CSCI 3901 FINAL PROJECT

MILESTONE 3 and 4

EXTERNAL DOCUMENTATION OF DATA STRUCTURES

Mobile Class

<u>MobileDevice(String configurationFile, Government contactTracer)</u>

1) Reading configurationFile using **FileInputStream**.
2) Using Properties(**java.util.Properties**) to pass this FileInputStream and get deviceName and networkAddress.
3) Using **regex** to validate network address and input validation for both network address and device name.
4) Concatenating deviceName and network address and generating hash value.
5) Saving the hash value and Government object in global class variables.

<u>recordContact(String individual, int date, int duration)</u>

1) A new **class Contact** is created which has variables hash, date and duration.
2) Parameters passed in recordContact method are added in this class and stored in a global list called contactList.
3) Incase of any SQL connection error, the data sent in mobileContact() method should be sent again. Again, updating the file would be difficult. So, all the contact information is stored as a **List<Contact>,** in synchronize(), the contacts are written in a file and passed.
4) After input validation, the values are added in the global list.
5) **LocalDate** is used to calculate the number of days from Jan 1,2020, to check if given date is not future date.
6) Duration is checked if it exceeds 1440(1 day=1440 minutes)

<u>boolean synchronize()</u>

1) **Xml** file is created with hash value as file name
2) Iterating the global List<Contact> and updating the XML file.
3) Each contact recorded is appended to the XML file using **Files.write.**

4)

```
<mobile>
        <device>
                <hash>......</hash>
                <date>......</date>
                <time>......</time>
        </device>
        <device>
                .........
        </device>
</mobile>
```

5) Positive test hash value is also appended into the XML file.
6)

```
<positive-test>
        <test-hash>……. </test-hash>
</positive-test>
```

7) mobileContact() method is called and hash value and file name is passed as arguments.

Additional functionalities to be handled:

1) Exceptions to be handled accordingly
2) In case of SQL connection error, or when DB is down, the old values should be sent again with the new ones.

positiveTest(String testHash)

1) After validating the test, the hash value is saved in a global variable.
2) Hash value is added to the XML file only during the synchronize() call.
3) Hash value is added to fp_mobile_dtls.test_hash column.
4) test_hash column in fp_mobile_dtls is foreign key which refers to fp_test_dtls.test_hash column.
5) If hash passed is doesn't match with the one in fp_test_dtls, it will throw an exception which will be passed to synchronize() and will be notified to the user using print statements.

Government class

<u>Government(String configFile)</u>

1) **Connection** variable is declared as a global variable.
2) Configuration file is read using **FileInputStream**.
3) Using Properties(**java.util.Properties**) to read the configuration parameters.
4) Values stored in the property file are,
    - hostURL=localhost
    - port=3306
    - username=
    - password=
    - schema=finalproject
5) SQL connection is created after framing URL and passing Database credentials and is stored in the local class variable.

<u>boolean mobileContact(String initiator, String contactInfo)</u>

1) In database, I am maintaining a separate table to store all mobile hash**. fp_mobile_dtls**.
2) To store the contact information and other details, another table **fp_device_log** is created.
3) If mobile hash passed is not registered in the fp_mobile_dtls, I am inserting a new record in the fp_mobile_dtls before proceeding.
4) To store the mobile hash in fp_mobile_dtls, I am using **createStatement()**.
5) To save the other contact values, I am calling one private method to the read the data from the XML file and insert into database.
6) Connection is created only once, when the object is created and it is used by all the methods.

fp_mobile_dtls
mobile_id
mobile_hash
test_hash

Column info:

- mobile_id – primary key
- mobile_hash – unique
- test_hash – forign key (Table : fp_test_dtls, coloumn : test_hash)

fp_mobile_log
log_id
mobile_hash
neighbour_hash
date
duration

Column info:

- log_id – primary key
- mobile_hash– foreign key (table : fp_mobile_dtls, coloumn: mobile_hash)
- neighbour_hasah - foreign key (table : fp_mobile_dtls, coloumn: mobile_hash)
- date
- duration

private void saveContact(String initiator,String contactInfo)

1) I am using **prepared statement** for inserting the contact information.
2) I am inserting the records in the fp_device_log table.
3) Prepared statement is created as ,
   "INSERT INTO fp_device_log (mobile_hash, neighbour_hash, date, duration) values(?,?,?,?)"
4) **File** object is created for the XML file passed.
5) **DocumentBuilderFactory**, **DocumentBuilder**, **NodeList**, **Node**, **Element** packages are used to read from the **XML** file.
6) Nodelist is iterated for each <device>…..</device> values and for each iteration, the contact is updated in the db.
7) Sql query is executed in the loop iteration because, if the values are appended to a string and executed at the end, incase of large data, the operation might fail. So, in every iteration, query is inserted.

