FoodKart

Description:

Implement a food order management system.
The system has the following features -

1. This system has a tie-up with restaurants, where each restaurant has a menu. The menu consists of the items and their corresponding prices.
2. Each restaurant has a maximum processing capacity in terms of number of items at any given time. It won't accept any further orders until the orders which are being processed are dispatched.
3. Once an order is dispatched it notifies the system about it. Depending on the processing capacity it can take more orders.
4. Only one restaurant can be selected for an order based on the restaurant selection strategy.
5. An order is accepted only if all the items can be fulfilled by the selected restaurant.

What should the system do?

1. Onboard new restaurant with its items menu and processing capacity. This can be loaded once and kept static.
2. Any restaurant should be able to change its menu or update price.
3. Any restaurant should be able to mark an order as dispatched.
4. The customer should be able to place an order by giving items.
5. Implement one restaurant selection strategy. eg: restaurant can be selected based on the lowest price of the item. This should be extensible for adding new strategies.
6. The system should be able to keep track of all items served by each restaurant.
7. Given an unordered list of commands, Execute the commands ordered by timestamp & occurrence.

   Example commands:

   ```
   5, place-order, order2, item1, item2, item3
   2, update-price, restaurant3,  item1, 50
   8, dispatch-order, order2
   5, place-order, order3, item2, item3
   3, place-order, order1, item1
   2, add-items, restaurant3, item4, 60
   6, remove-items, restaurant3, item2
   ```

Output

`>> 2, **update-price,** restaurant3,item1, 50`
Print Menu of the Restaurant

`>> 2, **add-items**, restaurant3, item4, 60`
Print Menu of the Restaurant

`>> 3, **place-order,** order1, item1`
Print Order Total and Restaurant name if it's placed successfully

`>> 5, **place-order**, order2, item1, item2, item3`
Print Order Total and Restaurant name if it's placed successfully

`>> 5, **place-order**, order3, item2, item3`
Print Order Total and Restaurant name if it's placed successfully

`>> 6, **remove-items**, restaurant3, item2`
Print Menu of the Restaurant

`>> 8, **dispatch-order,** order2`
Print the order

Expectations:

- Code should be demo-able (**very important**). Code should be functionally correct and complete.
    - At the end of this interview round, an interviewer will provide multiple inputs to your program for which it is expected to work
- Code should handle edge cases properly and fail gracefully. Add suitable exception handling, wherever applicable.
- Code should have a good object-oriented design.
- Code should be readable, modular, testable and extensible. Use intuitive names for your variables, methods and classes.
    - It should be easy to add/remove functionality without rewriting a lot of code.
    - Do not write monolithic code.

Guidelines**:**

- Input can be read from a file or STDIN or coded in a driver method.
- Output can be written to a file or STDOUT.

- Feel free to store all interim/output data in-memory. The use of databases & frameworks are **not** encouraged.
- Restrict internet usage to looking up syntax
- You are free to use the language of your choice.