***

Create a class Order with attributes like orderId, customerName, and totalPrice.

Code; Order.java

***Implementation:***

Implement Bubble Sort to sort orders by totalPrice.

Implement Quick Sort to sort orders by totalPrice.

Code: Order.java

***Analysis:***

1. Compare the performance (time complexity) of Bubble Sort and Quick Sort.

Bubble Sort:

1. Best Case: O(n)
2. Average Case: O(n^2)
3. Worst Case: O(n^2)

Quick Sort:

1. Best Case: O(n log n)
2. Average Case: O(n log n).
3. Worst Case: O(n^2)
4. Discuss why Quicksort is generally preferred over Bubble Sort.

Quick Sort is significantly more efficient on large datasets as it has an average time complexity of O(n log n), compared to Bubble Sort’s O(n^2).

Quick Sort can handle larger and more complex datasets better than Bubble Sort.

Quick Sort is an in-place sorting algorithm, meaning it requires a small, constant amount of additional storage space, making it more memory-efficient compared to Merge Sort, for example.