Pending functionalities:

1) Functionality to add test_hash from positiveTest() method is pending.
2) Return if the person has been near any affected persons in the last 14 days.
   - I am planning to use **GROUP_CONCAT** and **FIND_IN_SET** for this operation

```
SELECT COUNT(*) AS countValue FROM fp_test_dtls WHERE test_result="positive" AND
FIND_IN_SET (test_hash,
 (
                SELECT GROUP_CONCAT(test_hash) FROM fp_mobile_dtls WHERE
                FIND_IN_SET(mobile_name,
                                (
                                        SELECT GROUP_CONCAT(neighbour_hash)
                              FROM fp_device_log WHERE mobile_hash=1 AND DATE =10
                                   )
  ))
```

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

1) From database, I am passing the date value to the fp-devivce_log table and fetching the values from that table accordingly.

```
String sql=
"SELECT mobile_hash, GROUP_CONCAT( CONCAT(neighbour_hash,"=",duration ) )AS
neighbour_hash FROM fp_device_log WHERE date="+date+" GROUP BY mobile_hash";
```

- I am grouping the results by mobile_hash
- I am using **GROUP_CONCAT** to concat the neighnour_hash.
- I am **concatenating** neighbour_hash and duration column into **single column** with **“=”**
- Result will be like this,

Mobile_hash	**Neighbour_hash**
A	B=30, C=15, D=10
B	A=30, E=6, F=12
C	A=15, D=12
D	A=10, C=12, E=12
E	B=6, D=12, F=9
F	B=12, E=9

- I am storing this result, in a Map of Map as,

  Map< String, Map<String,Integer>>

  So, the map will typically look like,

```
{
    A={B=30, C=15, D=10},
    B={ A=30, E=6, F=12},
    C={ A=15, D=12},
    D={ A=10, C=12, E=12},
    E={ B=6, D=12, F=9},
    F={ B=12, E=9}
}
```

So, I am iterating the Map, and for the first pair of individual, I will take the map key and corresponding inner key as the first pair.

- Map will start iterating from ‘A’. A- first individual.
- Second individual should be connected with ‘A’ and not the immediate next element in the query result. So, there is one more for loop iterating the value (Map with list of connected nodes) inside the main for loop(iterating the whole loop).
- So, second individual will be ‘B’ and in a set, I am appending all neighbour nodes of A and B. I am using SET to avoid duplicates.
- If the set.size() is > minSize, I am calculating c (count the number of pairs of individuals c within S who contacted one another on the given date for at least minTime minutes).
- ‘c’ calculation is still pending (Logic to be determined).
- N – set size.
- If density calculated is greater than the density passed, increasing the gathering count and removing the nodes in Set from the Set of available nodes to search.
- Before the loop, I am saving all the nodes in a Set. So, if a large gathering is deducted, the nodes are removed from the set and Map iteration will occur only for the nodes which are in the set.

Pending List:

1) positiveTest() method and recordTestResult() to be implemented.
2) ‘c’ value calculation in findgatherings()
3) Input validation and Exception handling in all methods.
4) Minimising the overall code, database tables and other functions.

SQL scripts:

```
CREATE SCHEMA finalproject;

USE finalproject;

CREATE TABLE `fp_test_dtls`
(
 `test_id` INT(11) NOT NULL AUTO_INCREMENT,
 `test_date` INT(11) NOT NULL,
 `test_hash` VARCHAR(50) NOT NULL,
 `test_result` VARCHAR(50) NOT NULL,
 PRIMARY KEY (`test_id`),
	UNIQUE(test_hash)
 );


CREATE TABLE `fp_mobile_dtls`
(
`mobile_id` INT(11) NOT NULL
AUTO_INCREMENT, `mobile_name` VARCHAR(50)
NOT NULL, `test_hash` VARCHAR(50) , PRIMARY
KEY (`mobile_id`) ,
UNIQUE(mobile_name) ,
CONSTRAINT `fp_test_hash` FOREIGN KEY (`test_hash`) REFERENCES `fp_test_dtls`(`test_hash`)
)
;



CREATE TABLE `fp_device_log`
(
 `log_id` INT(11) NOT NULL AUTO_INCREMENT,
 `mobile_hash` VARCHAR(50) NOT NULL,
 `neighbour_hash` VARCHAR(50) NOT NULL,
 `date` INT(11) NOT NULL,
 `duration` INT(11) NOT NULL,
 PRIMARY KEY (`log_id`) ,
 CONSTRAINT `fp_mobile_hash` FOREIGN KEY (`mobile_hash`)
REFERENCES `fp_mobile_dtls`(`mobile_name`),
 CONSTRAINT `fp_neighbour_hash` FOREIGN KEY (`neighbour_hash`) REFERENCES
`fp_mobile_dtls`(`mobile_name`)
 );
```

SQL Queries:

1) Checking if the mobile_hash exists in fp_mobile_dtls table,
   `"select * from fp_mobile_dtls where mobile_name="+initiator`

2) Inserting mobile_hash in fp_mobile_dtls table,
   `"INSERT INTO fp_mobile_dtls(mobile_name) values ("+initiator+");"`

3) Inserting contact information in fp_device_log,
   ```
   "INSERT INTO fp_device_log(mobile_hash,neighbour_hash,date,duration)
   values(?,?,?,?)"
   ```

- statement.setString(1,initiator);
- statement.setString(2, hash);
- statement.setString(3, date);
- statement.setString(4, duration);

4) Storing test_hash from positive method
   "INSERT INTO fp_mobile_dtls(mobile_name,test_hash) values(?,?)"
   - statement.setString(1,initiator);
   - statement.setString(2, test_hash);

5) Fetch valules from Database, for find gatherings method, for given date
   "SELECT mobile_hash, GROUP_CONCAT(CONCAT(neighbour_hash,"=",duration))AS neighbour_hash FROM fp_device_log WHERE date="+date+" GROUP BY mobile_hash;"

MILESTONE 4 UPDATE

IMPLEMENTATION PLAN

1) Mobile class – Functionalities Completed, need to add input validations.
2) Government class – positiveTest() pending other functionalities completed
3) Database creation – completed
4) Test plan – pending
5) Internal and External documentation – pending
6) Exception jUnit, Overall complexity reduce – pending

Overall functionalities completed. Individual and end to end testing and method restructuring pending.