DAO Exercise

Re-submit Assignment

Due Mar 11 by 11:59pm **Points** 2 **Submitting** a file upload **File Types** java

Complete the exercise in Java and submit your work **here**.

The provided **source code** implements a simple console-based program for managing budgets. It uses the SQLite relational database to persist budget information. Unfortunately, the provided code does not apply the DAO pattern. The database code is distributed throughout the Main class and the various Presenter classes, which makes it difficult to change what database the program is using.

Your task is to refactor this program to apply the DAO pattern. All of the database code should be isolated in appropriate DAO classes that have appropriate abstract interfaces. This includes the database initialization code in the Main class.

You will also need to apply the Abstract Factory pattern to allow the program to create the various DAO objects it needs to interact with the database. No part of the program should know what kind of database is being used, except the few lines of code that initialize the concrete factory object. It should be possible to port the program to a different database by only implementing a new set of concrete DAOs and a new concrete factory to instantiate them.

Ideally, you would move all of the database code into appropriate DAOs, but to minimize the time required for this exercise, you are only required to move the following into DAOs:

1. The database initialization code in the Main class
2. The database code in the ExpensePresenter class

Here is a specific list of tasks you need to perform:

1. Define abstract interfaces for a database initialization DAO and an expense DAO. Put these interfaces in an appropriate package or folder (e.g., dataaccess).
2. Create concrete classes that implement these abstract interfaces, and move the database code from the Main and ExpensePresenter classes into them. Put these classes in an appropriate package or folder (e.g., dataaccess.sqlite).
3. Define an abstract factory interface that the program can use to create the DAO objects it needs. Put this interface in an appropriate package or folder (e.g., dataaccess).
4. Create a concrete factory class that implements your abstract factory interface, and instantiates your concrete DAO classes. Put this class in an appropriate package or folder (e.g., dataaccess.sqlite).
5. Modify the Main class to:
    1. Create the DAO factory to be used by the program

    2. Use the factory to create a database initialization DAO, and call it to initialize the database
    3. Make the DAO factory accessible to the ExpensePresenter (by passing it a reference to the factory, creating a singleton object to hold the factory, or making another factory to return an instance of the DAO factory)
6. Modify the ExpensePresenter to call the DAO factory to create the DAOs it needs to interact with the database. ExpensePresenter should not know what type of database is actually being used.

For **Java**, in your IntelliJ project you will need to create a Maven dependency on the SQLite

driver:org.xerial:sqlite-jdbc:3.28.0

Submission

- Submit the following files as .java files (DO NOT SUBMIT A ZIP FILE)
    - Updated Main.java
    - Updated ExpensePresenter.java
    - Files containing interfaces and concrete implementations for the following
        - DAO Initialization
        - Expense DAO
        - DAO Factory