DALHOUSIE
UNIVERSITY
Inspiring Minds

Project – Milestone 3

External Documentation

Author	Vamsi Krishna Utla
Email	vm271757@dal.ca
Student ID	B00870632
Created Date	14-10-2020

Data structures and their relations to each other

MobileDevice

- An array list consisting of a contact node at each location. The contact node in turn consists of details such as contact's hash, duration, date and a flag to identify if it has been synced with central database or not.
- A list of test hashes that resulted in a positive case.

Government

- A table to store contact information with columns as contact_from, contact_to, duration, date and a flag to identify if the contact has already been used for reporting the positive notification to 'contact_from'.
- A table to store information associated to tests which contains details such as test hash (unique identifier), date and result.
- A map of devices hashes as key that are tested positive and value as date.
- A map of device hashes as keys that were gathered on a given day and Boolean value to see if they have already been considered in one of the large gatherings.
- A two dimensional array list of device hashes that were in contact on a gathering on a given day where each row contains information associated to a contact in three columns i.e., 1st columns as contact one, 2d column as contact two and 3rd column as duration of contact (all strings).
- A set of device hashes to construct a set 'S' for finding the number of large gatherings.

Code Design

MobileDevice

MobileDevice(String configurationFile, Government contactTracer)

- The constructor uses default hash function to develop a hash value with the help of address and device name which are given as part of configuration file. This hash value is used as a unique identifier for each mobile device.

- In addition, the instance of contact tracer is also stored locally for later processing.
- The constructor throws required exceptions to represent the issues with respect to input parameters.

recordContact(String individual, int date, int duration)

- This method appends the contact to the list of contacts if it's not already present. Else, it will over write the previous contact information (if the contact already exists with old contact details).
- By default, the flag for each new contact is set to false so as to represent that the contact is still not yet synced with central database of the government.
- The method throws necessary exceptions in case of invalid inputs to better control the flow of the program.

positiveTest(String testHash)

- This method appends the new test hash to the list of tests that are resulted in a positive case.
- It throws required exception in case of any invalid input parameter.

synchronizeData()

- This method formats the data in an XML format by following the below sequence of steps:
    - Initially, this method appends the header of the XML to the result string.
    - The method will then iterate through the list of contacts and appends them to the result string if the contact has not already been synced previously. The details such as contact's unique hash, date and duration is appended as a separate row as part of appending the contact.
    - The method also appends the positive test hashes to sync with central database of the government.
    - In the end, the end tags are also appended to close the XML format without any issues.

   - Sample format of the XML:

     sample_xml_format.txt

- Finally, this method calls mobileContact() with the help of the contactTracer which is an instance of type Government by passing the device's hash and result string as parameters to it.
- The method will also return true/false to identify if the device has been in contact with other device whose user/owner has been tested positive for COVID in the last 14 days.
- In addition, the method also handles the exceptions thrown my mobileContact() in case of invalid inputs.

Government

Government(String configurationFile)

- The constructor is used for accessing the details required for connecting to the database. The details are present in the configuration file as per the format mentioned in the problem document.
- It uses Connection object and Class.forName() for connecting to the required database of the government.
- Tables for recording contacts and test details are also created in this constructor by using the necessary statements if they are not already present in the database.
- The constructor throws required exceptions in case there is any issue with respect to the data present in the configuration file.

mobileContact(String initiator, String contactInfo)

This is the core method which stores contact information provided by mobile device in the central database of the government.

- The method extracts the contact information such as contact's hash, date and duration and then imports into the database in the contacts table. The initiator is used as contact_from (column) and contact's hash is used as contact_to (column). All the new contacts by default have a flag set to false indicating that they have not yet been considered for reporting.

- In case of duplicates, the method will either update the previous value or ignore if there are no changes to the contact information.
- Also, the method will store the device's hash and date into the map if a positive test has been reported in the contact info string.
- The method will then query out the rows from the contacts table where contact_from matches the initiator's hash and then filters the result with rows that are not yet reported i.e., rows which have a flag set to false. These device hashes are then compared with the device hashes stored in map to check if the initiator has been in contact with any new COVID positive devices.
- Finally, the method will return true if any instance of the device hash is found in the map indicating COVID positive contact, else the method will return false.

recordTestResult(String testHash, int date, Boolean result)

- This method uses a query to insert the details of tests to the database by using the Statement object.
- It also throws necessary exceptions to represent the faulty scenarios.

findGatherings(int date, int minSize, int minTime, float density)

- The method initially queries out all the contacts that were made on a given date. It then adds the contacts to the two dimensional array list along with duration of contacts.
- The method then iterates through each contact from the array list containing contacts as 'A' and 'B' and checks if the map already contains 'A' and 'B' with true values. If yes, then next iteration is considered ignoring the current one as 'A' and 'B' have already been considered as a part of other large gathering. If not, the method identifies the devices that got into contact with 'A' and 'B' on the given day and then adds them to the set 'S' of devices for the current iteration.
- The method will then identify the 'c' by counting the number of contacts between the devices in set 'S' to calculate the expression 'c/(n(n-1)/2)' where 'n' is the number of contacts in the set 'S'. If the result is more than density, counter for large gathering is incremented and all the contacts in set 'S' are updated as true in the map to indicate that the contacts have already been used in one of the large gatherings and 'S' is cleared. If the result is less than density, the set 'S' is simply cleared.

- In both cases, the method will then move to next iteration until all contacts in the array list are covered.
- The method throws exceptions in case of any invalid scenarios.

Note: The exceptions mentioned for each method will be explained in detail in next sprints as the details are yet to be finalized with respect to designing the exception types and messages.

Key algorithms

No key algorithms were used for implementing a solution to this problem.

Additional white box tests

- Add contact details to central database of the government.
- Add test details to central database of the government.
- Ability to retrieve the data from the tables in the central database of the government.
- Check to see if the tables can be created in the central database of the government.

Status of feature development

The following plan is not a final plan and changes will be updated in case of any unplanned events on a regular basis.

Estimated date of completion	Feature	Class	Status
10-Nov-20	Analyze the problem statement and design the blue print of the solution architecture.	N/A	Completed
15-Nov-20	Create a new hash for each device based on the inputs in the configuration file.	MobileDevice	Completed
	Ability to store a positive test result.	MobileDevice	Completed
	Ability to record new/old contacts.	MobileDevice	Completed
	Format the contacts data in order to use it for sync purpose.	MobileDevice	Completed
20-Nov-20	Ability to read details and connect to a database from the given configuration file.	Government	Not Started
	Add the mobile contacts of a particular mobile device to a database.	Government	Not Started
	Ability to add test result details to a data base.	Government	Not Started
22-Nov-20	Ability to find large gatherings	Government	Not Started
25-Nov-20	Main class for interaction between end user and program.	N/A	Not Started
30-Nov-20	Refining Test Cases Document	N/A	Not Started
	Testing the implemetation and correcting bugs	N/A	Not Started
6-Dec-20	Documentation	N/A	Not Started
12-Dec-20	Final Review	N/A	Not Started