CSE 108: Java Term Project (Part 1)

Restaurant Database System

You are required to write a program that implements a simple restaurant database. This database system aims to keep a database of the restaurants with their food menu and perform operations such as searching for restaurants, searching for food items on the menu, adding restaurants, adding food items to the menu, etc.

The Restaurant Database System should provide the following features:

- Maintains a list (using a Java Collection class) of Restaurant objects
- Each Restaurant object represents a restaurant
- Each restaurant should contain the following information:
    - Id
    - Name
    - Score
    - Price ($, $$, or $$$)
    - Zip Code
    - Categories (A restaurant can have up to three different categories)
- Maintains a list (using a Java Collection class) of Food objects
- Each Food object represents a food item on the menu of a specific restaurant
- Each Food should contain the following information:
    - Restaurant Id
    - Category
    - Name
    - Price
- Restaurants can be added to the list
- Food items can be added to the menu list
- Lists the details of an existing restaurant (or restaurants)
- Produces a report of restaurants and food items based on some criteria
- Produces different statistics of the restaurants and food items
- Loads a list of restaurants from a text file and saves the list of current restaurants to a text file
- Loads a list of food items on the menu of restaurants from a text file and saves the list of current food items on the menu to a text file

When the restaurant database system starts, it should automatically load two text files called ‘restaurant.txt’ and ‘menu.txt’. The ‘restaurant.txt’ file contains details of all restaurants currently kept by the database. The “menu.txt" file contains details of all food items on the menu currently kept by the database. The actual format of this text file is described later in this document. The data loaded should be stored in some appropriate data structures. No other reading from or writing to the file is required while the program is in operation until the user chooses to exit, at this point, the program saves all the data in memory back to the same text files (‘restaurant.txt’ and ‘menu.txt’). In other words, the file I/O operations are performed automatically by the program and require no direct interactions with the user.

Main Menu
When the program is running, it should repeatedly display the command line based main menu with these options:

Main Menu:
1) Search Restaurants
2) Search Food Items
3) Add Restaurant
4) Add Food Item to the Menu
5) Exit System

Option (1) of the Main Menu
Option (1) of the main menu allows the user to search for restaurants in the database. The user should be asked what criteria(s) he/she wants to use to search for restaurants. The following sub-menu and options should be displayed:

Restaurant Searching Options:
1) By Name
2) By Score
3) By Category
4) By Price
5) By Zip Code
6) Different Category Wise List of Restaurants
7) Back to Main Menu

Inputs other than 1-7 should be rejected, and an error message should be printed. The menu should be displayed repeatedly until the user chooses Option (7).

If the user chooses option (1), you should ask the user to input a restaurant name and then search the database for any restaurant with the specific name. If found, display all information about this restaurant or display "No such restaurant with this name" if not found. The search needs to be case-insensitive.

If the user chooses option (2), you should ask the user to input two numbers as range and then search the database for any restaurant with a score in this range. If found, display all the restaurant or display "No such restaurant with this score range" if not found.

If the user chooses option (3), you should ask the user to input a category and then search the database for any restaurant with the specific category. If found, display all the restaurants or display "No such restaurant with this category" if not found. The search needs to be case-insensitive.

If the user chooses option (4), you should ask the user to input a price and then search the database for any restaurant with the specific price. If found, display all the restaurants or display "No such restaurant with this price" if not found.

If the user chooses option (5), you should ask the user to input a zip code and then search the database for any restaurant with the specific zip code. If found, display all the restaurants or display “No such restaurant with this zip code” if not found.

If the user chooses option (6), display the category-wise restaurant names. You do not need to take any input from the users. For example, the following output will be produced with the provided data:

Chicken: KFC
Fast Food: KFC, McDonalds
Family Meals: KFC, IHOP
Breakfast and Brunch: IHOP, Starbucks
Burgers: IHOP,McDonalds
Coffee and Tea: Starbucks
Bakery: Starbucks

If the user chooses option (7), the program should go back to the main menu.

Option (2) of the Main Menu
Option (2) of the main menu allows the user to search for food items on the menu of the restaurants. The user should be asked what criteria(s) he/she wants to use to search. The following sub-menu and options should be displayed:

Food Item Searching Options:
1) By Name
2) By Name in a Given Restaurant
3) By Category
4) By Category in a Given Restaurant
5) By Price Range
6) By Price Range in a Given Restaurant
7) Costliest Food Item(s) on the Menu in a Given Restaurant
8) List of Restaurants and Total Food Item on the Menu
9) Back to Main Menu

Inputs other than 1-9 should be rejected, and an error message should be printed. The menu should be displayed repeatedly until the user chooses Option (9)

If the user chooses option (1), you should ask the user to input a food item name and then search the database for any food item with the specific name. If found, display all the foot items or display “No such food item with this name” if not found. The search needs to be case-insensitive.

