**Exercise 3: Sorting Customer Orders**

**Explanation of Sorting Algorithms:**

1. **Bubble sort :** A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. This process repeats until the list is sorted.

* **Time Complexity:** O(n2) in the average and worst cases, where n is the number of elements in the list.

1. **Insertion Sort:** Builds the final sorted list one item at a time. It takes each element and inserts it into its correct position in a sorted part of the list.

* **Time Complexity:** O(n2) in the average and worst cases, but O(n) in the best case when the list is already sorted.

1. **Quick Sort:** A divide-and-conquer algorithm that picks a pivot element and partitions the array into two halves: elements less than the pivot and elements greater than the pivot. It then recursively sorts the subarrays.

* **Time Complexity:** O(n log n) on average, but O(n2) in the worst case. However, with good pivot selection, the worst-case scenario is rare.

1. **Merge Sort:** A divide-and-conquer algorithm that divides the list into equal halves until it can no more be divided. It then merges the smaller lists into new sorted lists until the entire list is merged and sorted.

* **Time Complexity**: O(n log n) in all cases (average, best, and worst).

**Implementation Details :**

**1. Setup :**

* **Order Class:** This class contains attributes like *orderId, customerName*, and *totalPrice* along with appropriate constructors, getters, and setters.

**2. Implementation:**

* **Bubble Sort:** Implemented in the *SortingAlgorithms* class, this method sorts the list of orders by their totalPrice.
* **Quick Sort:** Also implemented in the *SortingAlgorithms* class, this method sorts the list of orders by their totalPrice using the Quick Sort algorithm.

**Analysis:**

**1. Performance Comparison:**

* **Bubble Sort:**
* **Time Complexity:** O(n2)
* **Efficiency**: Less efficient, especially for large datasets.
* **Quick Sort:**
* **Time Complexity:** O(n log n) on average
* **Efficiency**: Generally much more efficient than Bubble Sort, especially for large datasets.

**2. Preference for Quick Sort:** because Quick Sort is preferred over Bubble Sort because it has a significantly better average-case time complexity of O(n log n) compared to Bubble Sort’s O(n2). This makes Quick Sort much more suitable for handling large datasets efficiently.

**Class Diagram of the Program-**

**The output of the program –**

