COVID 19 TRACKER

Assumptions:

- When any mobileDevice syncs with government database with positive COVID test; that test HashValue should already be present in database which will be done by government object. If not; then a message will be displayed showing that test HashValue is not yet in sync
- The value entered by user in mobileDevice and govtConfig files are key=value pairs with following format:
    - For mobileDevice:
        - address=127.0.0.1
          deviceName=Dhruv1
    - For governmentConfig file:
        - dbname=jdbc:mysql://db.cs.dal.ca:3306/dhruvp
          user=dhruvp
          password=B00868931
- The user has already configured following tables in government database (Please see the SQL script to create this tables)
    - TEST_RESULTS
    - CONTACT_LIST
    - POSITIVE_COVID_LIST

Design:

The main part of design is where we generate xml file to store contact and testHash of individual.

The format of xml is as follows:

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<covid_summary>
        <contact_list>
                <contact>
                        <contact_name>SHA value</contact_name>
                        <contact_date>50</contact_date>
                        <contact_duration>5</contact_duration>
                </contact>
                <contact>
                        <contact_name>SHA value </contact_name>
                        <contact_date>49</contact_date>
                        <contact_duration>9</contact_duration>
                </contact>
        </contact_list>
        <initiator_info>
                <initiator_name>SHA key</initiator_name>
                <initiator_testHash>mb1</initiator_testHash>
        </initiator_info>
</covid_summary>
```

Description of xml:

Covid_summary: The root tag
contact_list : This tag contains list of contacts for initiator
contact: This tag will contain information such as contact name, date and duration
initiator_info: This tag contains name and any test hash key for the user
initiator_name: This tag contains name of initiator
initiator_testHash: This tag contains any positive COVID-19 testHash given by user

Database Design:

There exists three table in our government database through which out solution is made.
The description of tables are:

The main table has the columns as **Field, Type, Null, Key, Default** for every table mentioned below.

1) CONTACT_LIST

person1_key	varchar(1000)	NO	PRI	
person2_key	varchar(1000)	NO	PRI	
contact_date	int(11)	NO	PRI	
contact_duration	int(11)	YES		
person1_contact_reported	tinyint(1)	YES		0
person2_contact_reported	tinyint(1)	YES		0

Description:

This table will contain contact between two devices.

Person1_key: SHA device value of person1
Person2_key: SHA device value of person2
Contact_date: Number of days since Jan 01, 2021
Contact_duration: Integer in minutes
Person1_contact_reported: Boolean value to signify that person1 contact person2 has been reported.
Person2_contact_reported: Boolean value to signify that person2 contact person1 has been reported.

SQL Query:

```
create table CONTACT_LIST(
        person1_key varchar(1000),
        person2_key varchar(1000),
        contact_date int,
        contact_duration int,
        person1_contact_reported boolean default 0,
        person2_contact_reported boolean default 0);

alter table CONTACT_LIST add primary key (person1_key,person2_key,contact_date);
```

2) TEST_RESULTS:

test_list	varchar(1000)	NO	PRI
test_date	int(11)	YES	
test_result	tinyint(1)	YES	

Description:

This table will store test synchronised with government directly.

Test_list: COVID 19 test ID

Test_date: Number of days since Jan 01,2021

Test_result: True is positive; false otherwise

SQL Query:

```
create table TEST_RESULTS(
        test_list varchar(1000),
        test_date int,
        test_result boolean);
alter table TEST_RESULTS add primary key (test_list);
```

3) POSITIVE_COVID_LIST

person_key	varchar(1000)	NO	PRI
test_key	varchar(1000)	NO	PRI
test_date	int(11)	YES	

Description:

This table will contain only positive cases identifying person associated with test and date the test was taken

Person_key: SHA Key for each device/individual

Test_key: COVID 19 test ID

Test_date: number of days since Jan 01,2021 at which the test was taken

SQL Query:

