WEEK – 1

Exercise 3: Sorting Customer Orders

Nipuna Amanapu

superset id : 6432842

**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. Order.java

public class Order {

String orderId;

String customerName;

double totalPrice;

public Order(String orderId, String customerName, double totalPrice) {

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

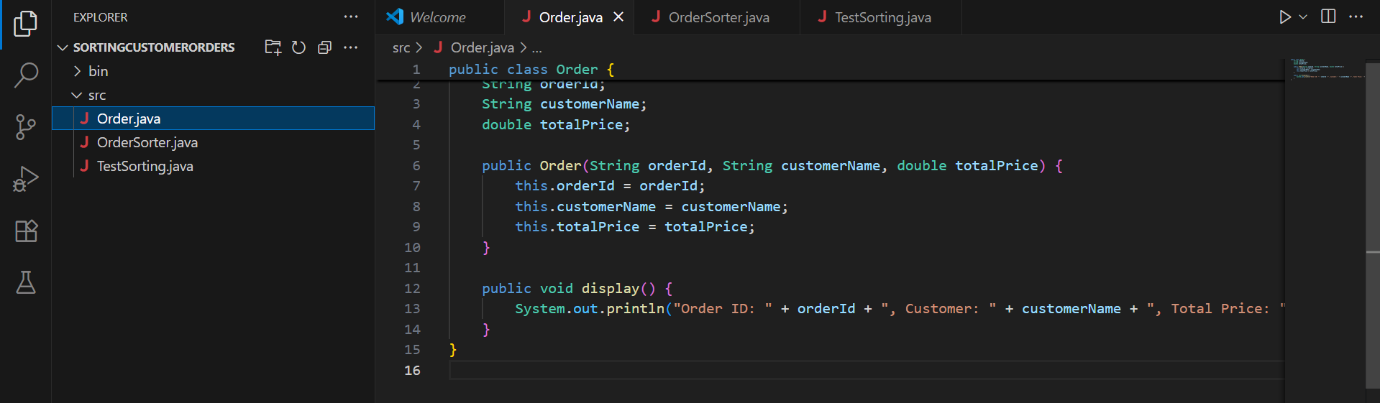
}

public void display() {

System.out.println("Order ID: " + orderId + ", Customer: " + customerName + ", Total Price: " + totalPrice);

}

}



1. OrderSorter.java

public class OrderSorter {

public static void bubbleSort(Order[] orders) {

int n = orders.length;

for (int i = 0; i < n - 1; i++) {

for (int j = 0; j < n - i - 1; j++) {

if (orders[j].totalPrice > orders[j + 1].totalPrice) {

Order temp = orders[j];

orders[j] = orders[j + 1];

orders[j + 1] = temp;

}

}

}

}

public static void quickSort(Order[] orders, int low, int high) {

if (low < high) {

int pi = partition(orders, low, high);

quickSort(orders, low, pi - 1);

quickSort(orders, pi + 1, high);

}

}

private static int partition(Order[] orders, int low, int high) {

double pivot = orders[high].totalPrice;

int i = (low - 1);

for (int j = low; j < high; j++) {

if (orders[j].totalPrice < pivot) {

i++;

Order temp = orders[i];

orders[i] = orders[j];

orders[j] = temp;

}

}

Order temp = orders[i + 1];

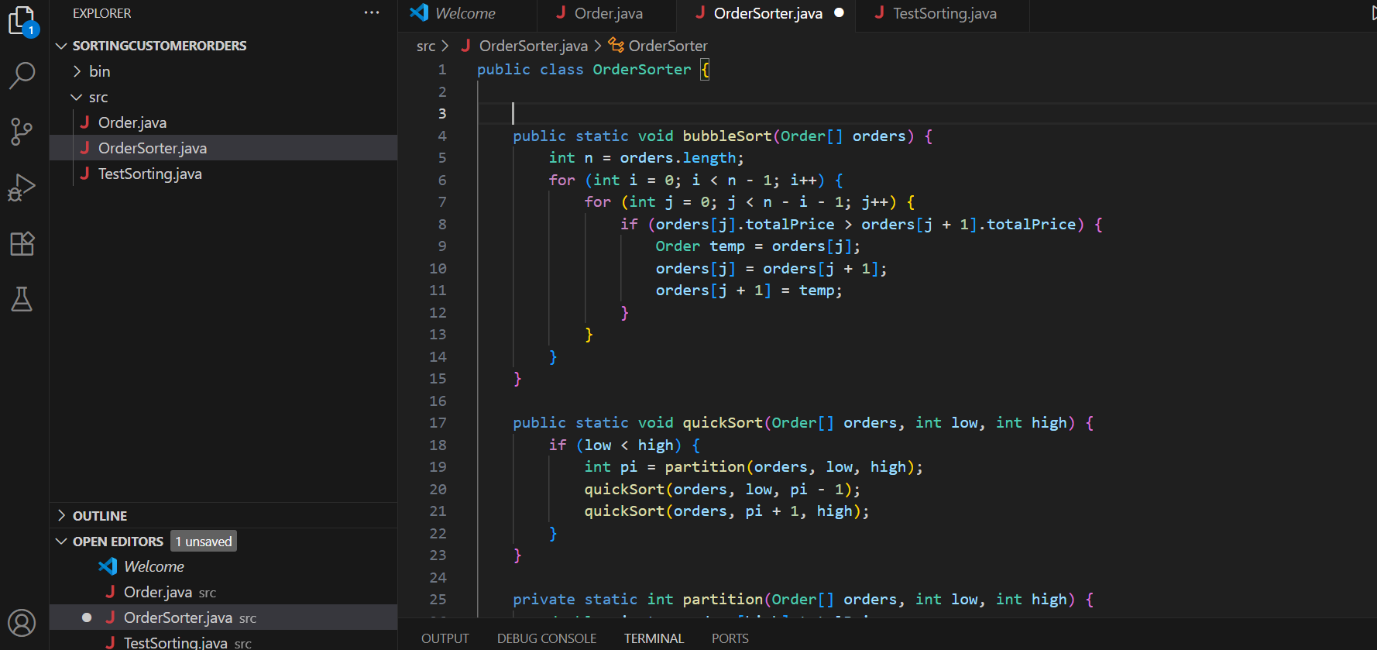
orders[i + 1] = orders[high];

