Software Development Concepts

Project

Submitted by –

Name - Alen Santosh John

Banner ID – B00930528

Email – al283652@dal.ca

Project Documentation

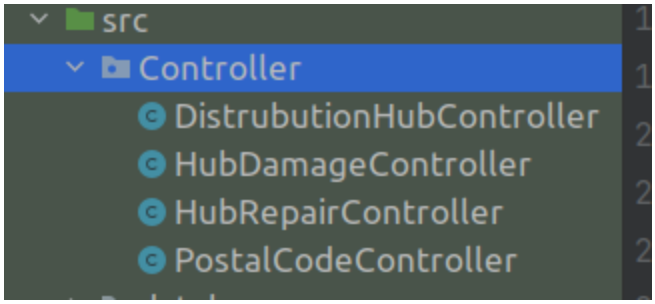
Overview –
This project implements a reporting and planning program for hurricane “Fiona”, that hit Canada in 2022. This program helps to optimize the rate at which power restoration activities are taking place.

Files Structure of the Program –
The program source (src) consists of the following -

1. **Controller -** This package has all the controllers used in this implementation, controllers facilitate the interaction between the model and the view class of the desired method. (explained in the architecture). All the data creation methods have their controllers in this package (see **Figure 1**).

Figure 1 - controller for Adding distribution Hub controller

```
package Controller;

import ...

public class DistrubutionHubController {
    /**
     * Model Object
     * View Object
     */

    private DistrubutionHubModel distrubutionHubModel;
    private DistrubutionHubView distrubutionHubView;

    /**
     * Distrubution hub controller constructor
     *
     * @param distrubutionHubModel
     * @param distrubutionHubView
     */
    public DistrubutionHubController(DistrubutionHubModel distrubutionHubModel,
            DistrubutionHubView distrubutionHubView) {...}

    /**
     * Adding the data through the model class
     *
     * @param hubIdentifier
     * @param postalCode
     */
    public void insertHubServiceArea(String hubIdentifier, String postalCode) {...}
```

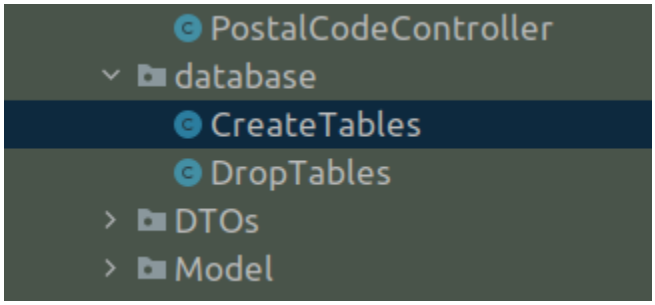


The controller is the only way a model and view class can interact.

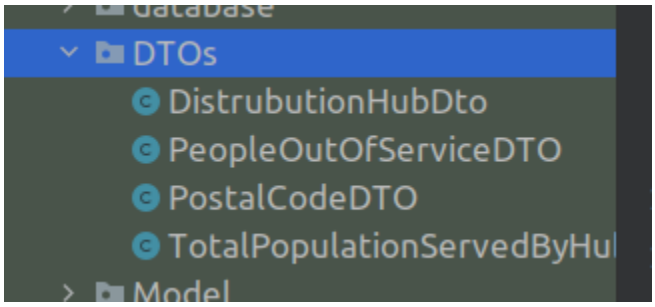
The instances of the model and view are of type private. This ensures data encapsulation

The model class contains all the interactions with the database and the view has a method to view the methods.

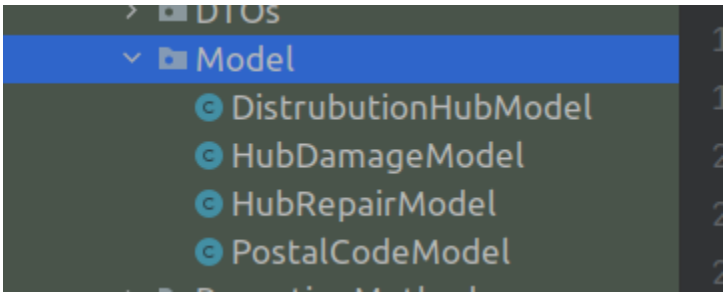
2. **database -** This package contains the code to create and drop tables. The create tables are helper functions for the model classes. All the database-creating logic exists here. The drop tables can be accessed by creating an instance (**The dropAllTables method can be called in case the user wants to drop all the tables**)



