Middle East Technical University | Department of Computer Engineering

CENG 351
Data Management and File Structures

Fall 2022-2023

Programming Assignment 1

Due date: 21 December 2022, Wednesday, 23:55
Submission: **via ODTUClass**

1 Introduction

In this assignment, you are asked to create a system by designing queries and implementing pre-defined functions to operate on a database for a food recommendation service FoodRec. You will provide specific tasks and a well-defined interface. Objective of this assignment is to help you get familiar with connecting and querying to MySQL Server using Java Database Connectivity (JDBC).

All necessary data files, classes, and the interface you will implement are provided to you in the source files. Do not confuse interface with graphical user interface (GUI). Interface is an abstract type in Java used to specify a behavior that classes must implement. (If you remember from the Programming Language Concepts course, a Java interface corresponds to a C++ class that all of its methods are purely virtual.)

You should first implement functions that create the necessary tables corresponding to the schema given in Section 3. Then, you should design queries to accomplish the given tasks. Lastly, you should implement the interface using the queries you have designed as they give the desired results when defined parameters are given. You will not implement the Evaluation class. It will be implemented by us to manipulate the database through the predefined interface and evaluate your implementations. Your task is to build up class(es) that implement(s) the provided interface.

2 Schema

You will use (strictly) the schema given below in the scope of this assignment:

- **MenuItems**(itemID:int, itemName:varchar(40), cuisine:varchar(20), price:int)
- **Ingredients**(ingredientID:int, ingredientName:varchar(40))
- **Includes**(itemID:int, ingredientID:int)
- **Ratings**(ratingID:int, itemID:int, rating:int, ratingDate:date)
- **DietaryCategories**(ingredientID:int, dietaryCategory:varchar(20))

MenuItems table holds the item ID, item name, cuisine name that the item belongs to, and the price of the item for all items in the database. An example menu item is as follows:
`itemID:101, itemName:Veggie Noodles, cuisine:Chinese, price:64`

Ingredients table holds the ingredient ID and the ingredient name for all of the items in the database. An example ingredient is as follows:
`ingredientID:1002, ingredientName:soy_sauce`.

Includes table holds the item ID and the ingredient ID for the ingredients included in menu items. An example is as follows:
`itemID:101, ingredientID:1002`.

Ratings table holds the rating ID, item ID, rating score, and the rating date of the item for all of the ratings in the database. An example rating is as follows:
`ratingID:3001, itemID:101, rating:4, ratingDate:2022-07-24`

DietaryCategories table holds separate tuples for all of the ingredients and the dietary categories they belong to. For example, the ingredient "soy_sauce" is vegan, vegetarian, and dairy-free. This means the following tuples exist in the table:
`ingredientID:1002, dietaryCategory:vegan`
`ingredientID:1002, dietaryCategory:vegetarian`
`ingredientID:1002, dietaryCategory:dairy_free`

Your task is to generate a class named `FOODRECDB` (it should belong to package `ceng.ceng351.foodrecdb`) which implements `IFOODRECDB` interface. You can create any additional classes if necessary. `FOODRECDB` class should be able to accomplish the following tasks:

- Create the database tables
- Insert data into tables
- List all menu items that include a given ingredient
- List menu items that do not have any ingredients in the database
- Find ingredients that have not been included in any menu items
- Find the menu item that includes the highest number of different ingredients
- List the average rating of each menu item
- List menu items suitable to a given dietary category
- Find the most used ingredient across all menu items
- Find the cuisines with the average rating of menu items belonging to that cuisine
- Find the average ingredient count of menu items for each cuisine
- Increase the price of all menu items that include a given ingredient by a given amount
- Delete all ratings that have an earlier rating date than the given date
- Drop the database tables

Tasks are explained in more detail in Tasks section. For each task, there is a corresponding method in `IFOODRECDB` interface. You need to implement them in `FOODRECDB` class. Necessary data files (for populating the tables) to accomplish these tasks are provided.

In the `data` folder there are 5 `.txt` files that correspond to each table. You will use these tables when you are inserting data. Data files, interfaces, and classes for fulfilling these tasks will be provided as source files. You can assume all information will be complete and consistent, i.e. all necessary data will be inserted before executing a query. You can find detailed descriptions of the usage of the functions in provided source files. Your results should not include any repetition. Therefore, please do not forget to use the `DISTINCT` keyword when appropriate in your queries.

Make sure you are using Java (version 14) before starting.

2.1 Files

Given `.txt` files for filling the tables are as follows:

- `Test_MenuItems.txt`
- `Test_Ingredients.txt`
- `Test_Includes.txt`
- `Test_Ratings.txt`
- `Test_DietaryCategories.txt`

3 Tasks

3.1 Creating the database tables

You will create all the tables according to the schema described in Section 2. You can assume that tables will be created before executing any other database operation. Do not forget to define **foreign keys** while you are creating tables.
Method: `createTables()`
Output: The number of tables that are created successfully.

3.2 Inserting data into tables

You will insert data into appropriate tables.
Methods: `insertMenuItems()`, `insertIngredients()`,
`insertIncludes()`, `insertRatings()`, `insertDietaryCategories()`
Output: The number of rows created successfully.

