**Exercise 3: Sorting Customer Orders**

Bubble Sort:

\* - Repeatedly compares adjacent elements and swaps them if in wrong order.

\* - Time Complexity: O(n^2)

Insertion Sort:

\* - Builds the sorted list one element at a time by inserting into the correct position.

\* - Time Complexity: O(n^2)

Quick Sort:

\* - Divide and conquer algorithm that picks a pivot and partitions array.

\* - Average Time Complexity: O(n log n)

\* - Worst case: O(n^2), but better in practice with good pivot choice.

Merge Sort:

\* - Divides array, sorts each half, and merges.

\* - Time Complexity: O(n log n)

public class Main {

static 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 "[" + orderId + "] " + customerName + " - Rs." + totalPrice;

}

}

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 - 1 - i; 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; // optimization

}

}

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;

}

public static void printOrders(Order[] orders) {

for (Order o : orders) {

System.out.println(o);

}

}

public static void main(String[] args) {

Order[] orders = {

new Order(1001, "kavya", 250.50),

new Order(1002, "suba", 125.75),

new Order(1003, "sruthik", 500.00),

new Order(1004, "deepu", 300.40),

new Order(1005, "udhaya", 180.90)

};

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

printOrders(orders);

// Bubble Sort

Order[] bubbleSorted = orders.clone();

bubbleSort(bubbleSorted);

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

printOrders(bubbleSorted);

// Quick Sort

Order[] quickSorted = orders.clone();

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

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

printOrders(quickSorted);

}

}