CSCI 3901

Software Development Concepts

Project: Publication Library

Intermediate Milestone Document

Name : Keyur Pradipbhai Khant

Banner ID : B00935171

Email ID: keyur.khant@dal.ca

Overview

The issue is the creation of a system to control data about academic publications, with a concentration on journal and conference publications. The system should be able to handle various levels of granularity in research fields and allow for the storage and retrieval of data regarding publications, authors, venues, research topics, and citations.

The set of authors, article title, journal name, page range, volume and issue numbers, publication month and year are the bare minimum details needed for a journal publishing. The required details for a conference publication are the group of authors, paper title, conference name, venue and year, and page range. Each venue for publication has a group that oversees the publication, a field of study it focuses on, and an editor or organizer with their contact information.

A class called "PublicationLibrary" that can store and manage data on publications, authors, venues, study fields, and citations should be included in the system's architecture. However, for this project, the class should only accept the IEEE citation format. The class should be capable of handling various publications and citation formats. Information can be stored by the class via internal data structures, databases, or files.

Overall, the system should provide a comprehensive and organized way to manage and analyze publication information in academia, making it easier for researchers to track their own work and that of others, as well as identify potential collaborators and seminal papers in their field.

Data structures and their relations:

We can use multiple data structures and ADT in this program. Those are mainly used to store publications, authors, venue and reference area’s data and other required information while operating.

Here are the main abstract data types (ADT) will be used while developing -

1. **Hash Map**
   Hash maps can be used to store the publications, where the keys are the alphanumeric citation strings associated with each publication, and the values are the publication information. This can provide efficient access to individual publications based on their citation strings. Furthermore, we can store venue information as well in the hash maps.

   Here are few examples -

   ```
   Map<String, String> venueInformation
   Map<String, String> publisherInformation
   ```

   Here each venue of publisher information is stored with specified relative id.

2. **Hash Set**
   Hash sets can be used to store the unique values such as collection of publication identifies, set of collaborators and parent areas.
   The **seminalPapers** method returns a set of publication identifiers because it allows for easy storage, retrieval, and iteration through the collection of seminal papers that meet the specified criteria. There are other places where we are using a hash set to store unique information.

```
Set<String> parentArea
Set<String> seminalPapers
Set<String> collaborators
```

3. **Graphs**
   Graphs can be used to represent the author collaborations between different publications. Each node in the graph would represent an author, and edges between nodes would represent collaborations between the authors in the form of co-authorship on a publication. This can help in identifying potential grant collaborators for an author.

   Currently, I am thinking about using Graph to develop. There isn't any prompt idea to use throughout the project.

Algorithm for given problem statement:

The higher level of an algorithm for publication library problem is as below -

1. Start
2. Create a class called Publication that will store all the information about a publication and methods to return information related to publication such as authors, title, page range and venue.
3. Create a class called Author that will store all the information about an author, including their name and publications.
4. In the Author class, include attributes such as name, contact information, and a list of publications. This publication will be the list of objects of publication.
5. Create a class called PublicationLibrary that will manage the information about publications and authors.
6. In the Publication class, include the following attributes: authors (list of Author objects), title, publication type (journal or conference), publication venue (name of journal or conference), page range, volume, issue number, publication date (month and year), research area, and publication ID (unique alphanumeric string associated with each publication).
7. In the PublicationLibrary class, include the following methods:
   a. **addPublication():** to add a new publication to the library
   b. **addAuthor():** to add a new author to the library
   c. **addReferences():** to add the reference for a specific publication
   d. **addVenue():** Adding venue for any publication in the library
   e. **addPublisher():** Adding publishers details in the library
   f. **getSeminalPapers():** to get a set of publication IDs that meet the criteria for being considered seminal papers in a given research area
   g. **authorCitations():** Number of citations any specific author has.
8. There are other methods as well to handle various operations.
9. Create a **researchArea** class to handle various research areas and its publication details.
10. Develop a method to convert each citation and its references to add in paper. In this situation, consider only IEEE format. (For example, **convertToIEEEFormat** method to handle)
11. End

UML diagram for PublicationLibrary:

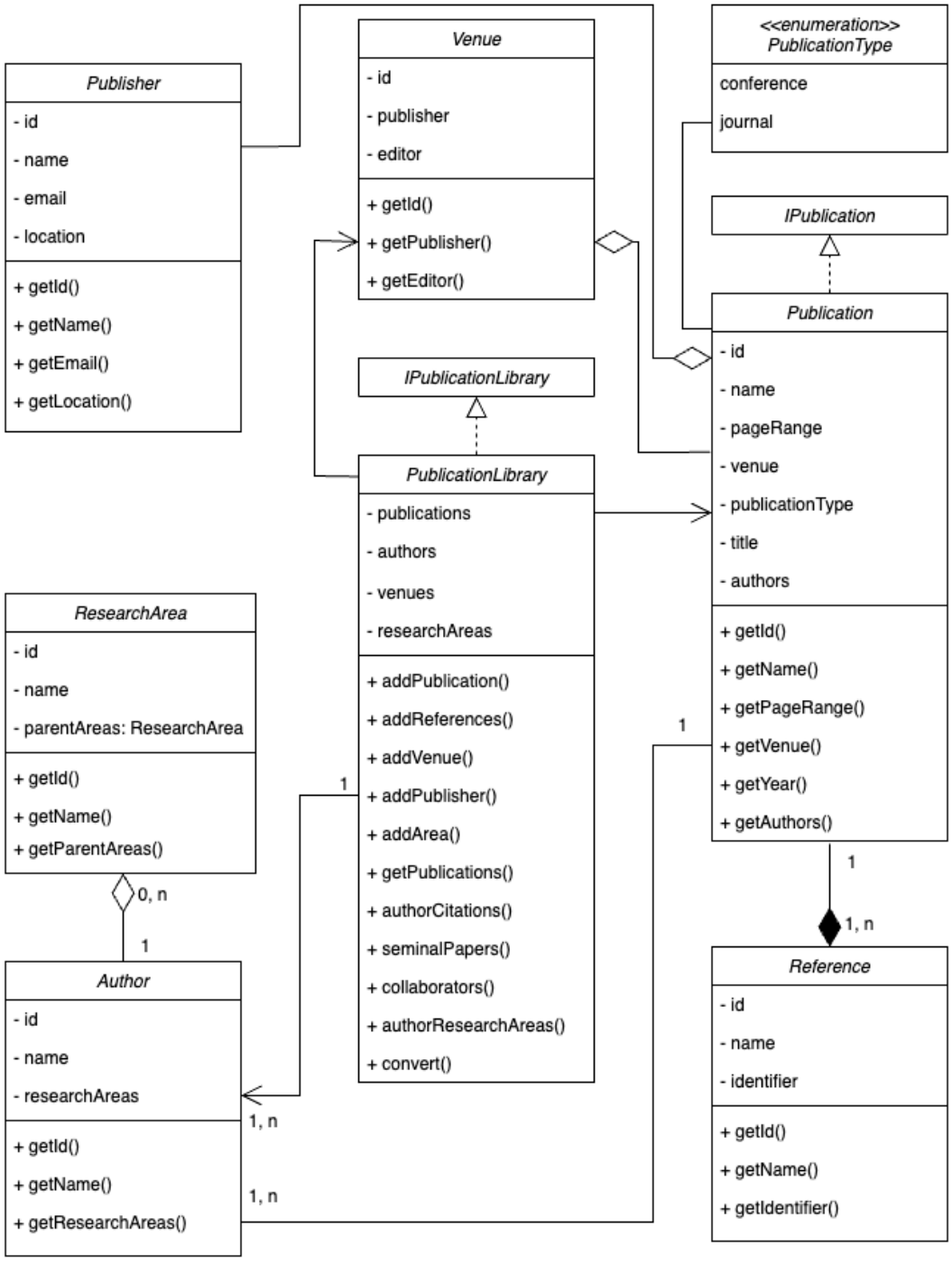

Figure 1: ***UML diagram of PublicationLibrary***

Test cases for PublicationLibrary

Black box testing is a testing technique in which the tester concentrates on the functioning of the product while not knowing anything about its internal structure or code.