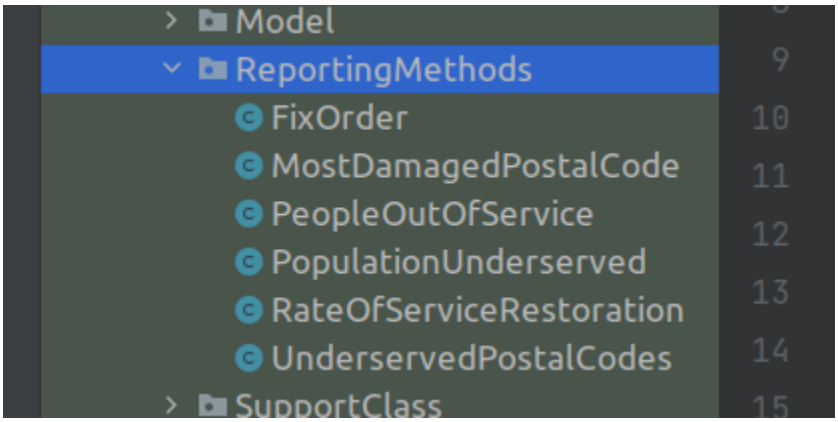
3. **DTOs - Data transfer Objects -** This package contains the blueprint of the data returned from the database. The data returned from a query is stored in a data structure of this type. Storing data in DTOs helps maintain as the data is strictly typed and a no of operations can be performed on it. It also helps maintain consistency.



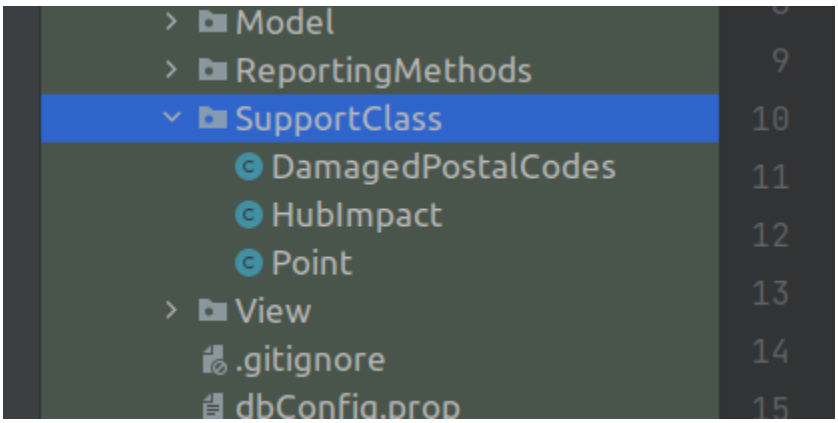
4. **Model -** This consists of all the interactions with the database. Here new values are inserted into the database and/ or new tables are created using the CreateTables.java from the database package.



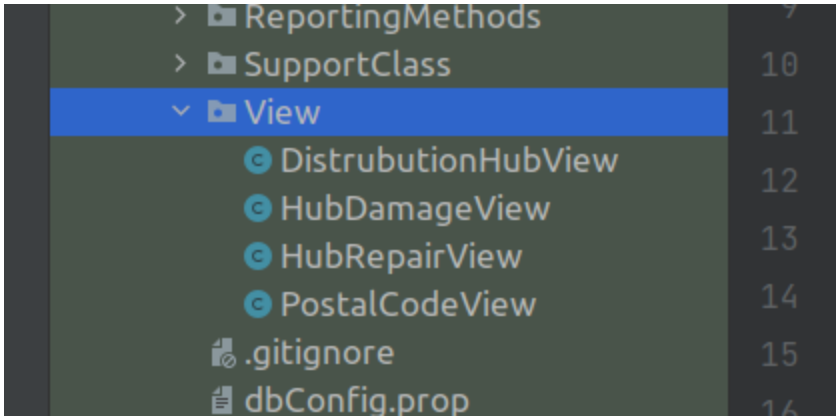
5. **Reporting Methods -** This package contains all the helper classes for the reporting methods called from the power service.



6. **Support Class -** This package contains the support classes as defined by the constraints of this assignment.



