CSCI 3901 Final Project

Name: Dhruvrajsinh Omkarsinh Vansia

Banner ID: B00891415

Overview:

This project will link individual information with one's biological family relation and also with ones' pictures. Individual information includes, their name, date and location of birth, date and location of death, gender, occupation, references to source material, notes on the individual. To link individual with the biological family relation, the system will create HashMap (key: parent id, value: child id), each id represents individual. Besides from parents/child relations, the system also includes partnering ceremony relations and partnering dissolutions. For the pictures in the archive, the system will store file name of the picture, date when picture was taken, location, tags, and name of the individual seen in picture. In individual's and picture's information, some information may be absent apart from name of the individual and name of the picture. The system will return following information from the information mentioned above:

1. Relationship between two individuals,
2. Notes and other material related to individuals,
3. Lists of descendants and ancestors of the individuals for n generations,
4. Lists of pictures of particular place/tag, immediate family members and for the set of people, within some time range.

Files and external data:

The program uses following files and external data.

1. BiologicalRelation.java: the file contains findRelation, getcounsinship and getRemoval method.
2. ConnectiontoDB.java: this class is used to makes connection with the MySQL database.
3. Constant.java: this interface stores constant variable values.
4. FamilyMap.java: the class creates HashMap to store parent and child information.
5. FileIdentifier.java: this class is used to store media information to the database.
6. Genealogy.java: all reporting methods are defined in the Genealogy class.
7. mainUI.java: this class contains main method of the program.
8. MediaInformation.java: this interface is defined to specify the behaviour of a FileIdentifier class.
9. PersonIdentity.java: this class is used to store person information to the database.
10. PersonInformation.java: this interface is defined to specify the behaviour of a PersonIdentity class.
11. ProjectTest.java: this file contains Junit tests for the program.
12. ShowTable.java: this class is used to display table data.
13. Project.sql: this file contains SQL queries to generate database for the program.
14. sampleData.sql: this file contains queries to insert data into the tables.

Assumption:

- In recordAttributes() and recordMediaAttributes(), the method returns true if any of the key's value will be added to the database. It will display the non-recorded attributes, too.
- If user only enter year as date, then program will store that value in yyyy-00-00 format. Similarly for year and month (yyyy-mm-00).

Data structures and their relation to each other:

1. Map<Integer, List<Integer>> familyMap: this data structure is used to store parent and child information. Here, key: parent ID and values: children's IDs.
2. Map<String, List<String>> possibleValues: this map stores all the possible values for the particular column values (for example: column name: dob, possible values: dateOfBirth, birthDate). Here, key: column value and values: list of possible alternatives.
3. List<Integer> childList and List<String> pValues: These two lists store child ID and possible values of the column names, respectively, and are used in familyMap and possibleValues.

Key Algorithms:

1. findRelation():
   This method finds biological relation between two persons. For that, it uses familyMap to retrieve information regarding parent and child. Firstly, it finds the parent/parents for the person one and person two and stores it in different list for later use. Now, from these parents ID, it finds their parents. This process continue till common parent will be found. In some cases, there are two parents in the familyMap. In such situation, it stores this information in temporary list. When it fails to find common parent and temporary list has some values, then it will search for common parent using available parent in temporary list. For termination condition, if the list size does not increase in three consecutive searches, then it will terminate the search process and give null object which means there is no biological relation. If it finds the common parents then it will count counsinship and removal from the position of the common parent in the list.
2. descendents() and ancestors():
   These two methods give person's descendents and ancestors. It uses childInfo table. It searches for child ID and parent ID respectively to find information. It uses two lists, permanent and temporary list. Temporary list is used to store particular generation people information and also to track the generation number. When generation number reaches to the generations given by the user, it will return permanent list that has all information regarding person's descendents and ancestors.
3. findBiologicalFamilyMedia():
   This method returns set of FileIdentifier objects of person immediate child/children. Firstly, it finds child information from the childInfo table and by

using findPerson() method, it creates set of PersonIdentity objects. This set will be used as parameter value for the findIndividualMedia(). By using this method, it will get set of FileIdentifier objects.

Test plans:

Black box testing:

Input Validation:

- addPerson()
    1. Null or empty string passed as name
- recordAttributes()
    1. Null or empty string passed as person
    2. Negative or empty value passed as date of birth or death
    3. Null or empty string passed for value of the gender attribute
    4. Null or empty string passed for value of the occupation attribute
- recordReference()
    1. Null or empty string passed as person or reference
- recordNote()
    1. Null or empty string passed as person or note
- recordChild()
    1. Null or empty string passed as person or child
- recordPartnering()
    1. Null or empty string passed as partner1 or partner2
- recordDissolution()
    1. Null or empty string passed as partner1 or partner2
- addMediaFile()
    1. Null or empty string passed as filelocation
- recordMediaAttribute()
    1. Null or empty value passed as fileIdentifier
    2. Null or empty string passed as year, city and date
- peopleInMedia()
    1. Null or empty value passed as fileIdentifier
    2. Null or empty list passed as people
- tagMedia()
    1. Null or empty value passed as fileIdentifier
    2. Null or empty string passed as tag
- findPerson() and FindMediaFile()
    1. Null or empty string passed as name
    1. Null or empty string passed as name
- descendents() and ancestors()
    1. Null or empty value passed as person

    2. Null passed as generation
    3. Negative integer passed as generations
- findIndividualsMedia ()
    1. Null or empty list passed as people
    2. Null or empty string passed as startDate or endDate

Boundary tests cases:
- addPerson()
    1. single character passed as name
- recordChild()
    1. single child added to parent
    2. multiple child added to parent
- peopleInMedia()
    1. Only one person in the list of people
- findName()
    1. Passed 1 as an id
- findIndividualMedia()
    1. Passed single person in the set of people

Control flow tests:
- addPerson()
    1. Add same name
- recordAttributes()
    1. Add two or more different Map for a person
- findPerson()
    1. finding a person that was not added to database
    2. Finding a person whose name is added multiple times in database.
- findRelation()
    1. Finding a relation when two persons are connected in biological relation
    2. Find a relation when two persons are in different generation
- descendents()
    1. Finding a descendents when there is none
- ancestors()
    1. Finding ancestor when there is no ancestor available.
- findMediaByTag()
    1. Finding a media when there is no available
    2. Finding a media when there is multiple available.
- findIndividualsMedia()
    1. Finding a media when there is no media available for the individual.
- findBiologicalFamilyMedia()
    1. Finding a media when there is no immediate children.
    2. Finding a media when there are multiple children.

