




# CSCI 3901 SOFTWARE DEVELOPMENT CONCEPTS FINAL PROJECT



Sushank Saini  
B00922727

## Overview

This program manages the information of publications. It adds the research areas, publication venues, conference and journal papers, the references cited in those conference and journal papers and the publishers/organizers of the publication venue. All the information is added in the respective tables in the database(refer **Figure 1**) as the data needs to survive between the executions of the program.

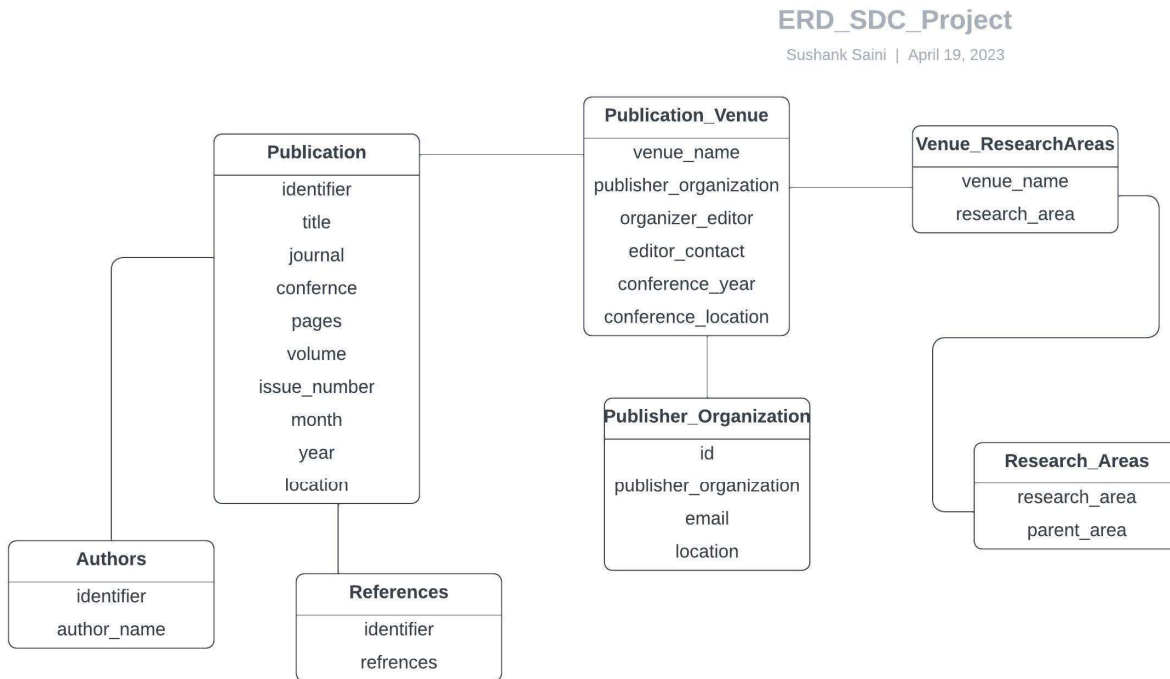


Figure 1: **Database tables design**

This program then retrieves the information from the library such as getting the publication information of a journal or a conference paper.

This program also converts the citations cited in a publication to the IEEE references format.

## Files and External Data

The program consists of the following files(refer **Figure 2** for the class design):

