------------------------------------------------------------------------------------------------------------

# CSCI 3901 – Method distributions, Additional Test cases and assumption and references.

**Objective :** The course project is the opportunity to demonstrate all the concepts from the course in one body of work.

**Report and Code by:**
● Shivam Bhojani (B00895637) - shivam.bhojani@dal.ca

## Methods Distribution across the class files.

**ManagePersonRecords.java**

1. PersonIdentity addPerson(String name)
2. Boolean recordAttributes(PersonIdentity person, Map<String, String> attributes)
3. Boolean recordReference(PersonIdentity person, String reference)
4. Boolean recordNote(PersonIdentity person, String note)
5. Boolean recordChild(PersonIdentity parent, PersonIdentity child)
6. Boolean recordPartnering(PersonIdentity partner1, PersonIdentity partner2)
7. Boolean recordDissolution(PersonIdentity partner1, PersonIdentity partner2)

**ManageMediaArchieve.java**

1. FileIdentifier addMediaFile(String fileLocationwithName)
2. Boolean recordMediaAttributes(FileIdentifier fileIdentifier, Map<String, String> attributes)
3. Boolean tagMedia(FileIdentifier fileIdentifier, String tag)
4. Boolean peopleInMedia(FileIdentifier fileIdentifier, List<PersonIdentity> people)

**Genealogy.java**

1. PersonIdentity findPerson(String name)
2. String findName(PersonIdentity id)
3. FileIdentifier findMediaFile(String filelocationwithName)
4. String findMediaFile(FileIdentifier fileId)
5. Set<FileIdentifier> findMediaByTag(String tag, String startDate, String endDate)
6. Set<FileIdentifier> findMediaByLocation(String location, String startDate, String endDate)
7. List<FileIdentifier> findIndividualsMedia(Set<PersonIdentity> people, String startDate, String endDate)
8. List<String> notesAndReferences(PersonIdentity person)

9. List<FileIdentifier> findBiologicalFamilyMedia(PersonIdentity person)
10. Set<PersonIdentity> descendents(PersonIdentity person, Integer generations)
11. Set<PersonIdentity> ancestores(PersonIdentity person, Integer generations)
12. BiologicalRelation findRelation(PersonIdentity person1, PersonIdentity person2)

## Step by Step implementation logic for each method and their assumptions.

**PersonIdentity addPerson(String Name)**

- If name == null or empty > Return Null
- Enters new row in person database and returns a Person object with Id and name in that object variables
- In case of failure, console gives error message and returns null

**Boolean recordAttributes (PersonIdentity, Map)**

- Checks if Person null.
- Checks if Map is empty or null.
- Date formatting is done by class called dateFormat. Whenever an attribute is there with attribute name as "birthDate" or "deathDate" it will call the dateFormat Class
- Formatting dates likes this will be handles by the code
    - YYYY-MM-DD
    - YYYY/MM/DD
    - YYYY\MM\DD
    - YYYY
    - YYYY-MM
    - YYYY\MM
    - YYYY/MM
- Checks if person is there in database – If not then returns false and gives user facing message
- Able to store different types of attributes.
- Return true if connection and update was successful
- Returns false incase of exception found wrt to SQL
- In case of failure, console gives error message.

**Boolean RecordReference (personIdentity, String reference)**

**Class – ManagePersonRecords**

- Checks if Person object is null
- Checks if Reference value is null, empty

- Checks if given person exists in database or not – if not, then returns false and gives user facing message.
- Inserts data in person_references table.
- Return true at the end.
- It will check for duplicate entries where a reference already exists for particular personID and it will return true without entering the data.

**Boolean Recordnote (personIdentity, String note)**

- Checks if Person object is null.
- Checks if Reference is null, empty
- Checks if given person exists in database or not – if not, then returns false and gives user facing message.
- Inserts note in person_notes table
- Return true at the end.
- It will check for a record with given person Id and note. If record exists then it will return true without inserting new data.

**Boolean recordChild (person object 1, person object 2)**

- If person1 or person2 is null, return false
- If both people are same, then it will return false
- Checks of parent and child is there in person database, if not- then returns false
- It will allow more than one parent. In other way, it will allow social relations
- If record already there for same parent child
- Gives error message when current child is recorded as parent of current parent.