3.3 Listing all menu items that include a given ingredient

Method(Input): `getMenuItemsWithGivenIngredient(givenIngredient)`
Output: itemID, itemName, cuisine, price
Order the results by ascending itemID

3.4 Listing menu items that do not have any ingredients

Due to a loss in the system, some menu items have no ingredients entered in the database. You need to find them.
Method: `getMenuItemsWithoutAnyIngredient()`
Output: itemID, itemName, cuisine, price
Order the results by ascending itemID

3.5 Finding ingredients that have not been included in any menu items

Method: `getNotIncludedIngredients()`
Output: ingredientID, ingredientName
Order the results by ascending ingredientID

3.6 Finding the menu item that includes the highest number of different ingredients

Method: `getMenuItemWithMostIngredients()`
Output: itemID, itemName, cuisine, price

3.7 Listing the average rating of each menu item

Method: `getMenuItemsWithAvgRatings()`
Output: itemID, itemName, avgRating
Order the results by descending avgRating

3.8 Listing menu items suitable to a given dietary category

For a menu item to be suitable to a given dietary category `givenCategory`, all the ingredients of this menu item should be from the given dietary category.
Method(Input): `getMenuItemsForDietaryCategory(givenCategory)`
Output: itemID, itemName, cuisine, price
Order the results by ascending itemID

3.9 Finding the most used ingredient across all menu items

Method: `getMostUsedIngredient()`
Output: ingredientID, ingredientName

3.10 Finding the cuisines with the average rating of menu items belonging to that cuisine

For each cuisine, you need to find the average rating of menu items belonging to that cuisine.
note: Having no rating DOES NOT MEAN having a 0-rating. You should not include the 0-ratings in the average, and for the cuisines with no ratings, you should return NULL.
Method: `getCuisinesWithAvgRating()`
Output: cuisine, averageRating
Order the results by descending averageRating

3.11 Finding the average ingredient count of menu items in each cuisine

For each cuisine, you need to find the average ingredient count of menu items belonging to that cuisine.
note: Having no ingredient MEANS having 0 ingredients. You should include them in the average.
Method: `getCuisinesWithAvgIngredientCount()`
Output: cuisine, averageCount
Order the results by descending averageCount

3.12 Increasing the price of all menu items that include a given ingredient by a given amount

If the given ingredient is included in a menu item, you need to increase the price of that menu item by the given amount.
Method(Inputs): `increasePrice(givenIngredient, givenAmount)`
Output: number of rows affected

3.13 Deleting all ratings that have an earlier rating date than a given date

Method(Input): `deleteOlderRatings(givenDate)`
Output: ratingID, itemID, rating, ratingDate
Order the deleted rows by ascending ratingID

3.14 Dropping the database tables

You will drop all the tables (if they exist).
Method: `dropTables()`
Output: number of tables that are dropped successfully

4 Regulations

- **Programming Language:** Java (version 14).
- **Database:** An account on the MySQL server on a remote machine will be created for each of you and an e-mail including credentials and connection configuration will be sent to your metumail. You must use JDBC driver to connect to the database. Your final submission must connect to the MySQL server on the remote machine. So, make sure that the connection information is correct before submitting your homework.
- **Attachments:** Necessary source files and JDBC driver is provided.
- **Input:** All strings will be case-sensitive and they will not include any non-English characters. Note that they may include punctuation.
- **Cheating:** We have zero tolerance policy for cheating. People involved in cheating will be punished according to university regulations.
- **Evaluation:** It is GUARANTEED that input files are correctly formatted and sample data will be given to you. There will be no surprises about the data, similar (and larger) data will be used while evaluating the homeworks. Your program will be evaluated automatically using "black-box" technique so make sure to obey the specifications. Please, be noticed that you have to accomplish tasks only within your SQL queries, not with any other Java programming facilities.

5 Submission

Submission will be done via ODTUClass. Create a compressed file named `ceng.tar.gz` that contains `FOODRECDB.java` and all other classes created by yourself. You will not submit the interface and class files provided by us. So, be sure you do not modify them during implementation. The compressed file should contain a directory tree same as the package tree. That is, you should compress the directory named `ceng` which contains a directory named `ceng351` which contains a directory named `foodrecdb` which contains your source files. The directory tree should look as follows:

```
ceng
└── ceng351
    └── foodrecdb
        ├── FOODRECDB.java
        ├── AnotherClassIfYouNeedIt1.java
        ├── AnotherClassIfYouNeedIt2.java
        ├── ...
        └── AnotherClassIfYouNeedItN.java
```

Important note: Make sure your project compiles outside of the IDE by adapting and executing the following command:

```
javac -classpath .:/path/to/mysql-connector-java-8.0.11.jar FOODRECDB.java IFOODRECDB.java
Evaluation.java FileOperations.java QueryResult.java MenuItem.java Ingredient.java
Includes.java Rating.java DietaryCategory.java AnotherClassIfYouNeedItN.java
```