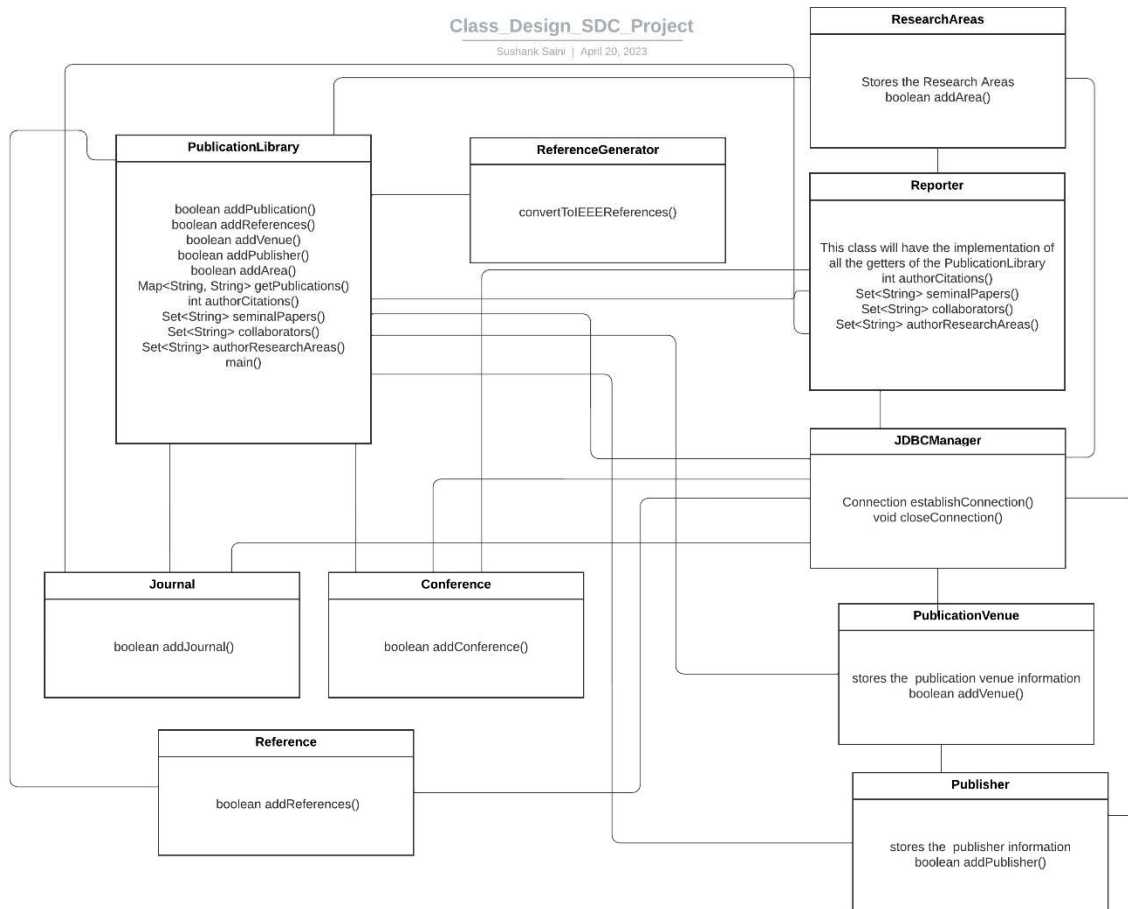


Figure 2: Class design

- *JDBCManager.java*: This is a manager class that handles the database connection.
- *PublicationLibrary.java*: This file has functions calls to all the methods of the PublicationLibrary. It also has the main() method which takes input from the user for input and output file names and then calls the function that converts the citations to IEEE format.
- *ResearchAreas.java*: This is a helper class that adds the research area and its parent research areas in the table research\_areas.
- *Publisher.java*: This is a helper class that adds the publisher information of a publication venue into the table publisher\_organization.
- *PublicationVenue.java*: This is a helper class that adds the publication venue information into the publication\_venue table. The research areas information for a publication venue is stored in a separate table called venue\_research\_areas for a given venue\_name.
- *Journal.java*: When addPublication() is called and the publication is a journal then this helper class adds the journal publication into the publications table. The authors for a given journal publication are added in a separate table called authors.
- *Conference.java*: When addPublication() is called and the publication is a conference then this helper class adds the conference publication into the publications table. The authors for a given conference publication are added in a separate table called authors.
- *References.java*: This is a helper class that adds and updates the references of a given publication.
- *Reporter.java*: This class consists of all the reporting functions like getPublications() etc.
- *ReferenceGenerator.java*: This class has the method that converts the citations into IEEE references.

## Data Structures

- *Map<String,String> journalInfo*: this map is used to return the publication information of a journal with its mandatory information only when getPublication() is called.
- *Map<String,String> conferenceInfo*: this map is used to return the publication information of a conference with its mandatory information only when getPublication() is called.
- *Map<String,Integer> citationsMap*: this map is used to keep track of the citations encountered in the input file and the corresponding reference numbers assigned to those citations.
- All other data is stored in the tables(see **Figure 1**).

## Assumptions

- Assuming that the research area for a publication venue already exists in the research\_areas table.
- Assuming that publisher/organization for a publication venue already exists in the publisher\_organization table.

- Assuming that the publication venue for a publication paper already exists in the table publication\_venue.
- Assuming that publisher/organization name is unique.
- Assuming that while converting to IEEE format, the alphanumeric identifier used in the input file has an entry in the publications table already.
- Assuming that no author will have four parts to their name.