**Boolear recordPartnering (person 1, person 2)**

- If person1 or person2 is null, return false
- If both are same, it will return false.
- It will allow duplicate entry because it may be the case that partnering – dissolution - partnering
- Allows recording single person with others

**Boolear recordDissolution (person 1, person 2)**

- If person1 or person2 is null, return false
- If both are same, it will return false.
- There can be dissolution only if they are partners
- It will allow dissolution because it may be the case that dissolution – partnering – dissolution
- Allows recording single person with others.

**FileIdentifier addMediaFile (String filelocation)**

- If filename == null or empty  > Return Null
- 
- Check if the file with same filename exists - If exists then return null with message
- In case of SQL error, it gives error with proper message
- If file does not exsist then it will insert it in database
- One file is added, the method will return an object with filename and ID in it.

**Boolean recordMediaAttributes( FileIdentifier fileIdentifier, Map<String, String> attributes )**

- File object null – map empty – map null
- Check if media file is there in database or not- If not then return false and gives user facing message.
- Date formatting is done by class called dateFormat. Whenever an attribute is there with attribute name as “date” it will call the dateFormat Class
- Formatting dates likes this will be handles by the code
  - YYYY-MM-DD
  - YYYY/MM/DD
  - YYYY\MM\DD
  - YYYY
  - YYYY-MM
  - YYYY\MM
  - YYYY/MM
- Return true if connection and update was successful
- Return false in case of SQL exception and gives user facing message.

**Boolean peopleInMedia( FileIdentifier fileIdentifier, List<PersonIdentity> people )**

- file null – list empty – list null
- Check if given medial file exists – if not then return false with user facing message.
- Check the given set of people are there in db or not
- Filter out the ones which are there and put it in the list.
- One by One insert record for media to people in people_in_media table
  - Also checking if the record already exists for that mediaId and peopleId

**Boolean tagMedia( FileIdentifier, fileIdentifier, String tag )**

- File object null – tag empty – tag null
- Checks of media file exists in database or not – if not, it returns false with user facing message
- **Validation to check for duplicate tags for same media file**
- Insert tags with media Id in media_tags table
- Incase of SQL exception, it gives user facing message

**REPORTING**

**PersonIdentity findPerson( String name )**

- Checks if name is empty or null.
- In case of multiple people with same name, then it will return custom exception.
- Runs select query in person table based on the given name.
- If found, then it returns person object with all the attrbutes
- If person not found, then it returns null object with user facing data.
- It returns User facing message incase of SQL Exception

**FileIdentifier findMediaFile( String name )**

- Empty or null name
- Runs select query in media_archieve table based on the given name.
- If found, then it returns file object with all the attrbutes
- If file not found, then it returns null object with user facing data.
- It returns User facing message incase of SQL Exception

**String findName( PersonIdentity id )**

- Null person object.
- For the given object, it checks the person table with the id in the object and gives back the Person name
- Returns null if record not found.

**String findMediaFile( FileIdentifier fileId )**

- Null file object – return null
- For given object, checks the SQL with given ID and gives name from Sqtable
- Returns null if record not found.
-

**BiologicalRelation findRelation( PersonIdentity person1, PesonIdentity person2 )**

- If person1 and person are same – return null with user facing message
- Check person1 and person2 for null values – return null if any of the person object is null
- Check person1 and person exists in db – return null if any one of the person is not in DB
- Fin ancestors of person1
- Find ancestors of person2
- Compare the ancestors list and find first common one
- If common found then cound cousinship and removal. (it will also check each other as ancestor)
- If not common, then return null

**Set descendents( PersonIdentity person, Integer generations )**

- Validate person for null value

- Validate generations for less than or equal to 0 value.
- Find decendants for the given person.
- If no decendants found, then return **empty hashset**

**Set ancestors( PersonIdentity person, Integer generations )**

- Validate person for null value
- Validate generations for less than or equal to 0 value.
- Find ancestors for the given person.
- If no ancestors found, then return **empty hashset**

**List notesAndReferences( PersonIdentity person )**

- Validate person for null value
- Validate person with database
- Find notes and put it in single list
- Find reference and put it in single list
- Return empty list incase of no notes or references found.

**Set findMediaByTag( String tag , String startDate, String endDate)**

- Check tag for null and empty value
- Find media id based on tags
- If no media Id found then return empty hashset
- If media Id found then return objects of media file
- Date formatting is done by class called dateFormat.
- Formatting startDate and endDate like this will be handles by the code
    - YYYY-MM-DD
    - YYYY/MM/DD
    - YYYY\MM\DD
    - YYYY
    - YYYY-MM
    - YYYY\MM
    - YYYY/MM
- Check if dates are null – If null then don’t consider it in SQL

**Set findMediaByLocation( String location, String startDate, String endDate)**

- Check tag for null and empty value
- Find media id based on location
- If no media Id found then return empty hashset
- If media Id found then return objects of media file
- Formatting startDate and endDate like this will be handles by the code
    - YYYY-MM-DD

    - YYYY/MM/DD
    - YYYY\MM\DD
    - YYYY
    - YYYY-MM
    - YYYY\MM
    - YYYY/MM
- Check if dates are null – If null then don't consider it in SQL
- Check if dates are null – If null then don't consider it in SQL

**List findIndividualsMedia( Set people, String startDate, String endDate)**

- Check people is null or empty
- Check for null and empty start date
- Check for null and empty enddate
- Formatting startDate and endDate like this will be handles by the code
    - YYYY-MM-DD
    - YYYY/MM/DD
    - YYYY\MM\DD
    - YYYY
    - YYYY-MM
    - YYYY\MM
    - YYYY/MM
- Check if dates are null – If null then don't consider it in SQL
- If no media found, it will give empty list

**List findBiologicalFamilyMedia(PersonIdentity person)**

- Check if person is null
- Checks if person found in database or not
- Check if that person has any child relation in the records
- If no, then return null with user facing message
- if children found then it store it in set and call peopleinMedia method

## Constraits

- Not Tested with more than one user running the code connected with same DB at a time.
- When there are more than one person with same name, then the user will get exception.

## Assumption

- For finding media files by location, the code will be check across the attribute name which is "location" in media_attributes table
- For finding media files on specific dates, the code will for the attribute name which is "date" in media_archieve.

## Extra Test Cases:

**RecordAttributes - Person**

- Pass person object which is not known to database.
- Pass person object as null
- With empty Map
- With null value as Map

**RecordReference**

- Enter same reference for the same person multiple times.
- Pass person object which is not known to database.
- Pass person object as null

**recordNote**

- Enter same note for the same person multiple times.
- Pass person object which is not known to database.
- Pass person object as null

**recordChild**

- When parent and child object are same
- When there is already a parentChild relation recorded between given objects
- One of the person objects are not known to database.

**recordPartnering**

- When partnet1 and partner2 object are same
- When there is already a relation recorded between given objects
- One of the person objects are not known to database.

**recordDissolution**

- When partnet1 and partner2 object are same
- When there is already a relation recorded between given objects
- One of the person objects are not known to database.

**addMediaFile**

- add filename which is already recorded earlier
- add filename which is unique

**recordMediaAttributes**

- Pass file object which is not known to database.
- Pass file object as null
- With empty Map

- With null value as Map

**tagMedia**

- Enter same tags for the same media file
- With file object not known to database
- When file object is null

**peopleInMedia**

- With file object not known to database
- When file object is null
- when few of the person in list are known to database
- when all the person in list is known to database
- when none of the given person in list are known to database.

