CSCI 5408

DATA MANAGEMENT AND WAREHOUSING

ASSIGNMENT – 1

Problem 2: Documentation of Design Principle

Banner ID: B00948977

Git Assignment Link:
https://git.cs.dal.ca/sukumaran/csci5408_f23_b00948977_balaji_sukumaran/-/tree/main/Assignment1

Table of contents

Section 1: Project folder structure..... **1**
Section 2: Factory design pattern for implementing different query types..... **2**
Section 3: Singleton design pattern for Transaction class..... **3**
References..... **4**

Section 1: Project folder structure

The custom database management system application has been built by adapting a **layered architecture**, where the entire codebase has been abstracted and split into the following layers:

- **Application layer:** Contains the code that will be used by the end-user.
- **Business service layer (service):** All the business logic code will be placed in this package.
- **Data access layer (data):** Data access-related code will be placed in this package.
- **Common:** Contains helpers and constants used throughout the entire codebase.

In a layered architecture pattern [1], components are structured in horizontal tiers, with each layer being self-contained and independent from the others. This conventional approach to software design ensures that while components are interconnected, they do not rely on each other for functionality.

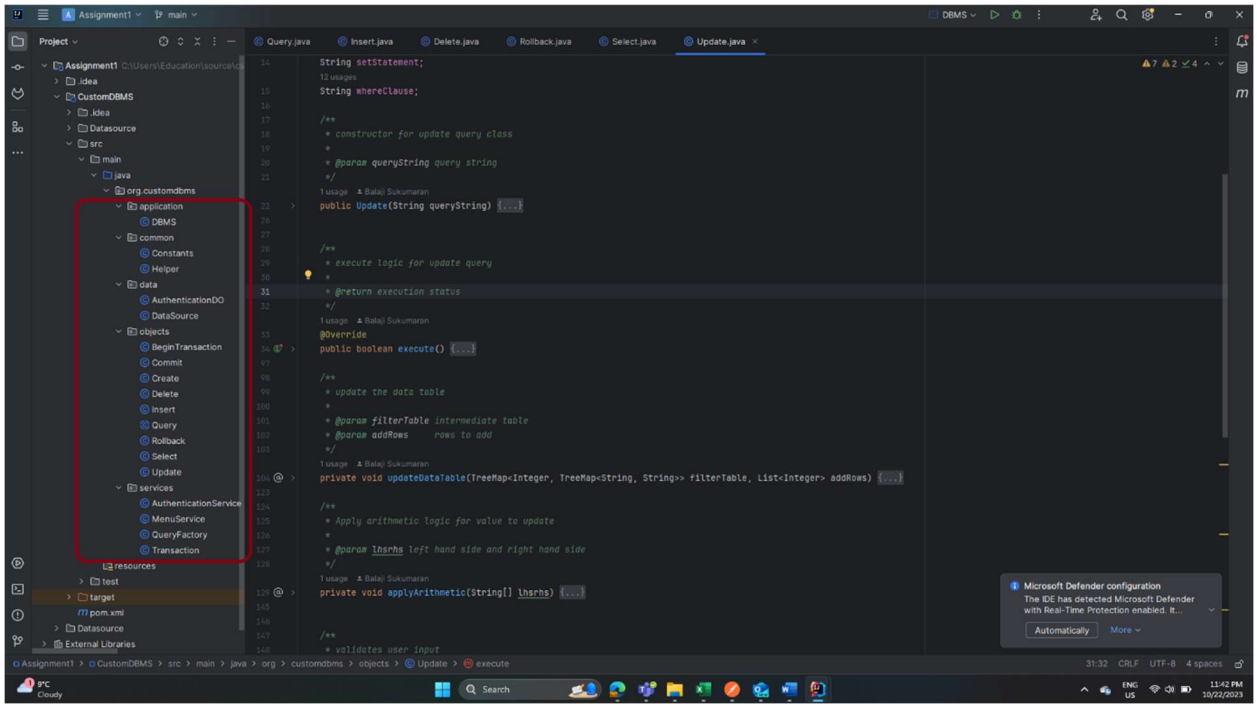

Figure 1: Folder structure for the custom database management system

Section 2: Factory design pattern for implementing different query types.

The factory method is a design pattern that offers an interface or an abstract class to create an object, giving the responsibility to its subclasses to determine the class to instantiate. This pattern falls under the category of creational patterns [2].

Based on the query string provided by the user, the QueryFactory assigns the appropriate object in the control flow, along with its corresponding implementation of the execute method.

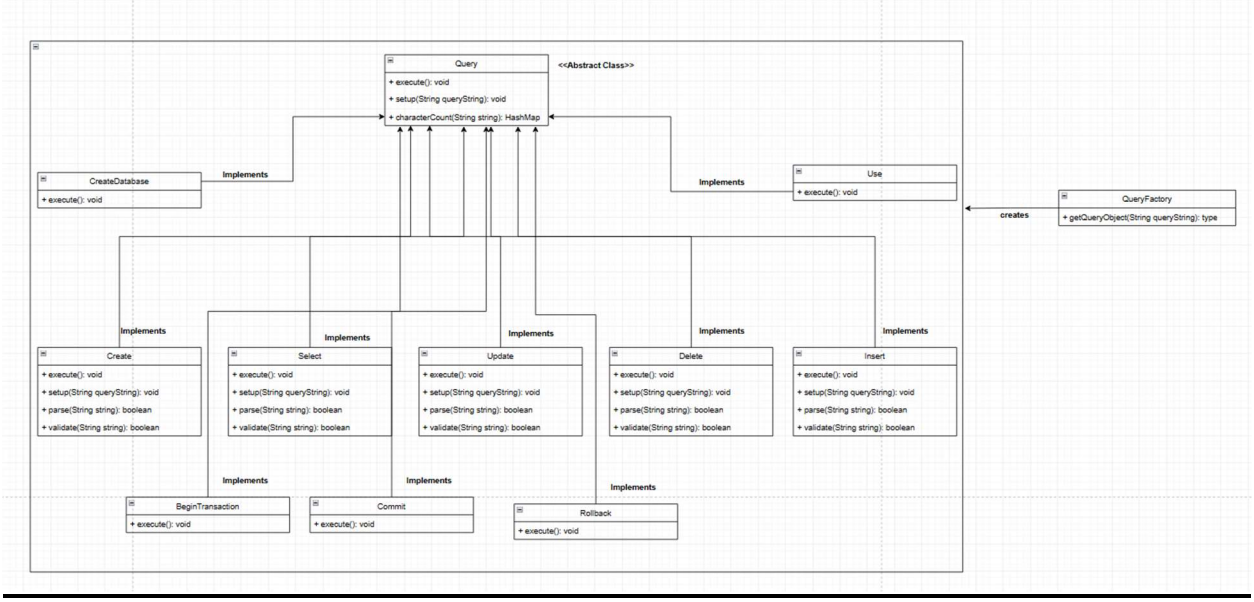

Figure 2: Factory design pattern in custom database management system

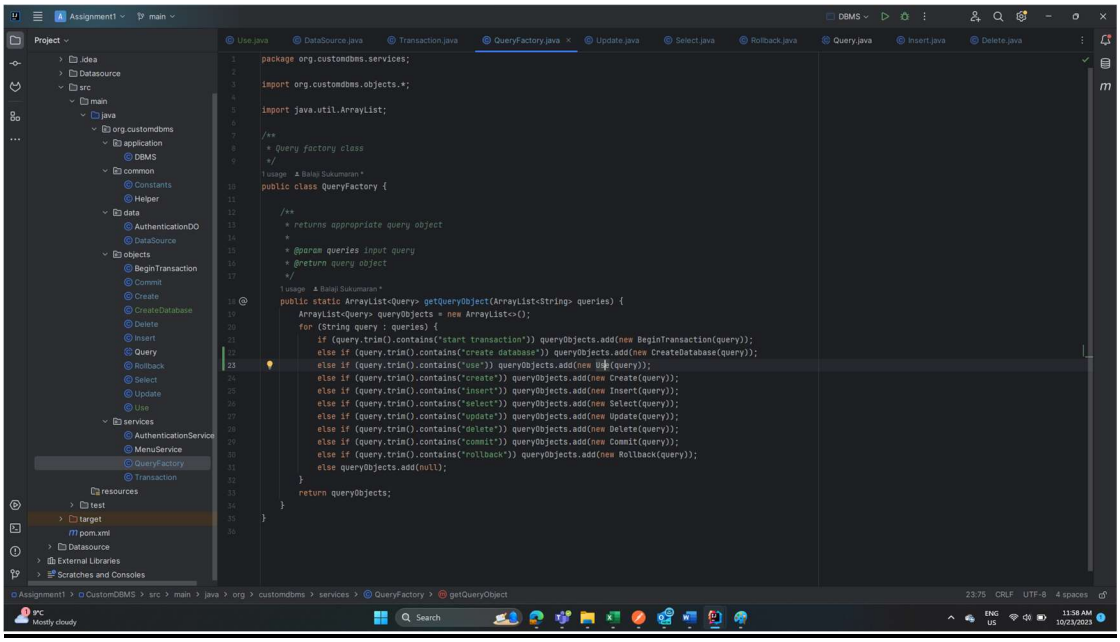

Figure 3: QueryFactory class implementation

Section 3: Singleton design pattern for Transaction class

The Singleton pattern [3] limits the creation of a class to just one instance within the Java Virtual Machine, guaranteeing that a single instance of the class is maintained. It is essential for the singleton class to offer a universal access point to retrieve the class's instance.

Since this is database supposed to maintain a **single transaction** management system. The system includes a thread safe implementation of singleton transaction class.

The getInstance() method is thread safe returns the transaction instance for the application flow.

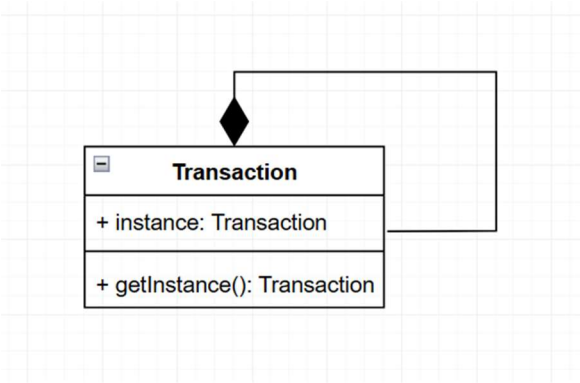

Figure 4: Thread safe singleton transaction class

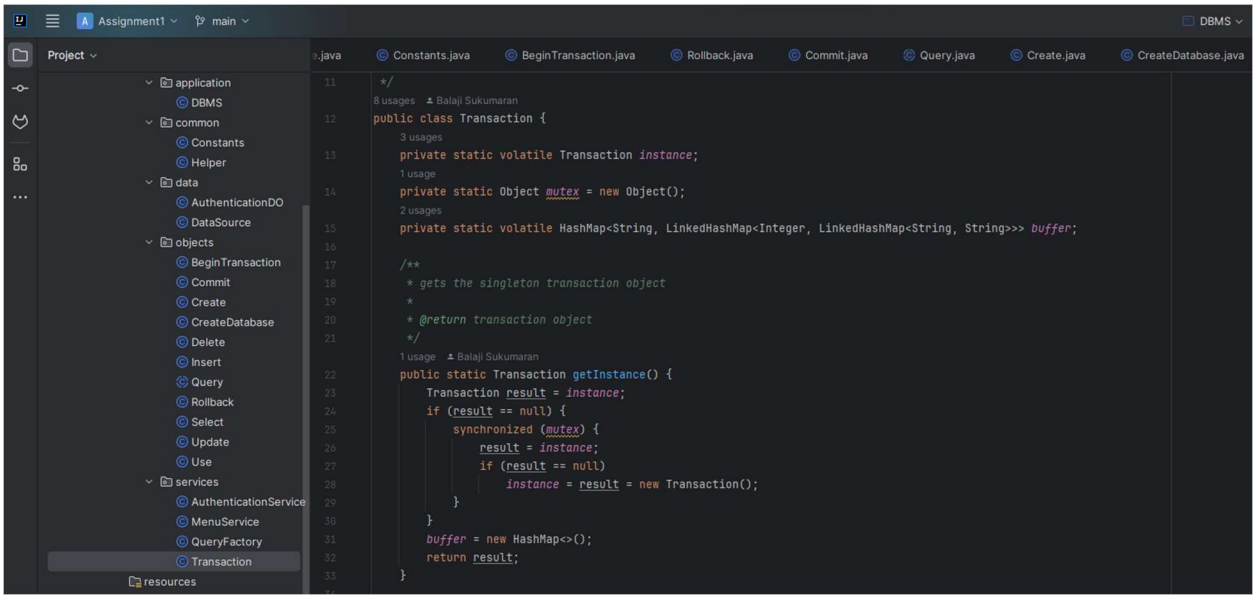

Figure 5: Thread safe implementation of singleton transaction class

References:

[1] Priyal Walpita, “Software Architecture Patterns — Layered Architecture,” *Medium,* [Online], July 9, 2019. Available: https://priyalwalpita.medium.com/software-architecture-patterns-layered-architecture-a3b89b71a057 [Accessed: October 24, 2023].

[2] Arshad Suraj, “Overview Of Factory Method Design Pattern,” *Medium,* [Online], May 23, 2021. Available: https://medium.com/geekculture/overview-of-factory-method-design-pattern-d3a6fe908ea4 [Accessed: October 24, 2023].

[3] P. Pankaj, “Java Singleton Design Pattern Best Practices with Examples,” *digitalocean,* [Online], August 3, 2022. Available: https://www.digitalocean.com/community/tutorials/java-singleton-design-pattern-best-practices-examples [Accessed: October 24, 2023].