**EXERCISE 3 : SORTING CUSTOMER**

**1. Understand Sorting Algorithms:**

**Bubble Sort**:

* **Description**: A simple sorting algorithm that repeatedly steps through the list, compares adjacent elements, and swaps them if they are in the wrong order. The process is repeated until the list is sorted.
* **Time Complexity**: O(n^2) in the worst and average cases, O(n) in the best case (when the array is already sorted).

**Insertion Sort**:

* **Description**: Builds the final sorted array one item at a time. It is much less efficient on large lists than more advanced algorithms such as quicksort, heapsort, or merge sort.
* **Time Complexity**: O(n^2) in the worst and average cases, O(n) in the best case.

**Quick Sort**:

* **Description**: A divide-and-conquer algorithm that works by selecting a 'pivot' element from the array and partitioning the other elements 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, but with good implementations, it is often faster in practice.

**Merge Sort**:

* **Description**: A divide-and-conquer algorithm that divides the unsorted list into n sublists, each containing one element, and then repeatedly merges sublists to produce new sorted sublists until there is only one sublist remaining.
* **Time Complexity**: O(n log n) in all cases.

**2. Setup:**

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

**3. Implementation:**

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

**4. Analysis:**

Compare the performance of Bubble Sort and Quick Sort and discuss why Quick Sort is generally preferred.

**Explanation:**

1. **Order Class**: Defines attributes like orderId, customerName, and totalPrice, with getters and a toString method.
2. **ECommercePlatform Class**:
   * **bubbleSort**: Implements the Bubble Sort algorithm.
   * **quickSort**: Implements the Quick Sort algorithm with a helper method partition.
3. **Main Method**:
   * Creates an array of orders.
   * Provides a menu for the user to choose between Bubble Sort and Quick Sort.
   * Measures and prints the time taken for each sort.
   * Displays the sorted orders.

**Time Complexity Analysis:**

* **Bubble Sort**: O(n^2) in the worst and average cases, O(n) in the best case.
* **Quick Sort**: O(n log n) on average, O(n^2) in the worst case (with poor pivot choices), but usually performs better in practice.

**Quick Sort** is generally preferred over **Bubble Sort** because it is much faster on average, especially for large datasets, due to its O(n log n) time complexity compared to Bubble Sort's O(n^2).