Data Flow tests:
- recordPatnering()
    1. Adding a relation to person who is already in relation
- recordDissolution()

1. Adding a dissolution to a person who is not in relationship.

• findIndividualMediaByTag (), findndividualMediaByLocation(), findIndividualsMedia()

1. Finding a media whose dates fall within date period
2. Finding a media whose dates are not in date range

In addition to that, there is a file name ‘ProjectTest.java’ in which I have coded Junit tests to check the program output.

Database:

For creating database, there is a SQL file ‘project.sql.’

ER diagram of the database:

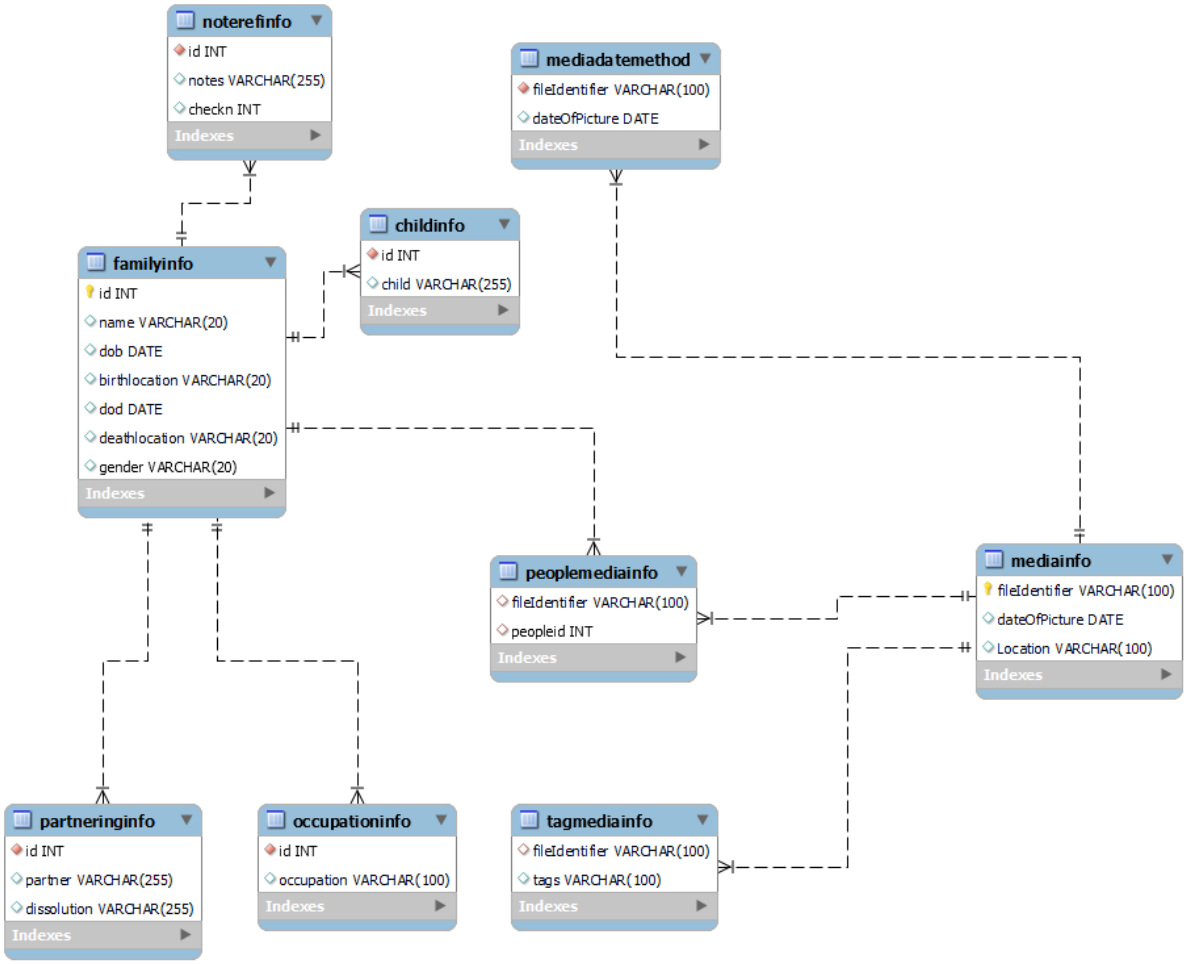

Table information:

1. familyinfo: This table is used to store person’s information such as, id, name, date of birth and location, date of death and location, gender.
2. occupationinfo: This table stores person’s occupation information.
3. childinfo: it stores person’s id and child id.
4. partneringinfo: it is used to store person’s partner and dissolution information.

5. noterefinfo: it stores note and reference information of the individual. If Boolean value is true (3rd column, checkn), there is a note in second column else reference information in the second column,
6. mediainfo: it stores media information such as file location, date of picture, location of the picture.
7. tagmediainfo: it stores tag for the media file.
8. peoplemediainfo: it stores information about people who are in the media.