orders[high] = temp;

return i + 1;

}

}



1. TestSorting.java

public class TestSorting {

public static void main(String[] args) {

Order[] orders = {

new Order("O1", "Alice", 250.0),

new Order("O2", "Bob", 100.0),

new Order("O3", "Charlie", 300.0),

new Order("O4", "David", 150.0)

};

System.out.println("Original Orders:");

for (Order order : orders) {

order.display();

}

Order[] bubbleSorted = orders.clone();

OrderSorter.bubbleSort(bubbleSorted);

System.out.println("\nOrders Sorted by Bubble Sort:");

for (Order order : bubbleSorted) {

order.display();

}

Order[] quickSorted = orders.clone();

OrderSorter.quickSort(quickSorted, 0, quickSorted.length - 1);

System.out.println("\nOrders Sorted by Quick Sort:");

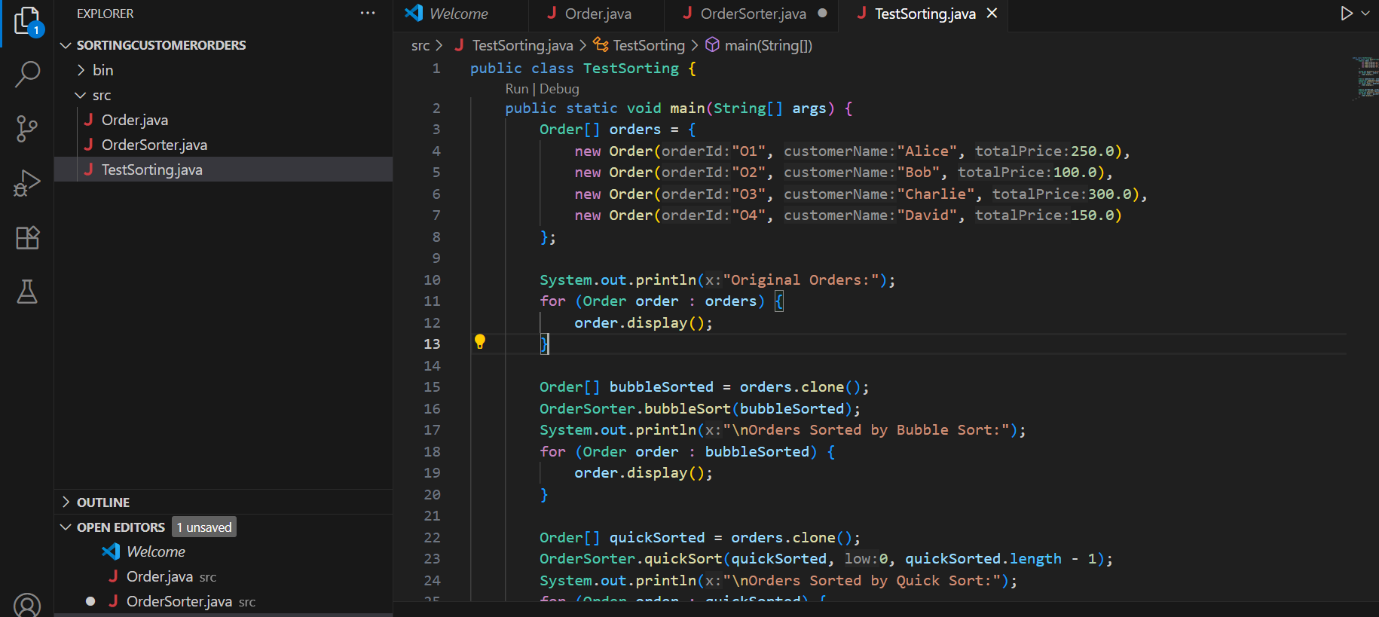
for (Order order : quickSorted) {

order.display();

}

}

}



Output:

