Final Project Report

The final project provides reporting and planning activities for the power grid to help maximize the rate at which the electrical grid is fixed so that as many people as possible have their power restored as quickly as possible. The assumptions, implementation details, database design, and test plan are all described in this report.

Assumptions:

1. Exception handling assumptions:
    - I am handling the all the Exceptions and errors using try catch block and printing the message in catch block using getMessage() method.

2. Database related assumptions:
    - Before running the main program, createdB.sql file needs to be run inorder to create a database schema.
    - There is a separate sql file dropallTables.sql file to drop(delete) all the tables in the schema. Also there is a mockdata.sql file to insert data into the tables.
    - There is a separate method resetDatabase.sql method to drop all the tables from the database and also it will create new tables with mock data.
    - There is a method flow in the java code, which needs to be followed otherwise the tables wont be created with mock data properly
    - In this project I have used Dalhousie university's database schema and made the connection to that db and used only that schema in this project.

3. Other Assumptions:
    - Postal Codes identifiers wont overlap in our code , meaning B3W1H5 and B3W wont be provided.
    - Also we wont delete postalcodes once we store them into our program.

Files Included for this Project:

Java Files:

Main.java – to test the dummy data with object call to Power Service Class object PowerService.java - the Power Service class contains all the methods of the power service and the repair plan of that service.

JDBC.java – this class is mainly used to make JDBC connection and it will return the connection object to the Power Service class.

Point.java – the poin class is used to store the x coordinates and y coordinates of the hub location in postalcode.

HubImpact.java - the main functionality of this java file is to store the Hub id and the impact of the hub in the Object.
DamagedPostalCodes.java – the class is used to store the postalcode and estimated repair time for the hubs in the postalcode.

SQL files:

Queries.sql – this file contains all the sql queries for table creation and insertion of data into table.

Database Design:

The database of the power restoration project has 4 tables:

- PostalCodes
- DistributionHub
- HubDamage
- HubRepair

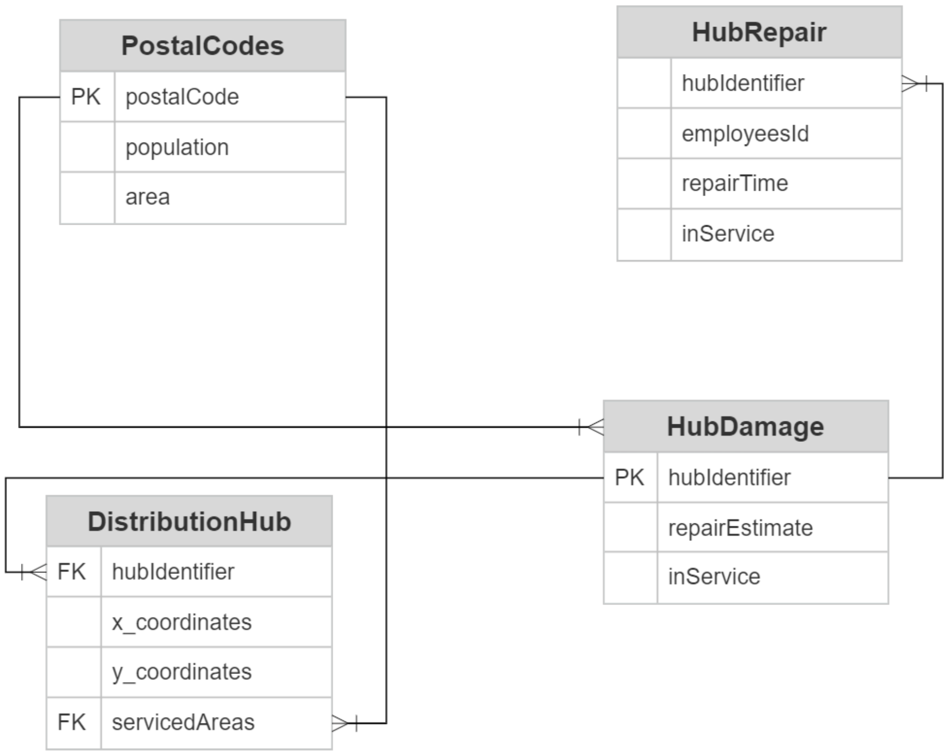
Note : First CreateDb.sql needs to be run before the java code, as this will create the database file and then our java code will insert the queries into this database.

PostalCodes : the tables stores all details for postal codes from the government provided website, such as Population in that postal codes and the total area of that postal code. This data wont be deleted as we are not deleting the postalcodes. To add the fields in this table we are calling addPostalCode() method in the java code. Here I have done the validation of the postalcodes from other tables.

