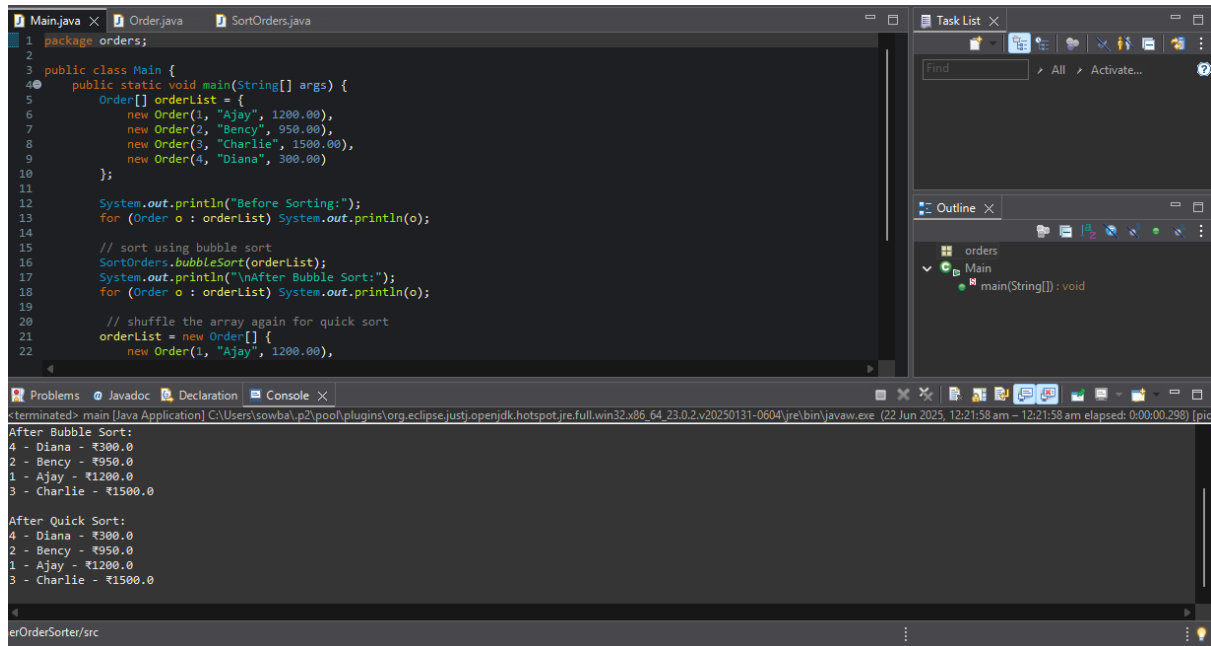


Exercise 3: Sorting Customer Orders

OUTPUT:



```
1 package orders;
2
3 public class Main {
4     public static void main(String[] args) {
5         Order[] orderList = {
6             new Order(1, "Ajay", 1200.00),
7             new Order(2, "Bency", 950.00),
8             new Order(3, "Charlie", 1500.00),
9             new Order(4, "Diana", 300.00)
10        };
11
12        System.out.println("Before Sorting:");
13        for (Order o : orderList) System.out.println(o);
14
15        // sort using bubble sort
16        SortOrders.bubbleSort(orderList);
17        System.out.println("\nAfter Bubble Sort:");
18        for (Order o : orderList) System.out.println(o);
19
20        // shuffle the array again for quick sort
21        orderList = new Order[] {
22            new Order(1, "Ajay", 1200.00),
23            new Order(2, "Bency", 950.00),
24            new Order(3, "Charlie", 1500.00),
25            new Order(4, "Diana", 300.00)
26        };
27
28        SortOrders.quickSort(orderList);
29        System.out.println("\nAfter Quick Sort:");
30        for (Order o : orderList) System.out.println(o);
31    }
32}
```

After Bubble Sort:

```
4 - Diana - ₹300.0
2 - Bency - ₹950.0
1 - Ajay - ₹1200.0
3 - Charlie - ₹1500.0
```

After Quick Sort:

```
4 - Diana - ₹300.0
2 - Bency - ₹950.0
1 - Ajay - ₹1200.0
3 - Charlie - ₹1500.0
```

In sorting customer orders, Quick Sort is preferred over Bubble Sort due to its faster average-case performance ($O(n \log n)$ vs. $O(n^2)$). Choosing efficient algorithms like Quick Sort improves system responsiveness and helps prioritize high-value orders effectively.