**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 prioritizing high-value orders.

**Steps:**

1. **Understand Sorting Algorithms:**
   * Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).
2. **Setup:**
   * Create a class Order with attributes like orderId, customerName, and totalPrice.
3. **Implementation:**
   * Implement Bubble Sort to sort orders by totalPrice.
   * Implement Quick Sort to sort orders by totalPrice.
4. **Analysis:**
   * Compare the performance (time complexity) of Bubble Sort and Quick Sort.
   * Discuss why Quick Sort is generally preferred over Bubble Sort.

**ANSWER:**

**Explain different sorting algorithms (Bubble Sort, Insertion Sort, Quick Sort, Merge Sort).**

#### **Bubble Sort**

Bubble Sort is a simple comparison-based algorithm. It repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The pass through the list is repeated until the list is sorted.

**Time Complexity**: O(n^2) in the worst and average case, O(n) in the best case (when the list is already sorted).

#### **Insertion Sort**

Insertion Sort builds the sorted list one item at a time. It takes each element from the input data and finds the location it belongs to within the sorted list, then inserts it there.

**Time Complexity**: O(n^2) in the worst and average case, O(n) in the best case.

#### **Quick Sort**

Quick Sort is a divide-and-conquer algorithm. It selects a 'pivot' element and partitions the array into two sub-arrays according to whether they are less than or greater than the pivot. The sub-arrays are then sorted recursively.

**Time Complexity**: O(n log n) on average, O(n^2) in the worst case (rare with good pivot selection).

#### **Merge Sort**

Merge Sort is a divide-and-conquer algorithm. It divides the array into two halves, recursively sorts them, and then merges the two sorted halves.

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

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

| **Cases** | **Bubble Sort** | **Quick Sort** |
| --- | --- | --- |
| **Worst case** | O(n^2) | O(n^2) |
| **Average case** | O(n^2) | O(n log n) |
| **Best case** | O(n) | O(n log n) |

**Discuss why Quick Sort is generally preferred over Bubble Sort.**

* Scalability
* Performance
* Efficiency