**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.

package dsOrder;

public class Order {

int orderId;

String customerName;

double totalPrice;

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

this.orderId = orderId;

this.customerName = customerName;

this.totalPrice = totalPrice;

}

*@Override*

public String toString() {

return "Order ID: " + orderId + ", Customer: " + customerName + ", Total: ₹" + totalPrice;

}

}

package dsOrder;

public class SortOrders {

public static void main(String[] args) {

Order[] orders = {

new Order(101, "Alice", 500.0),

new Order(102, "Bob", 1200.0),

new Order(103, "Charlie", 700.0),

new Order(104, "David", 300.0),

new Order(105, "Eva", 1500.0)

};

System.***out***.println("Before Bubble Sort:");

*printOrders*(orders);

*bubbleSort*(orders);

System.***out***.println("\nAfter Bubble Sort:");

*printOrders*(orders);

orders = new Order[] {

new Order(101, "Alice", 500.0),

new Order(102, "Bob", 1200.0),

new Order(103, "Charlie", 700.0),

new Order(104, "David", 300.0),

new Order(105, "Eva", 1500.0)

};

System.***out***.println("\nBefore Quick Sort:");

*printOrders*(orders);

*quickSort*(orders, 0, orders.length - 1);

System.***out***.println("\nAfter Quick Sort:");

*printOrders*(orders);

}

static void printOrders(Order[] orders) {

for (Order o : orders) {

System.***out***.println(o);

}

}

public static void bubbleSort(Order[] orders) {

int n = orders.length;

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

boolean swapped = false;

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;

swapped = true;

}

}

if (!swapped) break;

}

}

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;

}

}

Output:

