Simulation of a Covid-19 Contact Tracing Application

The paths of the config files for both the government class and the MobileDevice class are passed in their parameters.

The main method is invoked to create a object of government class and then the mobile device objects are called passing the government object in its parameter.

Example:

```
Government g = new Government("config_fileforGovernment.txt");
    MobileDevice A = new MobileDevice("MobileDevice1.txt", g);
    MobileDevice B = new MobileDevice("MobileDevice2.txt", g);
```

Project Structure

The project consists of two main classes – Government class and MobileDevice class.

MobileDevice Class

Global DataStructures:

ArrayList positivetest: `ArrayList<String> positivetest`, Stores all the positive test hashes for the mobile device user.

Map userdevicedetails: `Map<String, String> userdevicedetails`, Stores the address and the devicename of the mobile device user.

Map contacts: `Map<String, ArrayList<Integer>> contacts`, Stores the contacted device hash as the key and the value as the date and duration at which contact was established in the arraylist of integers.

Methods:

1. *MobileDevice (String configFile, Government contactTracer)*
   Constructor for the class. It accepts a configuration file path for a file that contains the device's network address (string) and device name (string).
   The configuration file will have one line for each configuration value.
   The format of a line is <keyword>=<value> where keyword is either address or devicename.
   contactTracer is the object that represents the government and that I am using to record my results to the governments database.

2. *Private boolean read (String Configfile)*

Reads the data from the mobile devices Configuration file from Configfile location for that file and stores the information in the 'userdevicedetails' Map using object of bufferedReader class. Returns true if the file location was successfully read and false if the file was not found or any other input validation occurred.

3. *recordContact(String individual, int date, int duration)*
   When the users device detects another device, that driver calls recordContact to locally record that <u>individual</u> (an alphanumeric string) was near us on <u>date</u> for <u>duration</u> minutes in the Map 'contacts'. The date is the number of days since January 1, 2021. These contacts do not go to the government system right away.
   They are stored in MobileDevice until the time comes to send the bulk of contacts to the government i.e. when the synchronizeData method is called by the mobile device user. This method checks for basic input validation like when the individual is null or empty or if the date is <0 or if the duration is <=0, then return.
   If the individuals device had a contact with the mobile device owner earlier on the same day then their duration times are added and only ONE record is stored in contacts.

4. *Private String hash(String deviceaddress)*
   Converts the deviceaddress that is passed in the parameter into a hash using the SHA-256 hash function from the MessageDigest class in java.security.

5. *positiveTest(String testHash)*
   The mobile device user will call positiveTest when the user asks to tell the system that they have tested positive for COVID-19. The testing agency will provide an alphanumeric string that identifies the test; the user will record that string to link those results to the individual who normally uses the mobile device. The positiveTest hash corresponding to the device hash will be stored locally in 'positivetest' Map until the user calls synchronizeData method. Basic input validation like testHash null or empty is checked before proceeding.

6. *boolean synchronizeData()*
   Checks if the private read method reads the data from the config file correctly. If not returns false else continues. This method periodically triggers an exchange with the government database whenever invoked by the mobile device user. It will package all of the information in the mobile device into a formatted XML string and send that information to the government through the Government object's mobileContact method. The information packaged up includes the device's hash, any contacts with the time and duration, and positive test hashes reported by the owner of the phone (as user calls to positiveTest).
   I am using object of DocumentBuilder to store all the data in an XML string from the 'userdetails', 'contacts' and 'positivetest' datastructures. Also using object of Transformer class to create a formatted XML structure and then streaming the result into a XML string. This xmlString alone with the Hash of the mobiledevice user i.e. the initiator of synchornizeData is send the contactTracer object of the government class.
   XML schema I am creating has the Document Type Definition (DTD) format as follows.

```
<!ELEMENT contacts_summary (positivehash, contact*)>
<!ELEMENT positivehash (testhash*)>
<!ELEMENT contact(individual, date, time)>
<!ELEMENT testhash (\#PCDATA)>
<!ELEMENT individual (\#PCDATA)>
<!ELEMENT date (\#PCDATA)>
<!ELEMENT time(\#PCDATA)>
```

This method gets back from the government an indicator of whether or not this mobile device has been near anyone diagnosed with COVID-19 in the last 14 days. That outcome is the return value of synchronizeData: true if the device has been near someone with covid-19 and false otherwise.
The government object, will then parse the string from the mobile device and store or retrieve data from the database.

Government Class

Global DataStructures:

ArrayList logindetails:`ArrayList<String> logindetails`, stores the logindetails for the government database, i.e. the database name, user name and the password for the database connection. It gets this information from the config file that is passed in the government constructor parameter.

Methods:

1. *Government(String configFile)*
   Constructor for the class. It accepts a configuration file location that contains the file which contains the domain name of the database for the government (string), the username to access the database (string) and the password to access the database (string). The configuration file will have one line for each configuration value. The format of a line is <keyword>=<value> where keyword is either database, user or password.

