**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.

**Solution:**

**1. Understand Asymptotic Notation**

**Big O Notation:**

Big O notation describes the performance or complexity of an algorithm in terms of input size (n). It expresses how the runtime or space requirements grow as the input size increases, helping developers:

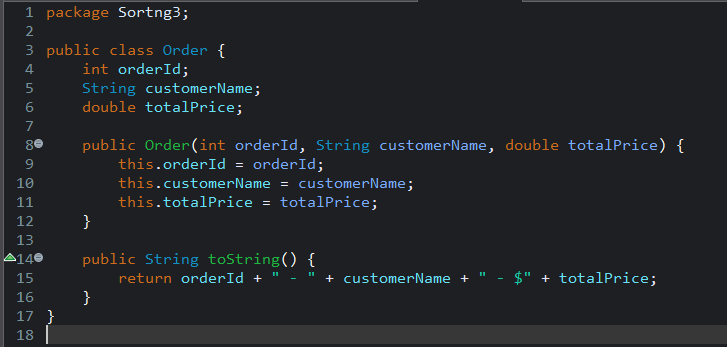
* Analyze scalability.
* Compare different algorithms.
* Choose the most efficient one for large datasets.

| Notation | Description | Example Use Case |
| --- | --- | --- |
| O(1) | Constant time | Accessing an array element |
| O(n) | Linear time | Linear search |
| O(log n) | Logarithmic time | Binary search |
| O(n²) | Quadratic time | Bubble sort |

**Best, Average, and Worst Case Scenarios:**

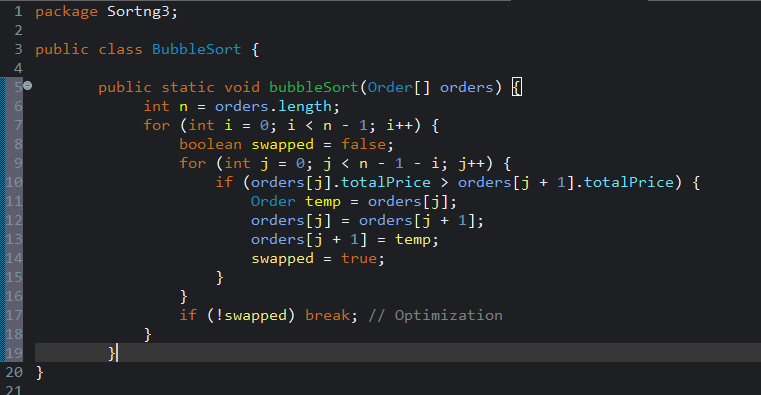
| Search Type | Best Case | Average Case | Worst Case |
| --- | --- | --- | --- |
| Linear Search | Element at first position | Element in middle | Element not found or last |
| Binary Search | Element at mid | Depends on distribution | Element not found |

**2. Setup: Create Order Class**

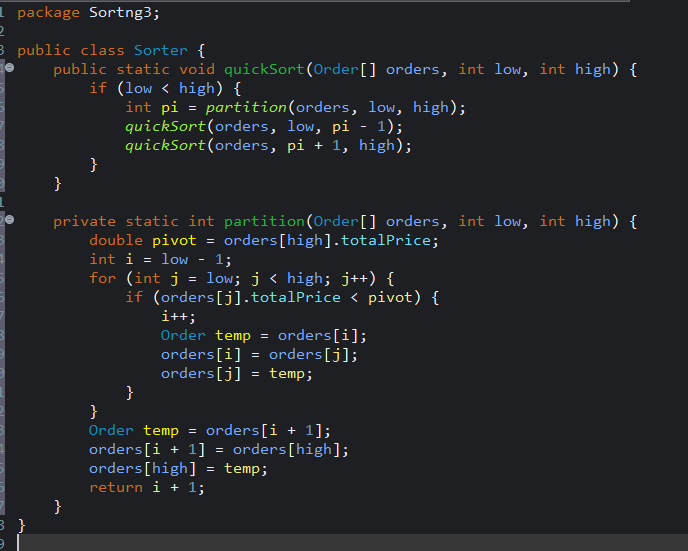
****

**3. Implementation**

**Bubble Sort by totalPrice**

****

**Quick Sort by totalPrice:**

****

**4. Analysis**

**Performance Comparison Table**

| **Algorithm** | **Best Case** | **Average Case** | **Worst Case** | **Space Complexity** | **Stable?** |
| --- | --- | --- | --- | --- | --- |
| Bubble Sort | O(n) | O(n²) | O(n²) | O(1) | Yes |
| Quick Sort | O(n log n) | O(n log n) | O(n²) | O(log n) | No |

**Why Quick Sort Is Preferred Over Bubble Sort**

* Much faster: O(n log n) vs. O(n²)
* Efficient for large datasets
* Less number of swaps and comparisons
* Widely used in system libraries (e.g., Arrays.sort())

Bubble Sort is easy to implement but impractical for production due to its inefficiency with large data.