**Descendents**

- When generation value is negative
- When generation value is positive
- When generation value is 0
- When person object is not known to database
- When person object is null

**ancestors**

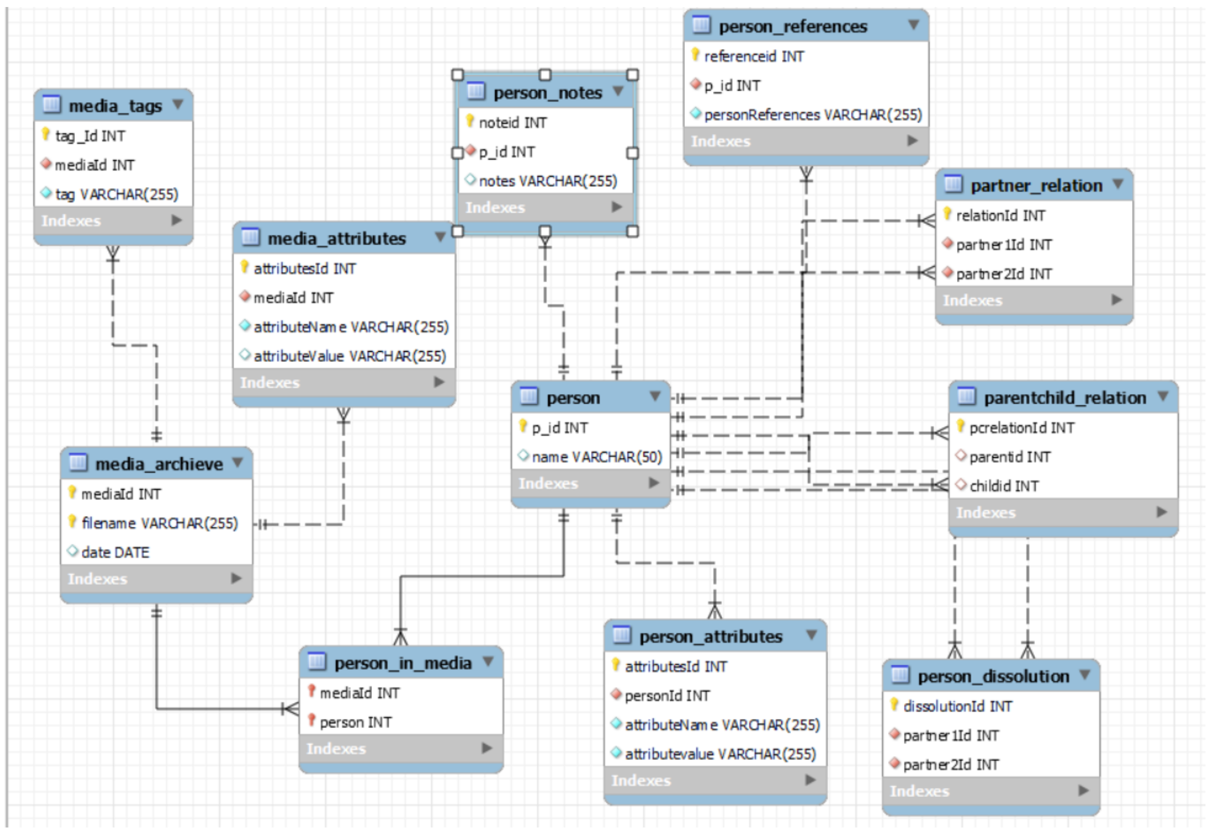
- When generation value is negative
- When generation value is positive
- When generation value is 0
- When person object is not known to database
- When person object is null

**findRelation**

- When person object is not known to database
- When one is ancestor of other
- When there is no common ancestor
- When both person are having more than one common ancestor

## ERD

<ERD file and sql script are added in repository>



## References:

| | |
|---|---|
| [1] | Getting all descendants of a parent, “Getting all descendants of a parent,” *Database Administrators Stack Exchange*, 11-Mar-2015. [Online]. Available: https://dba.stackexchange.com/questions/94932/getting-all-descendants-of-a-parent. [Accessed: 14-Dec-2021] |
| [2] | jay, “Recursive query in MySql to Determine Inheritance type,” *Stack Overflow*, 11-Oct-2020. [Online]. Available: https://stackoverflow.com/questions/64309880/recursive-query-in-mysql-to-determine-inheritance-type. [Accessed: 14-Dec-2021] |
| [3] | “SimpleDateFormat (Java Platform SE 7 ),” *Oracle.com*, 24-Jun-2020. [Online]. Available:https://docs.oracle.com/javase/7/docs/api/java/text/SimpleDateFormat.html. [Accessed: 14-Dec-2021] |