2. *Private Boolean Connection()*
   Reads data from the configFile using objects of File and BufferedReader class and stores the login details inside ‘logindetails’ arraylist. Connection creates a new instance of the JDBC driver. Returns false if any connection errors to the database occur and returns true on successful connection to the database. Also creates a connection of the SQL statement to the database connection.

3. *boolean mobileContact(String initiator, String contactInfo)*
   Initiator has the combined hash of the the device network address and the device name. contactinfo has the xmlString that contains the data passed from MobileDevice class method SynchronizeData(). mobileContact method is called by the mobile device *synchronizeData* method to send the contact information to the government. The caller is identified in the system by initiator. The purpose of hashing the device address and name is to allow a device
   to be uniquely identified without revealing any information about the user. For a cryptographically secure hash value, I am using the SHA-256 hash function from the MessageDigest class.
   All the contact information is contained in the contactInfo string in the form of a XML document which is then streamed into a xml string.
   The method returns whether or not the contacting mobile device has been near someone who tested positive for COVID-19 within 14 days of their contact. Only new COVID-19 contacts are reported back. These contacts must be stored in the government database.
   I am first checking if the connection is established or not using the connection method. If the connection is not established, then it returns false.

Then using SQL I execute the query to use the desired database. Then using objects of DocumentBuilderFactory, DocumentBuilder and InputSource to read the data from the XML string tag by tag.
I am first storing data from the positivehash tag, IF IT EXISTS, in the Government databases positiveResult table.

Then I am storing data from the contacts tag, IF IT EXISTS, in the Government databases contacts table. I am also checking if the database already contains the same contact on the same day. If it does then the database contact is updated with the total duration. Such a situation will only arise if the mobiledevice user has called synchronize before meeting the individual again on the same day and then calls synchronize again afterwards and keep storing the contacted userhash in the 'contacts'.

Then I traverse 'contacts' and check If the contacted user exists in the form of a DeviceHash. This is to check if the user has already been send a notification of it meeting a covid positive contact so that if the user calls synchronize again, then it is not notified of its contact again in the form of <u>true</u> or <u>false</u>.

4. *recordTestResult(String testHash, int date, boolean result)*
   Record in the database that a test, identified by the alphanumeric string testHash, had a collection taken up on date (the number of days since January 1, 2021), and came out positive (result = true) or negative (result = false). Stores this data into the Government database table Testing using SQL query.
   Checks if the connection is established or not before recording information. It also checks for basic input validation like testHash null or empty, if the date is less that 0 or if result is anything other than true or false.

5. *Void resetTables()*
   This is a public function as it is directly handing data from the governments database. This is a utility function that can be used to clear the tables in the database of its data completely. It returns the date on which the tables were cleared.

6. *int findGatherings(int date, int minSize, int minTime, float density)*
   Before proceeding I am checking if the connection is established or not. If it is not then -1 is returned. Then I check of some input validations like if date is less than 0 or minTime is less than 1 or if density is less than 0, if yes then return 0.
   Looking at the contacts reported on *<u>date</u>*, I return the number of large gatherings on that date. Again the <u>date</u> is the number of days since January 1, 2021. For this I filter out all the contacts from the database occurred on date and store them in an ArrayList of ArrayList 'pairs' with the duration for which they were met. I also store the individual with all the people it has met in another arraylist of ArrayList 'pairsonly'.

   To detect a gathering, consider all pairs of individuals A and B. Find the set S of people that both A and B contacted on that day (including A and B). I am only considering S as a gathering if it contains at least *<u>minSize</u>* individuals. For this I store the individuals that A contacted in a Set and individuals that B contacted in another Set. Then I find their intersection **X**. This **X** is a potential gathering.

   To determine whether the gathering X is large, I count the number of pairs of individuals c within X who contacted one another on the given date for at least *<u>minTime</u>* minutes. If **X** contains n individuals then there can be at most m = n(n*1)2 possible pairs. I then compare c with this maximum: if c=m is greater than density then the gathering X is deemed large and worth reporting. S contacts only sets of unique gatherings for that I check if S if a potential gathering has already been handled.

7. *private static String getCharacterDataFromElement(Element e)*

This is a private static method that is just used to convert the read data from the xml string fromxml element form into String form. The usage of this method is limited to the *mobileContact* method.

Note: Control flow testing was necessary to ensure that invalid test results are omitted.

Calling synchronize again and again should return true the first time if there was a contact with a covid positive patient in the 14 days of the test result irrespective of whether you were positive or not. But for the subsequent times false should be returned.

Calling getGathering on the same data again and again produced erroneous results. So data should be cleared the second time getGathering is called.

Connection should be established. If the connection is not established nullpointer is returned.

If a data is added to a table where the primary key already exists, then the sql exception iscalled to return a localized message.

In the positiveResult table, no positive testhashes cannot be same.

In the Testing table, testHashes should be unique. Test hash of positiveResult references thetesthash of testing table. positiveResult data is used to map the user devicehash to the covidtesthash data.

A person can meet the same person on different dates. Always new contacts are reported.