For PublicationLibrary, we can pre-plan the black box test scenarios for all of the class's given methods. Input validation, boundary case analysis, control, and data flow are the four key areas we can examine.

boolean addPublication (String identifier, Map<String, String> publicationInformation)

Input validation

1. The identifier parameter should not be null
2. The identifier parameter should not be empty.
3. The publicationInformation parameter should not be null.
4. publicationInformation map's size should be greater than 0.

Boundary cases

1. If the publicationInformation Map is empty or does not contain the required keys, the method should return false.
2. If the identifier already exists in the library, the method should return false.
3. If the authors, title, and either journal or conference key do not exist in the map then it should return false.

Control flow

1. Proper identifiers should be available.
2. Proper publicationInformation should be available.
3. Proper key for the publicationInformation should be available.
4. Identifiers should not already exist in the library.
5. Create a Publication object with the information from the publicationInformation Map.
6. Add the Publication to the library.
7. If publication is added then return true else return false in all other conditions.

Data flow

1. Two parameters of the method: identifier (String) and publicationInformation (Map).
2. The publicationInformation Map contains the information needed to create a Publication object.
3. The Publication object is generated and added to the library using the information from the publicationInformation Map.

boolean addReferences (String identifier, Set<String > references)

Input validation

1. The identifier is null.
2. The identifier is empty.

3. The references are null.
4. The reference is empty.

Boundary cases
1. If the publication with the given identifier does not exist in the library, return false.
2. If the references already exist with the given identifier, return false.

Control flow
1. Validate if the publication with the given identifier already exists in the library. If not, return false.
2. For each publication identifier in the references set:
   a. Check to see if the publication is available in the library. If not, add the publication to the library using as little information as possible because we don't have all of the relevant information.
   b. Add the current publication identifier to the list of references for the supplied publication.

Data flow
1. Using the publications map, see if the publication with the supplied identification exists in the library.
2. Check if the publication exists in the map using the identifier as the key for each identification in the references set.
3. If the publication does not already exist, build a new one with the bare minimum of information and add it to the map.
4. Using the references map, add the current publication identification to the list of references for the publication with the supplied identifier.

boolean addVenue (String venueName, Map<String, String> venueInformation, Set<String> researchAreas)

Input validation
1. The venueName is null.
2. The venueName is empty.
3. The venueInformation is null.
4. The researchAreas is null.

Boundary cases
1. If venueInformation is with 0 size, return true.
2. If researchAreas is with 0 size, return true.

Control flow
1. Check if the venue already exists in the library, and return false if it does.
2. If the venue exists in the library, return false.
3. Venues should be added in the library.
4. Research areas should be added to the library.

Data flow
1. The venueName is used to indicate the location of the venue in the library.
2. The venueInformation is a map that contains venue information.
3. The researchAreas is a string collection that represents the research areas linked with the venue.

boolean addPublisher (String identifier, Map publisherInformation)

Input validation
1. The identifier is null.
2. The identifier is empty.
3. The publisherInformation map is null.

Boundary cases
1. If the identifier is null or empty, return false.
2. If the publisherInformation map is null or with size 0, return false.

Control flow
1. If a publisher with an identifier already exists, return false.
2. If the publisher does not exist, generate a new one and provide information to it and add it to the library.

Data flow
1. The publisher information can be obtained from the publisherInformation map.
2. Create a new Publisher object with the retrieved data.
3. Add the new Publisher object.

boolean addArea (String researchArea, Set<String> parentArea)

Input validation
1. The researchArea is null
2. The researchAre is empty.
3. The parentArea set is null.

Boundary cases
1. If the researchArea is null or empty, return false.
2. If the parentArea is null, return false.

Control flow
1. Validate reseachArea is already exists or not, if it is, return false.
2. If the researchArea does not exist, generate a new one and add it to the set.
3. If parentArea is empty, generate an individual research area.

Data flow
1. Generate object of research area if all validated data exists.
2. Add a research area in the library.

Map getPublications (String key)

Input validation
1. The key is null or empty.

