**Order Sorting System Documentation**

**1. Overview**

The Order Sorting System is a Java application that allows users to input order details, store them, and sort them using different algorithms. The system demonstrates the implementation and comparison of two sorting algorithms: Bubble Sort and Quick Sort.

**2. Classes**

**2.1 Order**

The Order class represents an individual order in the system.

**Attributes:**

* orderId (int): Unique identifier for the order
* customerName (String): Name of the customer who placed the order
* totalPrice (double): Total price of the order

**Methods:**

* Order(int orderId, String customerName, double totalPrice): Constructor
* getOrderId(): Returns the order ID
* getCustomerName(): Returns the customer name
* getTotalPrice(): Returns the total price
* toString(): Returns a string representation of the order

**2.2 SortingAlgorithms**

The SortingAlgorithms class contains static methods for sorting lists of Order objects.

**Methods:**

* bubbleSort(List<Order> orders): Sorts the list using the Bubble Sort algorithm
* quickSort(List<Order> orders, int low, int high): Sorts the list using the Quick Sort algorithm
* partition(List<Order> orders, int low, int high): Helper method for Quick Sort

**2.3 OrderSortingMain**

The OrderSortingMain class contains the main method and serves as the entry point for the application.

**Methods:**

* main(String[] args): Runs the main program loop, handling user input and demonstrating sorting
* printOrders(List<Order> orders): Helper method to print a list of orders

**3. Functionality**

1. **Input Orders**: Users can input multiple orders by providing the order ID, customer name, and total price for each order.
2. **Display Original Orders**: The system displays the list of orders in their original input order.
3. **Sort Using Bubble Sort**: The system sorts the orders using the Bubble Sort algorithm and displays the sorted list.
4. **Sort Using Quick Sort**: The system sorts the orders using the Quick Sort algorithm and displays the sorted list.

**4. Sorting Algorithms**

**4.1 Bubble Sort**

* Implementation: The bubbleSort method in SortingAlgorithms class
* Time Complexity: O(n²) in the average and worst cases
* Space Complexity: O(1)
* Characteristics: Simple implementation, inefficient for large datasets

**4.2 Quick Sort**

* Implementation: The quickSort and partition methods in SortingAlgorithms class
* Time Complexity: O(n log n) on average, O(n²) in the worst case
* Space Complexity: O(log n) due to recursion
* Characteristics: Efficient for large datasets, in-place sorting

**5. Data Structure**

The system uses ArrayList<Order> to store and manage the list of orders. This allows for dynamic sizing and easy access to elements.

**6. User Interface**

The system provides a simple command-line interface. Users interact with the system by entering order details when prompted. The sorting results are displayed in the console.

**7. Program Flow**

1. The program prompts the user to enter order details.
2. Users input order information until they type 'done'.
3. The original list of orders is displayed.
4. The list is sorted using Bubble Sort and displayed.
5. The list is sorted using Quick Sort and displayed.
6. The program terminates
   1. Implementation

Link: [For Code](https://github.com/Akashmondal55/Akash_5016855/tree/main/Week-1/DSA/Exercise-3)

**9. Conclusion**

The Order Sorting System demonstrates the implementation and comparison of two fundamental sorting algorithms: Bubble Sort and Quick Sort. This system serves as an educational tool to understand how these algorithms work with custom objects (Orders) and compare their results.

Key points of the system:

* Practical application of sorting algorithms on custom objects
* Comparison of simple (Bubble Sort) vs. efficient (Quick Sort) algorithms
* Demonstration of working with lists and user input in Java

Potential enhancements for future versions:

* Adding more sorting algorithms for comparison (e.g., Merge Sort, Heap Sort)
* Implementing sorting by different criteria (e.g., order ID, customer name)
* Adding timing mechanisms to compare algorithm performance
* Creating a graphical user interface for easier interaction
* Implementing file I/O to save and load order data

This Order Sorting System provides a solid foundation for understanding and comparing sorting algorithms in a practical context, making it a valuable tool for learning and experimentation in computer science and software development.