## Choices made in design

- I have segregated implementation of each method into a separate helper class(see **Figure 2**) to provide more cohesion.
- I am storing the references and parent areas as a multivalued attribute in the database for easy retrieval(see **Figure 1**).
- I am storing the journal and conference publication information in a single table publications for easy retrieval of the information (see **Figure 1**).

## Key algorithms and design elements

- Updating references:
  - First retrieve the existing references for the given identifier
  - Convert the existing references into a set
  - Perform the union of the existing set of references and the updated ones
  - Convert the updated set of references to a multi valued attribute
  - Update the references in the table
- Converting the citations to IEEE references:  
For converting the citations to the IEEE references I am using the Regex API[1][2].
  - For each line in the input file:
    - Create the regular expression to match the identifier pattern \cite{alphanumerics} using Pattern class[3].
    - Search for this pattern in the line using Matcher class.
    - When a match is found, extract the comma-separated list of citation identifiers from the matched text using match.group() function[4][5].
    - Store the identifier and its corresponding reference number in the map and build a string that has identifier's corresponding reference number:
      - If the identifier is already present, its reference number is appended to the string of numbered references.
      - If not, then the new identifier and its reference number is added to the map and then reference number appended to the string of the numbered references
    - Replace the entire \cite{alphanumeric} string with the corresponding string of reference numbers enclosed with square brackets in the line
    - Append that line to the required output string
    - Repeat till all lines are read

- Create the reference list by iterating the map having the identifiers and the corresponding reference numbers:
  - For each identifier, fetch the publication information, abbreviate the author names and add it to the required output string.
  - Append that information to the required output string
- Write the output string to the output file.
- Abbreviating the author names:
  - Split the string of authors into an array
  - For each author name, check if the name has only a first or first and second or first, middle and second name and abbreviate the first or second name accordingly by retrieving the first character of the first and/or second name.

## Type of Exception Handling

I have used try-catch block to handle the exceptions. The catch blocks move from handling the specific exceptions to general exceptions.

## Limitations

- The program does not check if the publisher/organization already exists in the database when a publication venue is added.
- The program does not check if the research area already exists in the database when a publication venue is added.
- The program does not check if the publication venue of a publication already exists when a publication is added.
- The program does not have the functionality to find the collaborators, seminal papers and research areas of the author.

## References

- [1] "Java - Regular Expressions," *tutorialspoint* [Online]. Available: [https://www.tutorialspoint.com/java/java\\_regular\\_expressions.htm](https://www.tutorialspoint.com/java/java_regular_expressions.htm). [Accessed: April 17,2023].
- [2] "Java Regex," *javatpoint* [Online]. Available: <https://www.javatpoint.com/java-regex>. [Accessed: April 17,2023].
- [3] "How to write Regular Expressions?," *GeeksForGeeks* [Online]. Available: <https://www.geeksforgeeks.org/write-regular-expressions/>. [Accessed: April 18,2023].
- [4] "Regular Expressions in Java," *GeeksForGeeks* [Online]. Available: <https://www.geeksforgeeks.org/regular-expressions-in-java/>. [Accessed: April 17,2023].
- [5] K.Mangal, "Matcher group(int) method in Java with Examples," *GeeksForGeeks* [Online]. Available: <https://www.geeksforgeeks.org/matcher-groupint-method-in-java-with-examples/>. [Accessed: April 17,2023].

## Test Cases

addArea(String researchArea, Set<String> parentArea)

### *Input Validation/Boundary cases*

- researchArea is repeated/duplicate
- parentArea is null/empty
- researchArea is null/empty
- researchArea is not null/empty but parentArea is empty/null
- parentArea is not null/empty but is researchArea empty/null

### *Data Flow*

- Research area is added after adding publication venue or the publication/organization.