Boundary cases
1. If the given key does not exist, return false.

Control flow
1. A valid key exists in the library.
2. Get all publication information associated with the key from the library.
3. Generate a key-value pair of publication information and return in the form of a map.

Data flow
1. Valid publication map including references information should be returned

int authorCitations (String author)

Input validation
1. The author is null or empty.

Boundary cases
1. If the given author does not exist or is null, return 0.
2. If there are not any publications for a given author, return 0.

Control flow
1. Iterate through the library's publications.
2. Check whether the provided author is listed in the publication's "authors" class.
3. Iterate through the publication's "references" field if the author is founded.
4. Check the cited publication's "authors" field for the supplied author for each reference.
5. Increment the author citation count whenever a new author is founded.

Data flow
1. The method returns an integer representing the number of citations for the given author.

Set<String> seminalPapers (String area, int paperCitation, int otherCitations)

Input validation
1. The area is null or empty.
2. paperCitation is a non-negative integer.
3. The otherCitations is a non-negative integer.

Boundary cases
1. If no publications exist in the chosen research field, return an empty set.
2. Return an empty set if there are no publications in the library.
3. Return an empty set if all papers in the given study field have less direct citations than otherCitations.
4. Return an empty set if there are no publications in the given study area with fewer than paperCitation referrals to other publications in the same area.

Control flow
1. Check that the area parameter is not null and represents a legitimate research area.
2. Check that the parameters paperCitation and otherCitations are non-negative integers.
3. To store the seminal papers, create an empty set.
4. Iterate over each paper:
   a. If the publication has fewer than paperCitation references to other publications in the same area, skip it.
   b. If the publication has more than otherCitations direct citations, skip it.
   c. Otherwise, add the publication identifier to the set of seminal papers.
5. Return seminal paper set.

Set<String> collaborators(String author, int distance)

Input validation
1. The author is null or empty.
2. The distance is null
3. The distance is non null integer.

Boundary cases
1. If the given author does not exist in the library, return an empty set of strings.
2. If the distance parameter is 0, return a set containing only the author parameter.
3. If there are no authors in the library, return an empty set.

Control flow
1. Obtain all of the publications to which the author has contributed.
2. Get the set of all other authors who have contributed to the same publication as the author for each publication.
3. Check whether the author distance from the other authors is less than or equal to the distance.
4. If this is the case, include the author's name in the result set.
5. Return the set of results.

Set<String> authorResearchAreas (String author, int threshold)

Input validation
1. The author is null or empty.
2. The threshold is null

3. The threshold is a non zero integer.

Boundary cases

1. Return an empty set if there are no publications in the library.
2. If there are articles in the library but none of them are related with any research areas, return an empty set.
3. If the author has not published in any of the library's research areas, return an empty set.

Control flow

1. Obtain all articles related to the specified author.
2. Retrieve the research areas linked with each publication.
3. Check to see if each research area has already been added to the result set.
4. If it hasn't been added to the result set, see if the author has produced at least one threshold number of publications in that field.
5. Add it to the result set if the author has published at least one threshold number of papers in the research area.
6. Determine whether the research area has any parent areas.
7. If it does, include the parent regions in the result set.
8. if they haven't already been included.

Conclusion

- Till now, I have designed the whole program architecture with UML diagrams as described above.
- Worked on black box test scenarios of all given methods.
- As per the above description, I will start to implement the program for each class.
- There might be some changes in the class architecture according to the SOLID principles while developing the program.
- If there are any changes, I will update the UML diagram as per the changes and add it to the document directory on the git repository.

References

[1] "Flowchart maker & online diagram software," *Diagrams.net*. [Online]. Available: https://app.diagrams.net/. [Accessed: 13-Apr-2023].

[2] "Diagram maker for developers," *Gleek.io*. [Online]. Available: https://www.gleek.io/blog/class-diagram-arrows. [Accessed: 13-Apr-2023].

[3] G. McKeever *et al.*, "Black box testing," *Learning Center*. [Online]. Available: https://www.imperva.com/learn/application-security/black-box-testing/. [Accessed: 13-Apr-2023].