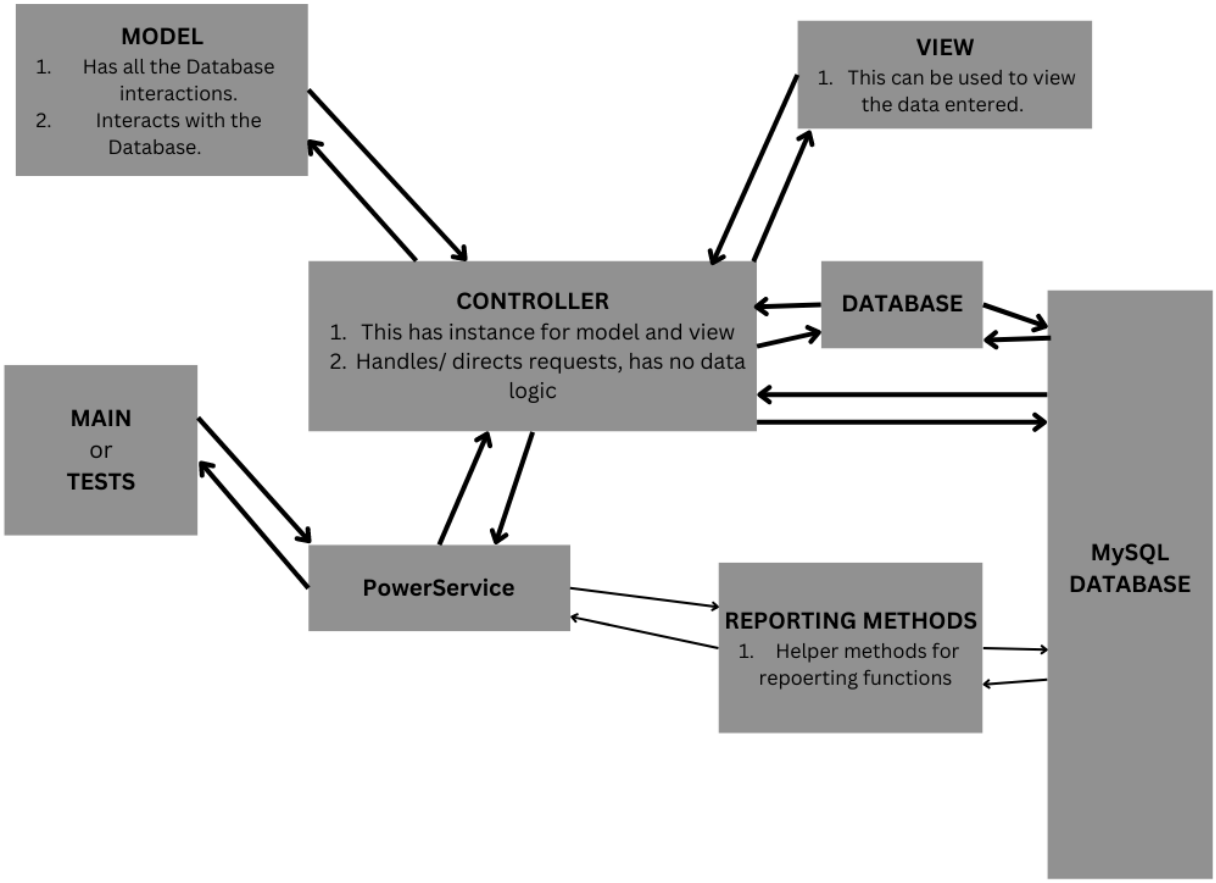
7. **View -** This package contains classes with view methods, i.e console logging. The values entered can be viewed by calling the respective function from the class.



The architecture used for the implementation -

Model view controller -

I have used the model view controller approach for this implementation. In this case, my program is interacting with the database instead of the API. This architecture mainly consists of three components -



1. **Model** - consists of the insert and the creative methods that are used to store data in the respective tables.
2. **View** - consists of a way to visualize the data that is being interacted/ received with. In this case, the view has methods that can be used to view the entered/ received data.
3. **Controller** - This component is between the model and the view components. if a specific function is supposed to be executed example - insert data in the database the user can call the insert method through the controller. No interaction happens directly between the user and the Model.

The architecture can be visualized in **Figure.** The blocks with words in capital letters denote various packages used in the implementation of the remaining blocks denote independent classes.

The reporting methods contain various helper classes for the reporting methods in the power service. The reporting classes also access the database to get the data required for the calculation to provide an answer.

Note - The controller class contains the private instance of the view and the model this helps in data encapsulation and this data cannot be accessed from outside the class.

Data Structure and their relation to each other.

The implementation uses various data structures throughout the implementation. The most common data structures are Lists, and Maps which are types of DTOs returned by accessing the data. Apart from the internal data structures used the implementation also has an external database that is being used.

The tables are created from the implementation of the program. If required the tables created can be dropped from the Drop Tables method in the database package in the implementation for running new test cases.

