**CS3042 Database Systems**
**CS3272 Embedded Database Systems**

---

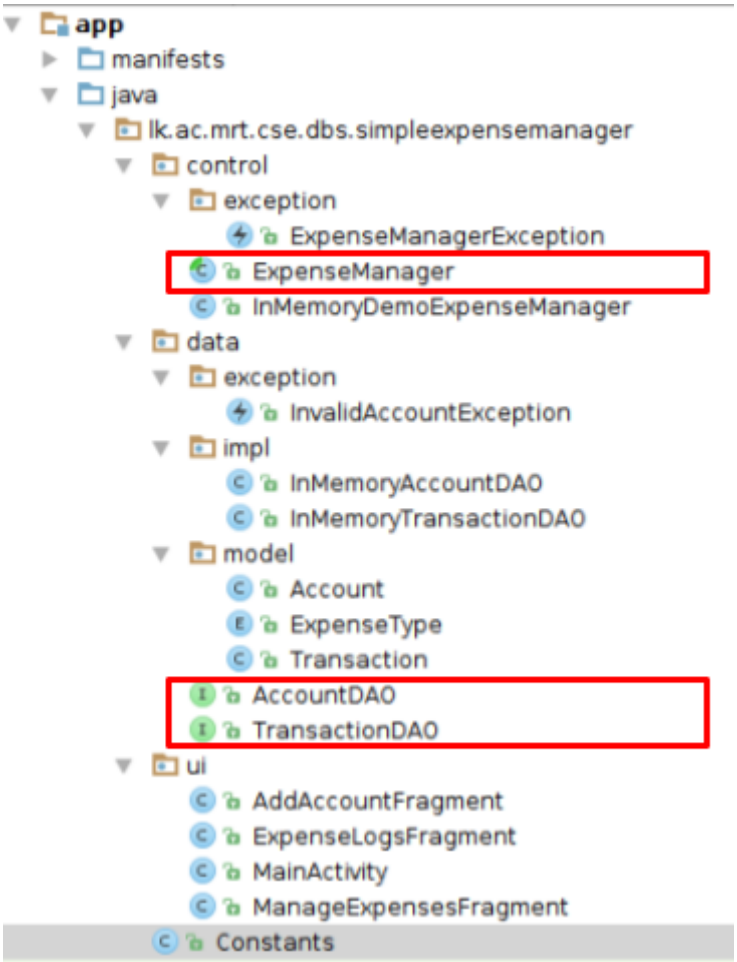
## Mini Project – Embedded Databases

You are required to complete this mini project as an individual exercise and the honor code applies for this. You may refer to online sources if required. Make sure that you read the guidelines properly and submit your answers accordingly. If you do not submit as requested you may get “0” marks for this project.

Instructions you need to follow are at the end of this guide.

# Description

For this assignment we will be self-learning how to use an embedded database in an android application. You have been provided an android application with a skeleton. Following is the structure of the application.



You can look at the current implementation of this application. The current implementation is non-persistent. Therefore all the account information and transactions are stored in the memory and once the application is closed, the information is lost.
Your task is to make the storage of account information and transactions persistent using an embedded database such as SQLite as the persistent storage. You are free to choose the appropriate embedded database.

In order to achieve this you should implement the two interfaces, "`AccountDAO` and `TransactionDAO`". These two interfaces follow the "Data Access Object" Design pattern [1] [2]. You can refer the current In-Memory implementation (`InMemoryAccountDAO`, `InMemoryTransactionDAO`) to get an idea of the current implementation.

After you have implemented the two interfaces, you can use these implementations to setup the application to use the persistent storage instead of the existing in-memory storage. In order to do that you should implement the "`setup()`" method of the abstract class `ExpenseManager`. You can refer the current concrete implementation (`InMemoryDemoExpenseManager`) of this class to get an idea.

After you have completed implementation of the `ExpenseManager` class, go to `MainActivity` class in the ui package and change the existing implementation to your implementation.

eg:

Current implementation

```
/***  Begin generating dummy data for In-Memory implementation  ***/
expenseManager = new InMemoryDemoExpenseManager();
/*** END ***/
```

Your implementation

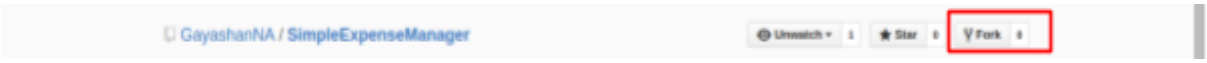
```
/***  Setup the persistent storage implementation  ***/
expenseManager = new PersistentExpenseManager(context);
/*** END ***/
```

You can make improvements to the project as you require. However this project is designed to act as a skeleton and minimize involvement in other components such as the UI. Therefore your main effort should be focused on implementing the persistent storage using an embedded database.

[1] http://www.oracle.com/technetwork/java/dataaccessobject-138824.html
[2] http://www.tutorialspoint.com/design_pattern/data_access_object_pattern.htm

# Instructions

1. Login to GitHub. (If you do not have a GitHub account yet, create one.)
2. Fork the GitHub project - https://github.com/GayashanNA/SimpleExpenseManager




3. Install the Android Studio IDE.
4. Clone your fork of the above project into your Android Studio IDE.
5. Implement the two interfaces
    a. `AccountDAO`
    b. `TransactionDAO`
6. Extend and implement the "`setup()`" method of the abstract class `ExpenseManager.`
7. Change current implementation to your implementation in the `MainActivity` ui class.

    **Current implementation**

    ```
    /***  Begin generating dummy data for In-Memory implementation  ***/
    expenseManager = new InMemoryDemoExpenseManager();
    /*** END ***/
    ```

    **Your implementation**

    ```
    /***  Setup the persistent storage implementation  ***/
    expenseManager = new PersistentExpenseManager(context);
    /*** END ***/
    ```

8. **Name the SQLite database (or the appropriate database) with your Index number.**
9. Commit your code and push to your forked repository in GitHub.
10. Download your project as a Zip from GitHub, rename the Zip file to *<your index number.zip>* and submit as the completed assignment.
11. Submit the link of your GitHub repository along with the zip file.
12. If you run into any issues, raise them in the moodle forum.