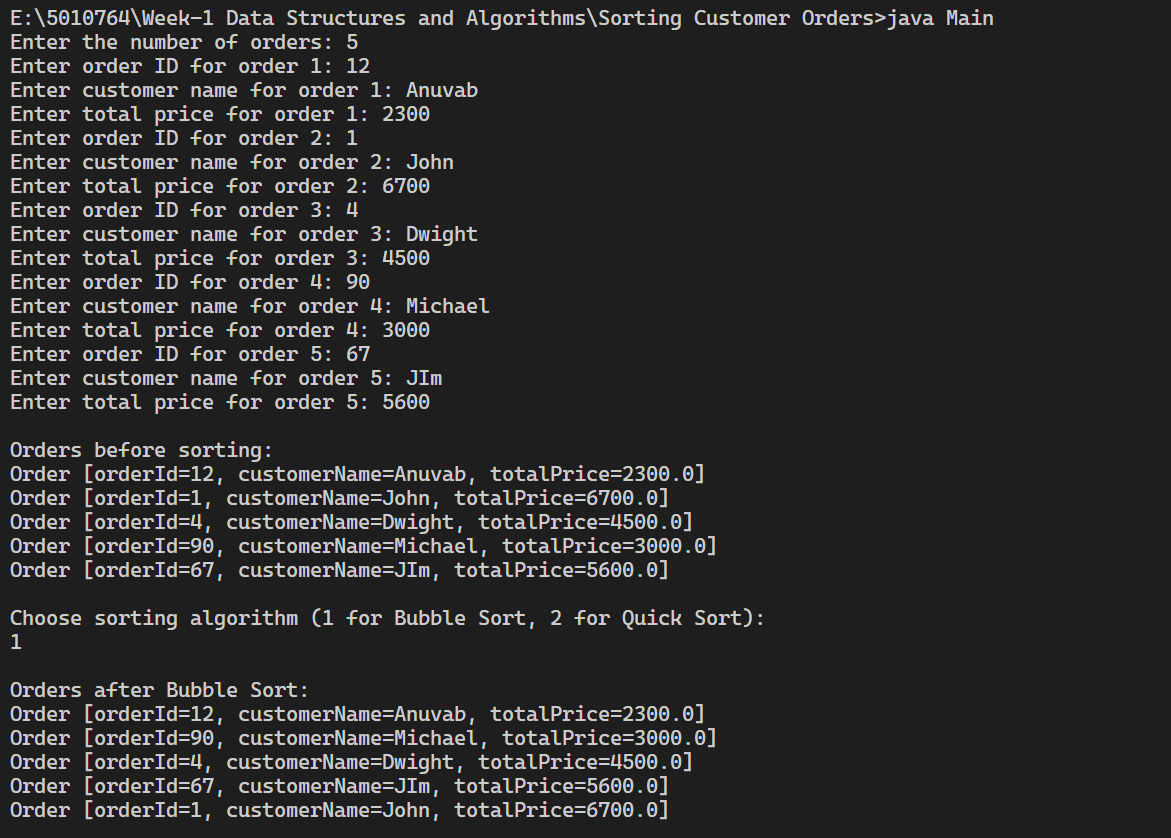
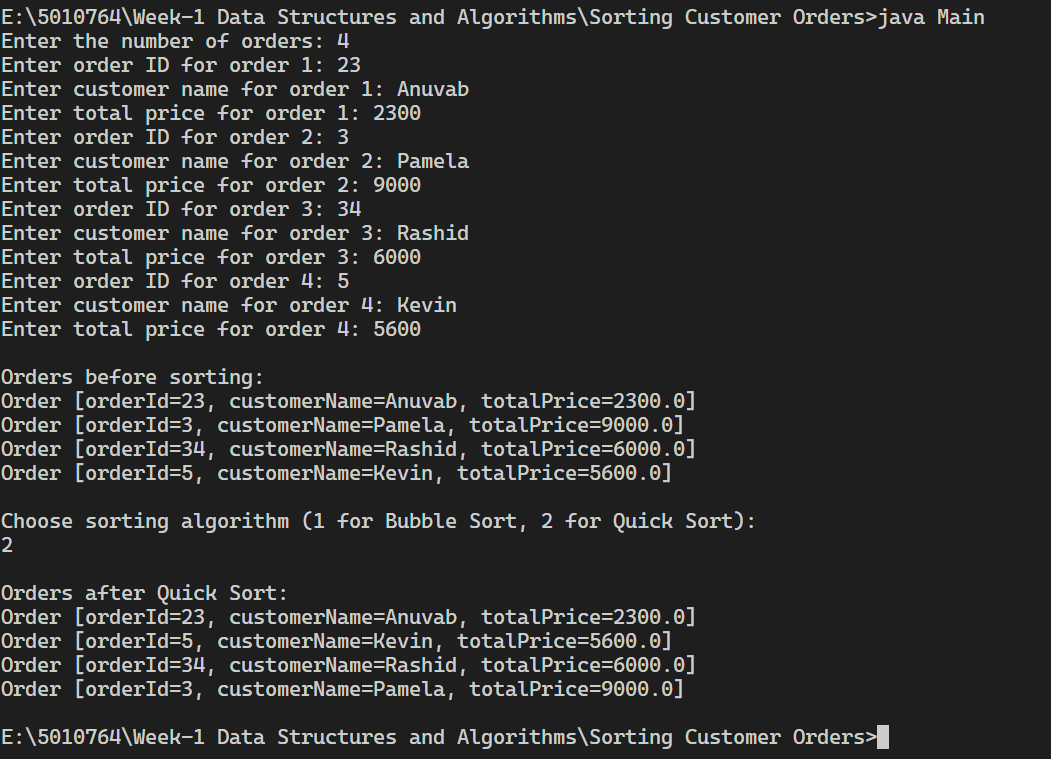
**SORTING CUSTOMER ORDERS:**

1. **Understanding Sorting Algorithms**
   * **Bubble Sort: A simple sorting technique that repeatedly compares and swaps adjacent elements if they are in the wrong order. It continues this process until no more swaps are needed.**
   * **Insertion Sort: This algorithm builds the final sorted array one item at a time by inserting each new item into its correct position among the previously sorted items.**
   * **Quick Sort: An efficient, divide-and-conquer algorithm that selects a 'pivot' element and partitions the array into sub-arrays, which are then sorted recursively.**
   * **Merge Sort: This algorithm divides the array into halves, sorts each half recursively, and then merges the sorted halves to produce the final sorted array.**
2. **Setup**
   * **Define an Order class with attributes such as orderId, customerName, and totalPrice to represent each customer order.**
3. **Implementation**
   * **Bubble Sort Implementation: This method will be used to sort orders based on totalPrice, by repeatedly comparing and swapping adjacent orders as needed.**
   * **Quick Sort Implementation: This method will be used to sort orders based on totalPrice, by selecting a pivot and partitioning the orders into those less than and greater than the pivot, and then sorting each partition.**
4. **Analysis**
   * **Performance Comparison:**
     + **Bubble Sort: Has a time complexity of O(n^2), where n is the number of orders. It is less efficient for large datasets due to its repeated comparisons and swaps.**
     + **Quick Sort: Typically has a time complexity of O(n log n), making it much faster for large datasets. It efficiently partitions the array and sorts the partitions recursively.**
   * **Algorithm Preference:**
     + **Quick Sort is generally preferred over Bubble Sort due to its significantly better average-case time complexity and efficiency in handling larger datasets. Bubble Sort’s O(n^2) complexity makes it impractical for large arrays, while Quick Sort's O(n log n) complexity provides faster performance and scalability.**
5. **Output**

****

****