The multivalue columns that may arise in situations like when a hub is serving multiple locations (i.e the Pincode column is a set with multiple values) the column is broken down into multiple records in the table. This is done in accordance with the 1st Normal Form.
The table used to store the hubs was having the following previously -

1. **hubIdentifier**
2. **x coordinate**
3. **y coordinate**
4. **Pincode**

Here, if a hub is serving multiple locations (Postal Codes) then the x coordinate, y coordinate and hub identifier column may be repeated multiple times. To avoid this I have divided the table into two tables -

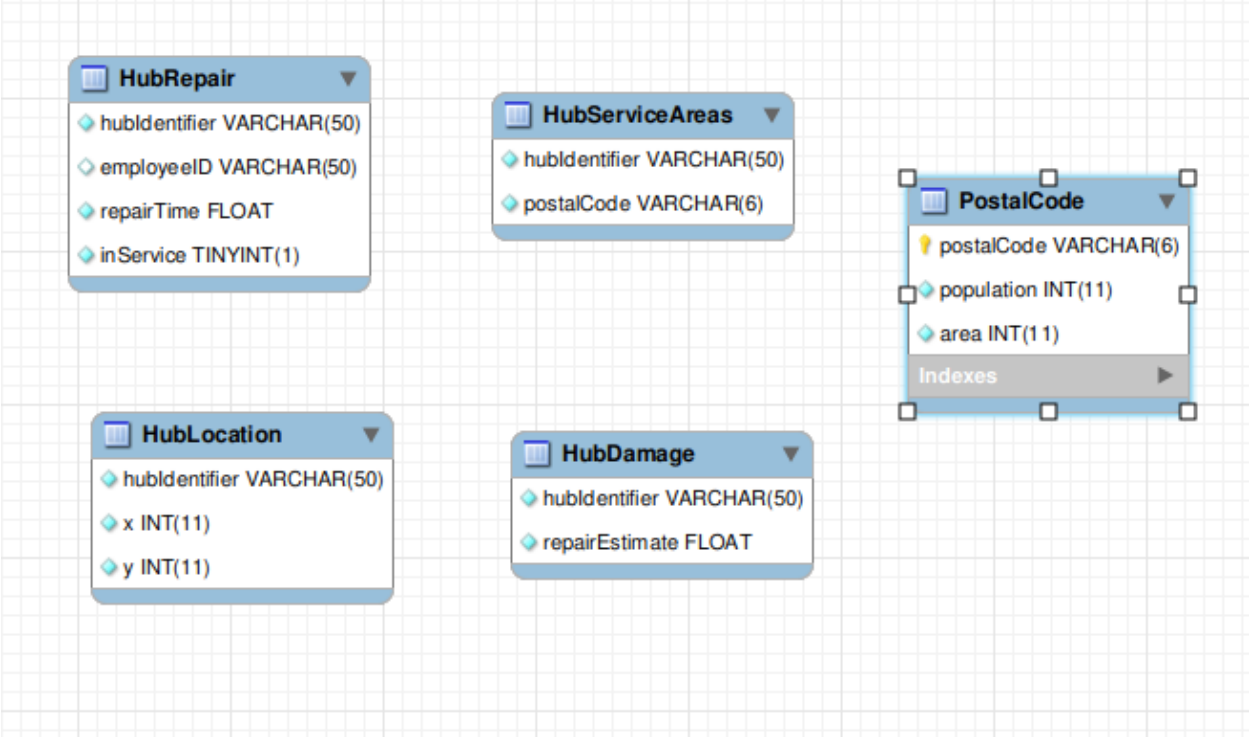
Hub Service area -

1. Hub identifier
2. Pincode

Hub Location

1. Hub identifier
2. x and y coordinates

The 2nd Normal form helps in reducing the redundancy in the data. The following **figure** shows the tables used for this implementation -



Limitations

- The database connection takes a lot of time, the program sometimes takes almost 30 secs to compile and execute when supplied with a large amount of data. This time can be reduced.

- The many-to-many table eg hub service areas could have set the two columns to foreign keys to connect the and reduce data redundancy.

- The reporting methods could have also been accessed from a separate class where the helper methods could have a private access specifier for better encapsulation

Assumptions

- There are some functions like peopleOutOfService that are required to be returning int but due to calculations may return a float value. These values are converted to int or rounded off.

- In case two entries are the same in the resulting query and the reporting method requires us to fetch the top 2 limits, the top two will be returned even if they are the same.

- In case the given limit for some reporting functions like underserved population exceed the returned list the program fetches the entire list. If the limit is smaller than the returned list then the limited (the limit that the user gives) amount of data is returned