## **Exercise 3: Sorting Customers Order**

## **Explanation of Sorting Algorithms:**

**1.Bubble Sort:**

In Bubble Sort, we compare two adjacent elements and swap them if they’re in the wrong order. This process is repeated again and again until the entire list is sorted. It’s simple but slow for large data because it takes O(n²) time in the worst case.

**2.Insertion Sort:**

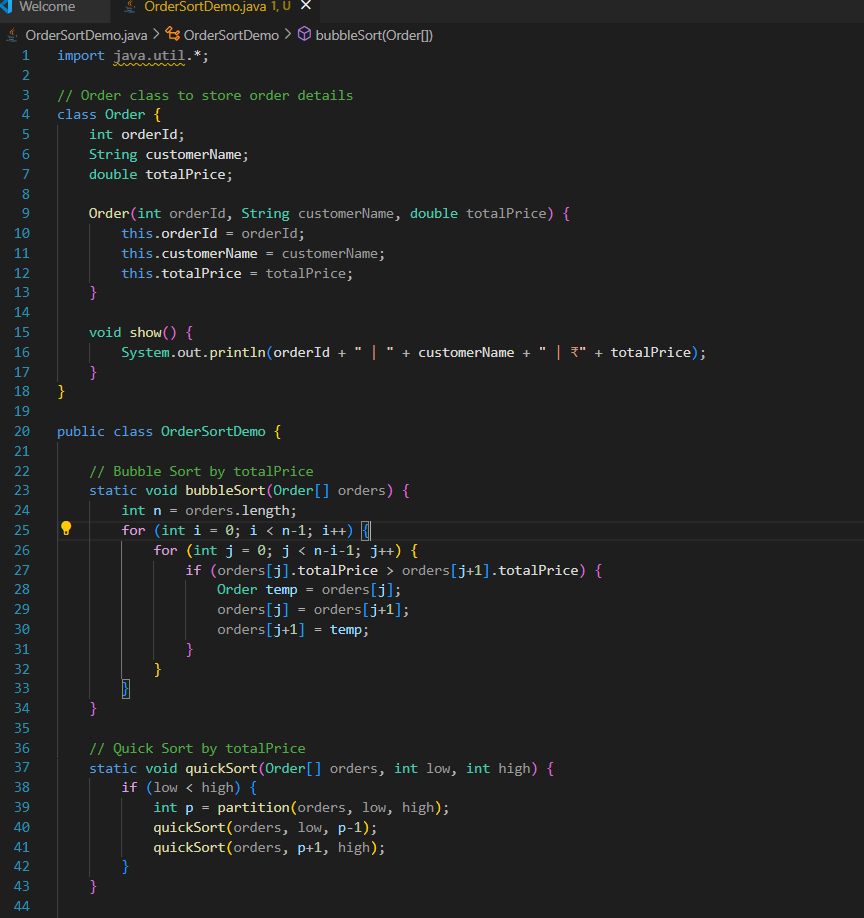
Insertion Sort picks one element at a time and places it in the correct position in the already sorted part of the list. It’s good for small or nearly sorted data but takes O(n²) time when the list is large and unsorted.

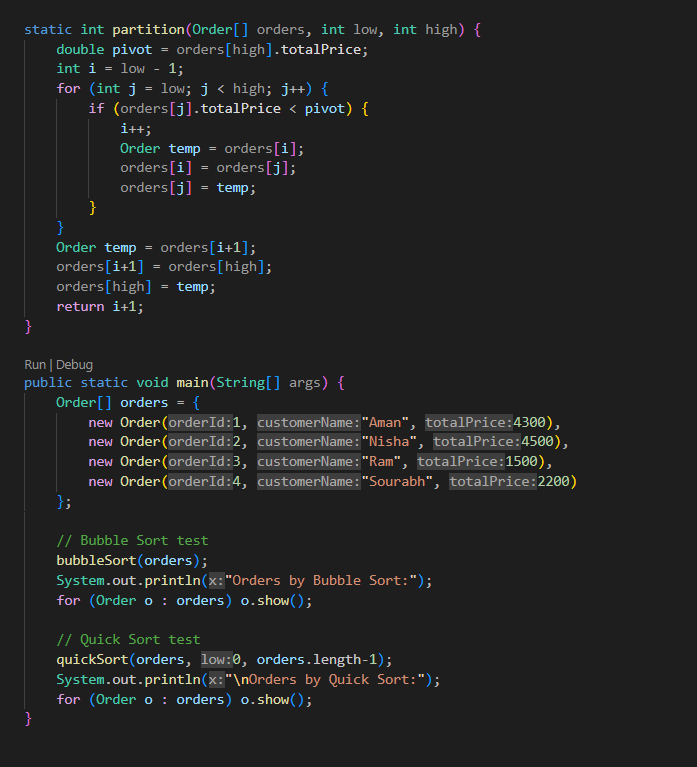
**3.Quick Sort:**

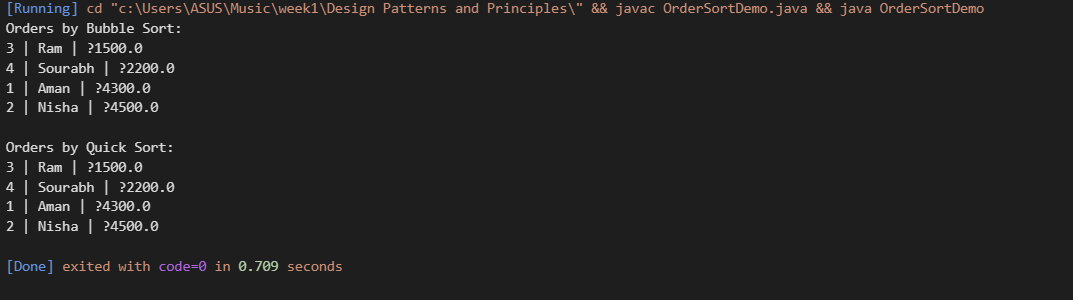
Quick Sort selects a ‘pivot’ element, then arranges the list so that elements smaller than the pivot come before it, and elements greater come after. It then repeats this process on the left and right parts. It’s very fast (O(n log n) average case) but can slow down to O(n²) if pivots are badly chosen.

**4.Merge Sort:**

Merge Sort divides the list into two halves, sorts them separately, and then merges the sorted halves back together. It always takes O(n log n) time in all cases but uses extra memory for merging.







## **Comparison of Bubble Sort and Quick Sort**

## **Bubble Sort**

This is a simple technique where each pair of neighboring items is compared, and if they’re not in the correct order, they are swapped. It keeps repeating this until the entire list is sorted.

**Time Performance:**

* **Best Case:** O(n) → when the list is already sorted.
* **Average Case:** O(n²)
* **Worst Case:** O(n²) → takes a long time if the list is in reverse order.

**Disadsvantages:** Very slow for big lists because it makes too many unnecessary comparisons and swaps.

### **Quick Sort**

Quick Sort uses a smart method called **divide and conquer**. It picks a 'pivot' value, divides the list into two sides — one with values less than the pivot, and one with values greater — then sorts those parts separately.

**Time Performance:**

* **Best Case:** O(n log n)
* **Average Case:** O(n log n)
* **Worst Case:** O(n²) → only if we pick the worst pivot every time (like the smallest or largest element)

**Advantage:** Very efficient for large lists and commonly used in real applications.

## **Conclusion:**

* **Bubble Sort** is simple but slow for big lists.
* **Quick Sort** is faster and better for handling large amounts of data because it reduces the number of steps needed to sort by breaking the list into parts.