Software Development Concepts

Milestone 2

Submitted by –

Name - Alen Santosh John

Banner ID – B00930528

Email – al283652@dal.ca

Test Cases

Input Validation -

1. **addPostalCode**
    - String postalCode is null -> returns false
    - String postalCode is empty -> return false
    - wrong data type entered for int area -> throws exception
    - wrong data type entered for int population -> throws exception
2. **addDistrubutionHub**
    - String hub identifier is null -> returns false
    - String hub identifier is empty -> returns false
    - serviced area set is null -> returns false
    - serviced area set is empty -> returns false
    - point location has invalid data -> returns false
3. **hubDamage (void no returns)**
    - String hub identifier is null
    - String hub identifier is empty
    - wrong data type entered -> throws exception
4. **hubRepair (void no returns)**
    - String hub identifier is null
    - String hub identifier is empty
    - employeeid identifier is null
    - employeeid identifier is empty
5. **mostDamagedPostalCode**
    - wrong data type entered -> throws exception
6. **fixOrder**
    - limit is negative -> returns empty list
    - wrong data type entered -> throws exception
7. **rateOfServiceRestoration**
    - wrong data type entered -> throws exception
8. **repairPlan**
    - starthub is null -> returns empty list
    - starthub is empty -> returns empty list
    - max distance is 0
    - max distance is negative
9. **underservedPostalByPopulation**
    - wrong data type entered -> throws exception
10. **underservedPostalByArea**
    - wrong data type entered -> throws exception

Boundary Cases -

1. **addPostalCode**

- postalcode has special char other than numbers and char -> returns false
- Postalcode entered has more than 6 char -> returns false
- postalcode has only 1 char -> returns false
- population is negative -> returns false
- area is negative -> returns false

2. **addDistrubutionHub**
   - hubidentifier has only 1 char -> returns true
   - hubidentifier is long and has many char -> returns true
   - serviced areas has only 1 elements -> returns true
   - serviced areas has only many elements -> returns true
   - serviced areas has invalid postal codes -> returns false
3. **hubDamage (void no return)**
   - hubidentifier has only 1 char
   - hubidentifier is long and has many char
   - repair estimate is zero
   - repair estimate negative
   - repair estimate exceeds max float value (overflow)
4. **hubRepair (void no return)**
   - hubidentifier has only 1 char
   - hubidentifier is long and has many char
   - employeeId has only 1 char
   - employeeId is long and has many char
   - repair estimate is zero
   - repair estimate negative
   - repair estimate exceeds max float value (overflow)
5. **mostDamagedPostalCode**
   - limit is negative -> returns empty list
   - limit value is more than the total no of postal codes -> returns empty list
   - limit is zero -> returns empty list
   - limit is 1 -> returns list with one element
   - limit is equal to the total no of postal codes -> returns a list with elements
6. **fixOrder**
   - limit is negative -> returns empty list
   - limit value is more than the total no of postal codes -> returns empty list
   - limit is zero -> returns empty list
   - limit is 1 -> returns list with one element
   - limit is equal to the total no of postal codes -> returns a list with elements
7. **rateOfServiceRestoration**
   - increment is zero -> returns a list of elements
   - increment is negative -> returns empty list
   - increment is equal to the max float value -> returns a list of elements
8. **repairPlan**
   - starthub has only 1 char -> returns list

- starthub is long and has many char -> returns list
- maxDistance is negative -> returns empty list
- maxDistance is zero -> returns empty list
- max distance is equal to the max value of Int -> returns list
- max time is zero -> returns empty list
- max time is negative -> returns empty list
- max time is max int value -> returns list

9. **underservedPostalByPopulation**
    - limit is negative -> returns empty list
    - limit value is more than the total no of postal codes -> returns empty list
    - limit is zero -> returns empty list
    - limit is 1 -> returns list with one element
    - limit is equal to the total no of postal codes -> returns a list with elements
10. **underservedPostalByArea**
    - limit is negative -> returns empty list
    - limit value is more than the total no of postal codes -> returns empty list
    - limit is zero -> returns empty list
    - limit is 1 -> returns list with one element
    - limit is equal to the total no of postal codes -> returns a list with elements

