Author: Shivangi Ajaykumar Bhatt
B00 # : B00863408
Subject Code: CSCI 3901
Date: Nov 25th, 2021
Project Milestone#: 3

Data Structure

The program will utilize the following three data structures:

1. List<PersonIdentity> listOfPeople: The list will store all the PersonIdentity objects for each individual so that they can be used easily to create a family Tree.
2. List<FileIdentifier> listOfMedia: This list would store all the media.
3. List<String> peopleAttributes: Stores the attribute names for people.
4. List<String> mediaAttributes: Stores the attribute names for media.
5. HashMap<PersonIdentity, List<PersonIdentity> FamilyTree: A tree is used to define the biological relationship between individuals.

Code design

The program uses the following tables in the database:

- Person: Stores the information about the person.
- Notes: Stores the notes about the person.
- References: Stores the references about the person.
- Descendants: Stores the parent-child relationship between two people.
- partnering: Stores the partnering between two people.
- dissolution: Stores the dissolution between two people.
- Media: Stores the information about media.
- People_in_media: Stores the people in media
- Tags: Stores the tag of the media.

The design of the code works as follows:

- Class People has the methods for managing the family tree are used to store the relevant information in the following tables: person, notes, references, relation.
- Class MediaArchive has the methods for managing the media are used to store the relevant information in the following tables: media, people_in_media, tags.
- The class PersonIdentity stores name and id of the person.
- The class FileIdentifier stores the filename and id of the media.
- The class BiologicalRelation will hold the degree of cousinShip and degree of Removal between two people.
- The class Family tree would create the family tree between the given individuals.

- The class Geneology would be used to report the output for methods to report information about people and media, including, finding relations between two people, finding descendants, finding ancestors, finding media for an individual, find media for a biological media.

Assumptions

Following assumptions have been made for the program:

- The date will always be entered in YYYY-MM-DD format.
- The partnering or dissolution between people that are biologically connected is not possible
- Recording Child is not possible if there is partnering or dissolution recorded for the given two people.
- For accessing methods for managing people and managing media, it is important to create instances of People and MediaArchive

Key algorithms

The classes and their algorithms are defined as follows:

1. **People**

   The algorithm of the seven methods of the class are explained as follows:

   - PersonIdentity addPerson(String name)
     - Insert a row in the table person.
     - Create the object PersonIdentity
     - Add it to the List<PersonIdentity> People
     - Return the object PersonIdentity

   - Boolean recordAttributes(PersonIdentity person, Map attributes)
     - Check if the person or attributes is not null. If they are, return false
     - Loop through the Map attributes:
       - Check if the key does not exist as column name in table person
       - If not, modify the table and add the column to the table.
       - End if
       - Update the row for PersonIdentity person
     - Return true if attributes have been added and false otherwise.

   - Boolean recordReference(PersonIdentity person, String reference)
     - Add the row in references table
     - Return true if successfully added and false otherwise.

- Boolean recordNote(PersonIdentity person, String note)
    - Add the row in notes table
    - Return true if successfully added and false otherwise.

- Boolean recordChild(PersonIdentity parent, PersonIdentity child)
    - Check if partnering is recorded for parent and child. If true, return false.
    - Check if dissolution is recorded for parent and child. If true, return false.
    - Check if there is a biological relationship between parent and child. If true, return false.
    - Add the row in the Descendents table.
    - Return true if successfully added and false otherwise.

- Boolean recordPartnering(PersonIdentity partner1, PersonIdentity partner2)
    - Check if partnering is recorded for parent and child. If true, return true.
    - Check if there is a biological relationship between parent and child. If true, return false.
    - Delete the dissolution record for partner1 and partner2.
    - Add the row in the partnering table.
    - Return true if successfully added and false otherwise.

- Boolean recordDissolution(PersonIdentity partner1, PersonIdentity partner2)
    - Check if partnering is recorded for parent and child. If true, return true.
    - Check if there is a biological relationship between parent and child. If true, return false.
    - Delete the dissolution record for partner1 and partner2.
    - Add the row in Relation table with relation as dissolution.
    - Return true if successfully added and false otherwise.

2. MediaArchive

The algorithm of the four methods of the class are explained as follows:

- FileIdentifier addMediaFile(String fileLocation)
    - Insert a row in the table media.
    - Create the object FileIdentifier
    - Return the object FileIdentifier

- Boolean recordMediaAttributes(FileIdentifier fileIdentifier, Map attributes)

- Check if the fileIdentifier is null, if true, return false
- If true, Loop through the Map attributes:
    - Check if the key does not exist as column name in table person
    - If not, modify the table and add the column to the table.
    - End if
    - Update the row for FileIdentifier.
    - Set the attributes of pers
- Return true if attributes have been added and false otherwise.

- Boolean peopleInMedia(FileIdentifier fileIdentifier, List people)
    - Get Id of fileIdentifier.
    - Loop through the list of people.
    - Get each person's id and insert a row for each people in people_in_media.
    - Return true if successfully added and false otherwise.