```
create table POSITIVE_COVID_LIST(
        person_key varchar(1000),
        test_key varchar(1000),
        test_date int);
alter table POSITIVE_COVID_LIST add primary key (person_key,test_key);
```

Design of methods:

We are going to discuss each method and see how we generate and pass above xml string to be stored in government database:

Methods in mobileDevice:

1) configData: This method will return SHA 256 hash key of input string
2) rootTags: This method will start xml tags covid_summary, initiator_info and also set initiator_name
3) getConfig: Getter for config string
4) recordContact: This method records contact as in xml tag <contact> will be made and data inside that will be populated here
5) positiveTest: This method will initialize and populate < initiator_testHash >
6) synchronizeData : This method will pass the xml data to government and also clean xml after it is stored in database

Methods in Government:

1) mobileContact:
    a. We first check if initiator_testHash is present;
    b. if yes we fetch that and if the value is synced by government and if the test is not duplicate; then we store in government table TEST_RESULTS.
    c. Further, for each <contact> tag, we store it in CONTACT_LIST.
    d. If contacts are duplicated we add the duration.
    e. We then join CONTACT_LIST and POSITIVE_COVID_LIST also checking for 14 days difference between contact date and test date.
    f. If we find any such data and if it was not previously reported we increment count
    g. We also update person1_contact_reported or person2_contact_reported which tells us that this contact has been reported
    h. If count>0 then we return true
2) checkNewTestHash:

    This method checks in POSITIVE_COVID_LIST if the testKey is new or old.
3) checkTestHash:

    This method checks in TEST_RESULTS if testKey is present
4) insertPositiveData:

    This method stores values in POSITIVE_COVID_LIST
5) recordTestResult:

    This method stores values in TEST_RESULTS
6) findGatherings:
    a. Find all contacts on passed date from CONTACT_LIST table
    b. Make contactSet of list having each contact from above query
    c. For each such entry from (a.) make adjacency map containing the individuals contacted for each individual in each entry
    d. Find intersection of the individuals with help of adjacencyMap
    e. For each intersected individual; suppose the tuple is (A,B) for which we find the intersection as list containing C,D. Add following tuple to contact_TempSet (A,C),(B,C),(A,D),(B,D)
    f. Add all such tuples to person Set
    g. Check if size of person is greater than minSize; then it can be called as gathering otherwise it is ignored
    h. If the value is greater; we go to find the contact for atleast minTime

i. If the resultSet generated above has both the contact present in person then we increment count
j. Calculate density as c/m
k. If we find the calulate density above the passed density then we increment gatherCount
l. Delete this tuple so that it wont be considered next time
m. Return gatherCount

Testing

Testing was automatically without any manual intervention

Take a look in **MainUnitTests.java**

The following methods and test cases are considered and **all of them are tested for all test cases mentioned** in test case scenarios file

- constructorTest
  - governmentTest
  - mobileDeviceTest
- RecordContact
  - Input validation
  - Boundary
  - Control Flow
- PositiveTest
  - Input validation
  - Boundary
  - Control Flow
- synchronizeData
  - Boundary
  - Control Flow
- RecordTestResults
  - Input
  - Boundary
  - Control Flow
- findGatherings
  - Input
  - Boundary
  - Control Flow

The list of methods that did not need testing:

In MobileDevice:

- getSha256
- bytesToHex
- removeChilds
- configData
- rootTags
- getStringFromDocument

In government:

- checkIfExists
- checkNewTestHash
- checkTestHash
- insertPositiveTestData
- getCharacterDataFromElement
- intersection

The above methods are used to generate results for findGatherings and synchronizeData; if our output is right then we can safely assume that we this methods are working correctly.

Focus of testing: synchronizeData and findGatherings were the two methods to focus our testing on as they are the main functionalities.

How test cases were decided: Test cases were decided on the basis of live application. Considering scenarios which complement live environment.
For example; A device coming in contact multiple times to another device subsequently. A device coming in contact with itself. etc

Important Points on UnitTests:

1) You will need to replace the govPath and mobPath variables. Please see comments to replace them.
2) Replace them with your location of where the file is kept.
3) Run 57 tests and see where the test passes and fails.
4) Ideal case should be all 57 test must pass.
5) These unit test cover every test case mentioned in test case
6) The successful output defines the integrity of the unit test

Description of each test suites in UnitTests:

- [root]
- Test suites
  - Ability to find gatherings at particular date
    - Control flow
      - Call findGatherings before recording any contact
      - Change findGatherings value by recording new contact
      - Normal operation

    - Boundary cases
        - minTime=0 in findGatherings
        - Date=0 in findGatherings
        - density=1 in findGatherings
        - density=0 in findGatherings
        - minSize=0 in findGatherings
    - Input Validation
        - Negative date in findGatherings
        - Density not between 0 and 1 in findGatherings
        - Negative minTime in findGatherings
        - Negative Density in findGatherings
        - Negative Minimum Size in findGatherings
- Store covid test result in database
    - Control flow
        - Pass duplicate testHash key again
    - Boundary cases
        - Date=0
        - Pass long TestHash
        - Pass TestHash with length=1
    - Input Validation
        - Negative date in RecordTestResult
        - Empty string in RecordTestResult
        - Null string in RecordTestResult
        - Only spaces string in RecordTestResult
- Store data to government database and inform if user has come into contact with someone who tested COVID-positive
    - Control flow
        - Multiple sync without inserting new data between subsequent sync
        - Change sync result from false to true by inserting positive contact
        - Call syncData() before recording contact or before any positive test
        - Call syncData() on the individual tested positive
        - Multiple sync with inserting new data
    - Boundary cases
        - Absolute Difference between contact date and test date is 14
        - Absolute Difference between contact date and test date is less than 14
        - Absolute Difference between contact date and test date is more than 14
        - Absolute Difference between contact date and test date is 0
- Records positive testHash to Mobile Device
    - Control flow
        - Normal operation
        - Pass duplicate TestHash
        - Government does not have Test hash OR Test Hash does not match with govt test hash
    - Boundary cases

        - Store long string
        - Only one letter/one digit string
    - Input Validation
        - Null String in positiveTest
        - Only spaces string
        - Empty String in positiveTest
- Ability to record contact
    - Control flow
        - Normal operation
        - Duplicate contact. We will add the duration in this case and not duplicate it again
        - Contact with itself
    - Boundary cases
        - Date=0
        - Pass only space string
        - Duration=0
    - Input Validation
        - Null individual string
        - Empty individual string
        - Invalid Date passed (Negative Date)
        - Invalid Duration(Negative Duration)
- Constructor testing : Government and MobileDevice
    - Mobile Device Constructor Testing
        - Invalid path passed in mobile Device config file
        - Passing null instead of government object
        - Valid mobileDevice constructor
        - Passing configFile with .properties extension
    - Government Constructor Testing
        - Invalid Credentials passed in govtConfig File
        - Invalid path passed in govtConfig File
        - Passing file path with .properties extension at the end
        - Valid Government constructor
        - Invalid Government Database path passed in govtConfig File