Control Flow -

1. **addPostalCode**
    - postalCode already exists -> returns true (replaces old entry)
    - postalCode do not exists -> returns true
    - area is in a different units -> returns false
2. **addDistrubutionHub**
    - hub identifier already exists -> returns true (replaces old entry)
    - hub identifier do not exists -> returns true
    - postalCode do not exists -> returns true
    - location provided is invalid -> returns false
    - postal codes provided within servicedAreas is not valid -> returns false
    - empty servicedArea set -> returns false
    - location points to a different hub - > (returns true replaces old entry)
3. **hubDamage (void no return)**
    - hub identifier already exists
    - hub identifier do not exists
    - repairtime is invalid i.e in different format or negative
4. **hubRepair (void no return)**
    - hub identifier already exists
    - hub identifier do not exists
    - employee id already exists
    - employee id do not exists

- repair time is greater than estimated time of repair
- inService true even if hub is not operational

5. **peopleOutOfService**
    - if all hubs are serviced -> returns 0
    - if no hubs are serviced-> returns total no of people out of service
    - multiple hubs serve one postal code -> returns the no of people out of service
6. **mostDamagedPostalCode**
    - limit is greater than the no of total available postal codes -> returns empty list
    - limit set after all hubs are repaired -> returns empty list
    - repair time for multiple hubs are same -> returns list based on the no of population affected (assumption)
7. **fixOrder**
    - limit is greater than the no of total available postal codes -> returns empty list
    - limit set after all hubs are repaired -> returns empty list
    - 2 or more hubs have same population -> returns list
    - repair time for multiple hubs are same -> returns list
8. **rateOfServiceRestoration**
    - all hubs are repaired -> returns empty list
    - no hubs are repaired -> returns list
9. **repairPlan**
    - start hub already been serviced -> returns empty list
    - maxdistance is less than the distance of the endhub -> returns empty list
    - no other hubs exists between start and end hub -> returns list
    - repair time is more than the maxTime -> returns list with other hubs
    - end hub has already been serviced -> returns list with all other hubs
10. **underservedPostalByPopulation**
    - all hubs are serviced -> returns empty list
11. **underservedPostalByArea**
    - all hubs are serviced -> returns empty list

Data Flow -

1. addPostalCode() called first
2. addDistrubutionHub() called before addPostal code
3. hubDamage() called
    - after postalCode()
    - before addDistrubutionHub()
4. hubRepair called -
    - called after hubDamage()
    - called after addDistrubutedHub()
    - called before addPostalCode()
5. peopleOutOfService()
    - called before addPostalcode()
    - called after hub repair()

    - called before add distribution
6. mostDamagedPostalCode()
    - called before addPostalCode()
    - called before peopleOutOfService
    - called before hub damage
    - called before hub repair()
7. fixOrder()
    - called before addPostalCode()
    - called before addDistrubutionHub
    - called before hub repair
    - called before hub damage
    - called before people OutOfService
8. repairPlan()
    - called before add postal code
    - called before add distribution hub
    - called before hub damage
    - called before hub repair
    - called before people Out of service
    - called before fix order
9. underservedPostalByPopulation
    - called before add postal code
    - called before add distribution hub
    - called before hub repair
    - called before hub damage
    - called before hub repair
    - called before people Out of service
    - called before fix order
    - called before repairPlan()
10. underservedPostalByArea
    - called before add distribution hub
    - called before add postal code
    - called before hub damage
    - called before hub repair

Plan and Timelines

27/11/2022 - Start with the structure of the project, designing of solution- thinking about the algorithm to be used, data structure to be used and the complexity of the solution

creation of classes, and helpers. Starting with the implementation. creating the database and stored procedures to store data. Test the flow of data i.e data addition to database and data retrieval from the database.

6/12/2022 - start with the internal and external documentation, implementation of various methods, Data is successfully stored and retrieved from the database and completion of unfinished methods.

7/12/2022 - external documentation submission, creation of unit test cases.

10/12/2022 - completion of unfinished methods and checking for modularity

12/12/2022 - code refactoring and modularity and cleaning

13/12/2022 - checking all test cases and bug fixing

14/12/2022 - reviewing code and internal documentation, documenting test cases through internal comments

15/12/2022 - final review and submission.