DistributionHub : the table stores the details of Hubs, such as Hub Identifier , x and y coordinates of the hub, postal codes in which the hub is located. In order to store the details of Distribution Hub, we are calling the addDistributionHub method in java code. Here the x and y coordinates are stored as UTM.
Validation of the hub Id and Postal Code from other tables has been done here in this table.

HubDamage : it stores the repair estimates time required to for the hub to restore into fully working condition, also we are storing the status of that hub whether it is in working condition or not. To store all of these values we are calling hubDamage method in our java code.

HubRepair : it is storing the employee id with repair time they took on the hub to repair the hub, also the table store the status of the hub condition, after it got repair from the employess, to store all these details we are calling hubRepair method in java code. Here I have done the hub validation which will validate the hubs from hub Damage table.



E-R Diagram

Test Cases for final Project:

1.Input Validation Test Cases:

1. boolean addPostalCode (String postalCode, int population, int area)

- Send a null string.
- Send an empty string.
- Send a duplicate postalCode.
- Send a population of negative size.
- Send a population as 0.
- Send an area of negative value.
- Send an area of 0 sq. meter.

2. boolean addDistributionHub (String hubIdentifier, Point location, Set<String > servicedAreas)

- Send a null hubIdentifier.
- Send an empty hubIdentifier.
- Send a duplicate hubIdentifier.
- Send a non-alphanumeric hubIdentifier.
- Send a empty location.
- Send an null location.
- Send only y coordinates in location.
- Send only x coordinates in location.
- Send negative x coordinates in location.
- Send negative y coordinates in location.
- Send all negative coordinates.
- Send a null servicedAreas.
- Send an empty servicedAreas.

3. void hubDamage (String hubIdentifier, float repairEstimate)

- Send empty hubIdentifier.
- Send null hubIdentifier.
- Send non-alphanumeric hubIdentifier.
- Send zero repairEstimate.
- Send empty repairEstimate.
- Send negative repairEstimate.

4. void hubRepair (String hubIdentifier, String employeeId, float repairTime, boolean inService)

- Send empty hubIdentifier.
- Send null hubIdentifier.
- Send non-alphanumeric hubIdentifier.
- Send empty employeeId.
- Send null employeeId.
- Send zero repairTime.
- Send null repairTime.
- Send negative repairTime.
- Send non-boolean value in inService.

5. int peopleOutOfService ()

6. List<DamagedPostalCodes> mostDamagedPostalCodes (int limit)

- Send 0 value in limit.

- Send negative limit.

7. List<HubImpact> fixOrder (int limit)
   - Send 0 In limit.
   - Send negative value in limit.

8. List<Integer> rateOfServiceRestoration (float increment)
   - Send negative val in increment value.
   - Send 0 in increment value.

9. List<HubImpact> repairPlan (String startHub, int maxDistance, float maxTime)
   - Send null in startHub.
   - Send empty startHub.
   - Send zero maxDistance.
   - Send negative maxDistance.
   - Send zero maxTime.
   - Send negative maxTime.
10. List<String> underservedPostalByPopulation (int limit)
    - Send a 0 in limit.
    - Send negative in limit.
11. List<String> underservedPostalByArea (int limit)
    - Send 0 in limit.
    - Send negative in limit.

2.Boundary Cases

1. boolean addPostalCode (String postalCode, int population, int area)
   - duplicate postalCode.
   - arbitrary or non-existent postalCode.
   - null postalCode.
   - empty postalCode.
   - valid postalCode.
   - very short or too long postalCode.
   - negative or 0 population.
   - valid population.
   - 0 or negative area.
   - valid area .

2. boolean addDistributionHub (String hubIdentifier, Point location, Set<String > servicedAreas)

   - duplicate hubIdentifier.
   - empty hubIdentifier.
   - null hubIdentifier.

- very short or too long hubIdentifier.
- arbitrary or non-existent hubIdentifier.
- non-alphanumeric hubIdentifier.
- valid hubIdentifier.
- neg coordinates in location.
- invalid location.
- valid location.
- arbitrary or non-existent servicedAreas.
- empty servicedAreas.
- valid servicedAreas.

3.void hubDamage (String hubIdentifier, float repairEstimate)

- duplicate hubIdentifier.
- null hubIdentifier.
- empty hubIdentifier.
- very short or too long hubIdentifier.
- arbitrary or non-existent hubIdentifier.
- non-alphanumeric hubIdentifier.
- valid hubIdentifier.
- negative repairEstimate.
- not float value of repairEstimate.
- valid repairEstimate value.

