Machine Coding Round: FLIPKART

Description
We are required to build an app that lets customers search for and book flights. An airline can login to the portal and declare their flight schedule for the given day. Customers can search for flights between two cities. For simplicity you can assume that the airlines declare the schedule for a particular day and customers also book for the same day.

1. The airlines are naturally competitive and there can be multiple flights from City A to City B, often with different prices.
2. Due to logistical constraints, not all airlines may have an available flight between a pair of cities.

Data

1. Every city is uniquely identified by a combination of three alphabets. Example: BLR, DEL, NYC.
2. Airline names are also unique in the system, have no spaces, and can be of any length greater than 2. Example: JetAir, Indigo, Delta
3. Cost of Flight per passenger is always a positive value

Initial Input
A set of lines in this format:

AirlineName	**SourceCity**	**DestinationCity**	**Price**
JetAir	DEL	BLR	500

Problems:
Customers will input source and destination city; implement the feature which prints the following:

1. **Minimum Number of Hops:** Example output for DEL to NYC can look like
   DEL to LON via Delta for 2000
   LON to NYC via Delta for 2000

   Total Flights = 2
   Total Cost = 4000

   If there are multiple options with the same source and destination and each option has the same number of hops then you should return the one which has the minimum cost.

2. **Cheapest Flight**: Example output for DEL to NYC can look like
   DEL to BLR via JetAir for 500
   BLR to LON via JetAir for 1000
   LON to NYC via Delta for 2000

Total Flights = 3
Total Cost = 3500
Here the total cost (3500) is the lowest possible cost for the trip amongst all airlines and route options.

If there are multiple options with the same source and destination and each option has the same cost then you should return the one which has the minimum number of hops.

3. **Filter for Search:** Assume that all IndiGo flights serve meals.
   Customers when searching for flights between cities will also provide the input on whether they want to limit their search to flights with a meal service. This should be an extensible feature to other properties like providing drinks, excess baggage, economy class, business class etc.

Bonus Question

1. Extending your app to provide sorting functionality i.e. if you were to start storing the departure and arrival times, users get the capability to sort the results by their departure/arrival times in ascending/descending order
2. Handling concurrency wherever applicable

Requirement:
Should support this using in-memory DS constructs, use of DB not allowed.

Expectations:
Create the sample data yourself. You can put it into a file, test case or main driver program itself. Unit test cases are not expected.

1. Code should be demo-able.
2. Write a driver class for demo purpose which will execute all the commands at one place in the code and test cases.
3. Code should have basic modular design. Please do not jam the responsibilities of one class into another.
4. Code should be extensible. Wherever applicable, use interfaces and contracts between different methods. It should be easy to add/remove functionality without re-writing the entire codebase.
5. Code should handle edge cases properly and fail gracefully.
6. Code should be legible and readable.
7. CLI, Web based application, REST API and UI are not expected.

Guidelines:

1. You have 90 minutes to solve this problem.
2. Please discuss the solution with the interviewer
3. Please do not access internet for anything EXCEPT syntax

4. You are free to choose any language of your choice to code.
5. All code should be your own, do not use any libraries and **usage of AI tools is not allowed**.
6. Please focus on the Bonus questions only after ensuring the required features are complete and demoable.

Sample Test cases:
The input/output need not be exactly in this format but the functionality should remain intact

```
i: input
o: output

i:register flight-> JetAir -> DEL -> BLR -> 500
o: JetAir DEL -> BLR flight registered

i:register flight-> JetAir -> BLR -> LON -> 1000
o: JetAir BLR -> LON flight registered

i:register flight-> Delta -> DEL -> LON -> 2000
o: Delta DEL -> LON flight registered

i:register flight-> Delta -> LON -> NYC -> 2000
o: Delta LON -> NYC flight registered

i:register flight-> IndiGo -> LON -> NYC -> 2500
o: Indigo LON -> NYC flight registered

i:register flight-> IndiGo -> DEL -> BLR -> 600
o: IndiGo DEL -> BLR flight registered

i:register flight-> IndiGo -> BLR -> PAR -> 800
o: IndiGo BLR -> PAR flight registered

i:register flight-> IndiGo -> PAR -> LON -> 300 -> True
o: IndiGo PAR -> LON flight registered


i:search flight-> DEL -> NYC
o:
* Route with Minimum Hops:
DEL to LON via Delta for 2000
LON to NYC via Delta for 2000

Total Flights = 2
Total Cost = 4000
```

```
* Cheapest Route:
DEL to BLR via JetAir for 500
BLR to LON via JetAir for 1000
LON to NYC via Delta for 2000

Total Flights = 3
Total Cost = 3500


i:search flight-> DEL -> NYC -> TRUE
o:
* Route with Minimum Hops:
DEL to BLR via Indigo for 600
BLR to PAR via Indigo for 800
PAR to LON via Indigo for 300
LON to NYC via Indigo for 2500

Total Flights = 4
Total Cost = 4200

* Cheapest Route:
DEL to BLR via Indigo for 600
BLR to PAR via Indigo for 800
PAR to LON via Indigo for 300
LON to NYC via Indigo for 2500

Total Flights = 4
Total Cost = 4200
```