### *Control Flow*

- connection is closed before the query is executed
- query is executed after closing the prepared statement

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

### *Input Validation/ Boundary cases*

- venue name is repeated/duplicate
- venueName is empty/null
- venueInformation is empty/null
- researchAreas is empty/null
- venueName is not empty/null, venueInformation is empty/null, researchAreas is empty/null
- venueName is not empty/null, venueInformation is not empty/null, researchAreas is empty/null
- venueName is empty/null, venueInformation is not empty/null, researchAreas is not empty/null
- venueName is not empty/null, venueInformation is empty/null, researchAreas is not empty/null
- venueName is not empty/null, venueInformation is not empty/null, researchAreas is empty/null
- venueName is not empty/null, venueInformation is not empty/null, researchAreas is empty/null

### *Data Flow*

- Venue is added before the publisher or research area is added
- Venue is added after publication and references are added

### *Control Flow*

- queries are executed before checking if it's a journal or conference
- research areas are added before the query is executed
- connection is closed before the query is executed
- query is executed after closing the prepared statement

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

### *Input Validation/Boundary cases*

- Identifier is not alphanumeric
- Identifier is null and publicInformation is empty
- Identifier is null and publicInformation is not empty
- Identifier is not null and publicInformation is empty
- Identifier is not null and publicInformation is not empty
- Identifier is repeated/duplicate
- publicationInformation has null values for some of the keys
- publicationInformation does not have the mandatory keys for either the journal or publication

### *Data Flow*

- Publication is added before adding the publisher, venue or research area
- Publication is added after adding the references

### *Control Flow*

- authors are added before executing the query
- connection is closed before the query is executed
- query is executed after closing the prepared statement

addReferences(String identifier, Set<String> references)

### *Input Validation/Boundary cases*

- identifier is repeated or duplicate
- identifier does not exist in the publicationLibrary
- identifier is not alphanumeric
- identifier is null/empty and references is empty/null
- identifier is null/empty and references is not empty/null
- identifier is not null/empty and references is empty/null
- identifier is not null/empty and references is not empty/null

### *Data Flow*

- References are added before adding the publication



### *Control Flow*

- the cursor of resultset is not set to next before performing operation on it
- connection is closed before the query is executed
- query is executed after closing the prepared statement
- connection is closed before the resultset
- resultset is closed before the query is executed

addPublisher(String identifier, Map<String, String> publisherInformation)

### *Input Validation/Boundary cases*

- identifier is repeated/duplicate
- Identifier is null/empty and publisherInformation is empty/null
- Identifier is null/empty and publisherInformation is not empty/null
- Identifier is not null/empty and publisherInformation is empty/null
- Identifier is not null/empty and publisherInformation is not empty/null

### *Data Flow*

- Publisher is added before research area is added
- Publisher is added after venue is added

### *Control Flow*

- connection is closed before the query is executed
- query is executed after closing the prepared statement

Map<String, String> getPublications(String key)

### *Input Validation/Boundary cases*

- key is null
- key is empty
- the key does not exist in the publications table

### *Data Flow*

- Method is called before adding that particular publication and its references

### *Control Flow*

- check for differentiating between a journal or conference is done after executing the query
- the cursor of resultset is not set to next before performing operation on it
- connection is closed before the query is executed
- query is executed after closing the prepared statement
- connection is closed before the resultset
- resultset is closed before the query is executed

int authorCitations(String author)

*Input Validation/Boundary cases*

- author is null
- author is an empty string
- author does not exist in the authors table
- there are multiple authors with same name

*Data Flow*

- Method is called before adding those publications that use the that authors publications

*Control Flow*

- connection is closed before the query is executed
- query is executed after closing the prepared statement
- connection is closed before the resultset
- resultset is closed before the query is executed

convertToIEEERefrences(inputFileName,outputFileName)

*Input Validation/Boundary cases*

- input and output files do not exist

*Data Flow*

- function is called before that publication is added to the table but is cited in the input file

*Control Flow*

- file is closed before the pattern matching is done
- pattern is matched before the pattern is set
- citations in the map are checked after appending the number to the reference number string
- reference number is incremented before making an entry into the map
- line is replaced before getting the matched group.
- Check for author having just the first name is done before checking for the middle and last name
- Check for journal or conference is done after appending the information to the output.

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

*Input validation/Boundary cases*

- area is null
- area is empty string
- area does not exist in the research\_areas table
- paperCitation is 0
- otherCitations is 0
- paperCitation and otherCitations are both more than the total number of papers in the given area

*Data Flow*

- method is called before adding the publication and the references

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

*Input validation/Boundary cases*

- author is null
- author is empty string
- author does not exist
- author has no collaborators or distance is 0
- author has collaborators but they are not in the given distance

*Data Flow*

- method is called before the publication is added and references are added

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

*Input validation/Boundary cases*

- author is null
- author is empty string
- author does not exist
- threshold is 0
- threshold is more than the total number of papers published by the author

*Data Flow*

- method is called before the research areas are added and publications are added