4. void hubRepair (String hubIdentifier, String employeeId, float repairTime, boolean inService)

- duplicate hubIdentifier.
- null hubIdentifier.
- empty hubIdentifier.
- arbitrary or non-existent hubIdentifier.
- non-alphanumeric hubIdentifier.
- valid hubIdentifier.
- arbitrary or non-existent employeeid.
- valid employeeid.
- negative repairTime.
- repairTime is float type.
- not Boolean of repairTime.

5.List<DamagedPostalCodes> mostDamagedPostalCodes (int limit)

- negative limit.
- valid limit.
- Invalid limit

6. List<HubImpact> fixOrder (int limit)

- negative limit.

- valid limit.
- Invalid limit

7. List<HubImpact> repairPlan (String startHub, int maxDistance, float maxTime)
    - null startHub.
    - empty startHub.
    - arbitrary or non-existent startHub
    - valid startHub.
    - negative maxDistance.
    - valid maxDistance.
    - negative maxDistance.
    - zero maxDistance.
    - valid maxDistance.

8. List<Integer> rateOfServiceRestoration (float increment)
    - negative increment.
    - zero increment.
    - valid increment.

3.Control Flow Test Cases:

1) boolean addPostalCode(String postalCode, int population, int area)

- Invalid postalCode.
- Valid postalCode.
- population is invalid.
- Population is valid.
- Area is invalid.
- Area is valid.

2) boolean addDistributionHub(String hubIdentifier, Point location, Set<String> servicedAreas)

- hubIdentifier is combined of number.
- hubIdentifier is combined of letters.
- hubIdentifier is not alphanumeric.

- Invalid hubIdentifier.
- Valid hubIdentifier.
- No coordinates set in the location object.
- Coordinates is invalid because one value can be zero.
- Coordinates is invalid.
- Coordinates are valid.
- servicedAreas is invalid because it is an empty.
- servicedAreas is valid.

3) void hubDamage(String hubIdentifier, float repairEstimate)

- hubIdentifier is combined of number.
- hubIdentifier is combined of letters.
- hubIdentifier is alphanumeric.
- hubIdentifier is invalid.
- hubIdentifier is valid.
- repairEstimate is negative and invalid.
- repairTime is small and invalid.
- repairTime is too large and invalid.
- repairTime zero or negative and its invalid.
- repairTime is valid.

4) Void hubRepair(String hubIdentifier, String employeeId, float repairTime, boolean inService)

- hubIdentifier is combined of number.
- hubIdentifier is combined of letters.
- hubIdentifier is alphanumeric.
- hubIdentifier is invalid.
- hubIdentifier is valid.
- employeeId is valid
- repairEstimate is negative and invalid.
- repairTime is too small and invalid.
- repairTime is too large and invalid.
- repairTime is negative and invalid.
- repairTime is valid.
- inService is invalid.
- inService is valid.

5) List<DamagedPostalCodes> mostDamagedPostalCodes (int limit)

- DamagedPostalCodes class is invalid.
- DamagedPostalCodes is passed invalid parameters.
- DamagedPostalCodes class is valid.
- Limit is invalid.
- Limit is negative or zero.

- Limit is invalid because not even postalcodes in database.
- Limit is valid.

7) List<HubImpact> fixOrder(int limit)

- Limit is invalid.
- Limit is negative or zero.
- Limit is invalid because number of postalcodes are different .
- Limit is valid.
- HubImpact class is invalid.

8) List<Integer> rateOfServiceRestoration (float increment)

- List is of other datatype.
- Increment is invalid because it is zero or negative.
- Increment is valid.

9) List<HubImpact> repairPlan (String startHub, int maxDistance, float maxTime)

- startHub is invalid.
- startHub is invalid as it is null.
- startHub is valid.
- maxDistance is invalid because of same startHub and endHub.
- maxDistance is having out of order cordinates.

- maxDistance is not of matching datatype.
- maxTime is invalid.
- maxTime is valid.

4. Data Flow Test Cases:

- Call addPostalCode() method multiple times in a row to add the postal Codes, population and area.
- Call addDistributionHub() method multiple times in a row to add the distribution hubs, their location as x and y coordinates, and the servicedAreas in which they are located.
- Call the hubDamage() as many times to add hubId, its repairEstimate, and status of the hub's working conditions.
- Call hubRepair() multiple times in row to add repair work done by the Employee with repairtime taken by the employees and the status of the hub's working condition.
- Call peopleOutOfService() as many time after executing above methods to fetch the count of people those who are outOfService.
- Call mostDamagedPostalCodes() as many time after executing above Methods to retrieve the list of mostDamagdPostal codes
- Call fixOrder() as many time as needed to get the order of hubs
- Call other methods as many time as needed.