If the user chooses option (2), you should ask the user to input a food item name and a restaurant name and then search the database for any food item on the menu of the input restaurant with the specific name. If found, display all the foot items or display “No such food item with this name on the menu of this restaurant” if not found. The search needs to be case-insensitive.

If the user chooses option (3), you should ask the user to input a food item category and then search the database for any food item with the specific category. If found, display all the foot items or display “No such food item with this category” if not found. The search needs to be case-insensitive.

If the user chooses option (4), you should ask the user to input a food item category and a restaurant name and then search the database for any food item on the menu of the input restaurant with the specific category. If found, display all the foot items or display “No such food item with this category on the menu of this restaurant” if not found. The search needs to be case-insensitive.

If the user chooses option (5), you should ask the user to input two numbers as range and then search the database for any food item with a price in this range. If found, display all the foot items or display “No such food item with this price range” if not found.

If the user chooses option (6), you should ask the user to input two numbers as range and a restaurant name and then search the database for any food item on the menu of the input restaurant with a price in this range. If found, display all the foot items or display “No such food item with this price range on the menu of this restaurant” if not found.

If the user chooses option (7), you should ask the user to input a restaurant name and then display the costliest food items on the menu of the input restaurant.

If the user chooses option (8), display the total food items on the menu for every restaurant. You do not need to take any input from the users. For example, the following output will be produced with the provided data:

KFC: 60
IHOP: 93
Starbucks: 144
McDonalds: 92

If the user chooses option (9), the program should go back to the main menu.

Option (3) of the Main Menu
Option (3) of the main menu allows the user to add a new restaurant to the database. You must take input for all the information of a restaurant. Please note that a restaurant must have at least one and at most three categories.

Option (4) of the Main Menu
Option (4) of the main menu allows the user to add a food item to the menu of a specific restaurant. You first take the restaurant name as input and then take input for all the information of a food item. Please note that the input restaurant must exist in the database to add the food item to its menu.

Option (5) of the Main Menu
Option (5) of the main menu exits the program. All the restaurants currently in memory are automatically saved back to ‘restaurant.txt’ if any new restaurant is added. All the food items in memory are automatically saved back to ‘menu.txt" if any new food item is added.

Inputs other than 1-5 to this main menu should be rejected, and an error message should be printed. The menu should be displayed repeatedly until the user chooses Option (5).

Input File Format
The input data file (‘restaurant.txt’) has the following format for each line: (Note that there is no space between a word and a comma).

Id,Name,Score,Price,ZipCode,Category1,Category2,Category3

Name, Price, ZipCode, Category1, Category2, and Category3 are Strings. Category2 and Category3 can be empty. Id is a positive integer, and Score is double.

The input data file (menu.txt) has the following format for each line: (Note that there is no space between a word and a comma).
RestaurantId,Category,Name,Price

Category and Name are Strings, and Price is double. RestaurantId is a positive integer, and the id must belong to an Id of a restaurant in the database.

Important Assumptions
You should assume the following when implementing your program:
- You are not allowed to hardcode anything
- You have to reuse your code later, so design and code accordingly
- All restaurant names are unique - if a restaurant with the inputted name is already in the database, you cannot add this restaurant, and should generate an error
- There cannot be two food items with the same name and same category in a specific restaurant. You need to consider this while adding food items
- There is no limit to how many restaurants can be stored
- There is no limit to how many food items can be stored
- When performing searches, all search strings are case-insensitive
- The data file is always in the correct format - i.e., no need to validate the data when reading it in
- All on-screen input/output should be formatted in a user-friendly manner. Sensible error messages should be displayed whenever appropriate (e.g., when searching for a restaurant which is not in the database, a “Not Found” error message should be displayed)
- You are not allowed to use JavaFX or Java Swing for this part; everything should be command line based
- What does $$ price mean in a restaurant? $, $$, and $$$ refer to the cost per person for an average meal ($ for $1 to $10, $$ for $11 to $20, and $$$ for $21 to $30). We will only process the strings $, $$, and $$$ for this assignment