**Exercise 3: Sorting Customer Orders**

**1. Understand Sorting Algorithms**

Sorting helps in prioritizing data like high-value orders.

* **Bubble Sort**: Compares adjacent elements and swaps them. Time Complexity: O(n²).
* **Insertion Sort**: Builds the sorted list one item at a time. Time Complexity: O(n²).
* **Quick Sort**: Divide-and-conquer method, partitions the list. Time Complexity: O(n log n) average, O(n²) worst-case.
* **Merge Sort**: Stable and fast with O(n log n), but uses extra space.

**2. Setup**

Define a class Order with:

* orderId
* customerName
* totalPrice

**3. Implementation**

👉 Visit the code in the repository to see:

* Bubble Sort and Quick Sort to sort orders by totalPrice.

**4. Analysis**

| **Algorithm** | **Time Complexity** | **Stable** | **Use Case** |
| --- | --- | --- | --- |
| Bubble Sort | O(n²) | Yes | Small datasets |
| Quick Sort | O(n log n) | No | Large datasets |

\***Quick Sort is preferred for large-scale sorting due to efficiency.**