- Boolean tagMedia(FileIdentifier, fileIdentifier, String tag)
    - Add the row in tags table
    - Return true if successfully added and false otherwise.

3. Geneology

The algorithm of all methods of the class are explained as follows:

- PersonIdentity findPerson(String name)
    - Find the person from the person database with given name.
    - Return its object.

- FileIdentifier findMediaFile(String name)
    - Find the media from the media database with the given name.
    - Return its object.

- String findName(PersonIdentity id)
    - Return the name of the individual with given personIdentity object

- String findMediaFile(FileIdentifier fileId)
    - Return the name of the media for given FileIdentifier object.

- BiologicalRelation findRelation(PersonIdentity person1, PesonIdentity person2)
    - Check if person1 or person2 is null. If yes, return null
    - Create tree
    - get family tree

- Call findCommonAncestor() and store its returned value in String called distance.
- Get the distance from person1 and person2 by splitting the string.
- If the distance from person1 and person2 are equal to Integer.minvalue, return null
- Else, create the object of biological relation and return it.

- String findCommonAncestor(FamilyTree familyTree, PersonIdentity person1, PersonIdentity person2)
  - Loop through the height of the tree, with iterator i
  - Get the ancestors for person1 with generation as i
  - Loop through the height of the tree with iterator j
  - Get the ancestors of person2 with generation as j
  - Check if there is common ancestry of both person1 and person2
  - If true, return “distance from person1”+” ”+”distance from person 2”
  - If no common ancestor found return string with a minimum value of the integer.

- Set descendents(PersonIdentity person, Integer generations)
  - Initialize set of descendants
  - Check if person is null or generations is zero. If true, return descendants
  - Initialize familyTree instance
  - Create tree
  - get family tree
  - Return getDescendantsRecursive()

- Set getDescendantsRecursive(currentGeneration, generations, person, tree, descendants)
  - Get the children of person and add all of them to the descendants
  - Increment current generation
  - if currentGeneration is equal to the generation, return the list of descendants
  - else, get descendants recursively of all the children

- Set ancestores(PersonIdentity person, Integer generations)
  - Initialize the set of ancestors
  - Check if person is null or generations is zero. If true, return ancestors

- Initialize familyTree instance
- Create tree
- set currentGeneration to 0
- Return getAncestorsRecursive()

- Set getAncestorsRecursive(int currentGeneration, Integer generations,PersonIdentity person, FamilyTree familyTree, Set<PersonIdentity> ancestors)
  - if currentGeneration is equal to the generation, return the list of ancestors.
  - Get the parents of a person and add all of them to the ancestors.
  - Increment current generation
  - Check if current generation is not equal to the generation
  - If yes, get ancestors recursively of all the parents
  - Return ancestors

- List<String> notesAndReferences(PersonIdentity person)
  - Execute the query to find the notes and reference for a person from the database and order by date
  - Add rows to the list
  - Return the list

- Set findMediaByTag(String tag , String startDate, String endDate)
  - Initialize set of fileIdentifier
  - Run the query to find media with the given tag that exists between the given start and end date.
  - Add all the output for the query to the set
  - Return the set.

- Set findMediaByLocation(String location, String startDate, String endDate)
  - Initialize set of fileIdentifier
  - Run the query to find media with the given location that exist between the given start and end date.
  - Add all the output for the query to the set
  - Return the set.

- List findIndividualsMedia(Set people, String startDate, String endDate)
  - Create a null list<FileIdentifier>, say output
  - Loop through the set of people:

- Find the media that consist of that people and exists between the given start and end date.
- Check if the media already exist in the list, if not add it to the list
- Return the list

- List findBiologicalFamilyMedia(PersonIdentity person)
  - Create a null list<FileIdentifier>, say output
  - Get the children of person.
  - Iterate through the set and find the media for each child.
  - Check if the media already exist in the list, if not add it to the list
  - Return the list

4. FamilyTree

- getFamilyTree(){
  - Return family tree

- createTree(List<PersonIdenity> listOfPeople)
  - Iterate through list of people called person.
  - Get the person object and store it as parent
  - Execute the query to get the descendants of the person.
  - Add the parent as key and list of descendants to the hashmap of tree.

- findPerson(int id, List<PersonIdentity> listOfPeople)
  - Return the person object from the list with given id

- FindParents(Map<PersonIdentity, List<PersonIdentity>> tree, PersonIdentity person)
  - Initialize arraylist parents
  - Loop through the tree
  - Check if value of list entrySet contains person in the list.
  - If yes, add its key to the arraylist.
  - Return parents

- getTreeHeight()
  - Set maxLength as 0
  - Loop through the hashSet of tree,
    1. Get the size for every key's value
    2. If size is greater than maxLength, set maxLength =size
  - Return maxLength