CSCI – 3901: Project

Overview

Implement a Java program to perform tasks with the database to insert data in the database in an appropriate manner and access those data to perform the required task. The basic task for the code is to manipulate that in such a manner that the user will upload an external document based on the research paper they are writing and this Java code will convert that paper with citations to the specific IEEE format and make an output document.

Requirements:

The requirements for this problem from the user's perspective:

I think of the structure with two users. 1. Admin 2. Actual user

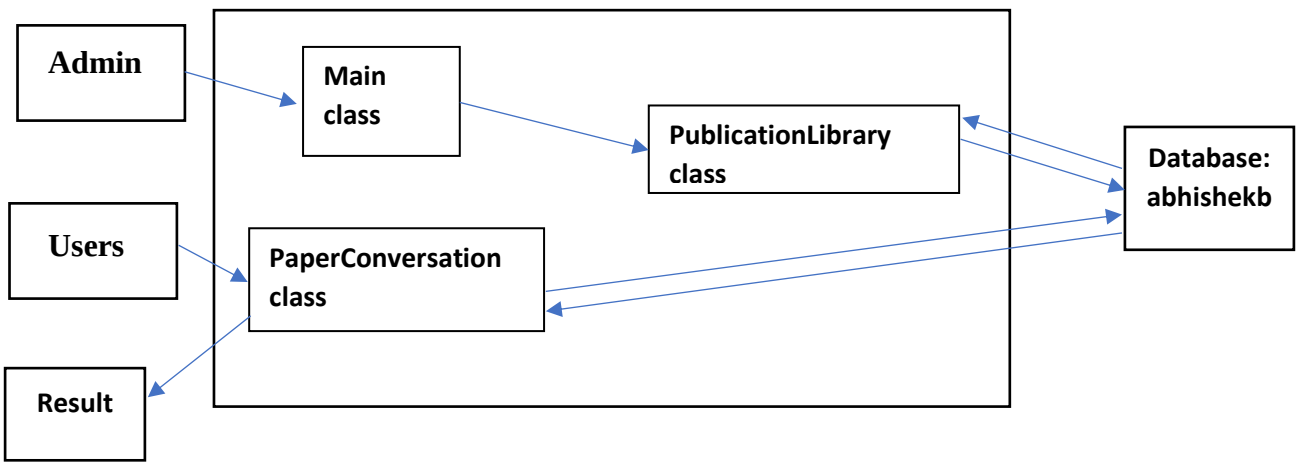
For the admin:

1. Admin can enter the data in the database based on the information they have.
    - Add Author data.
    - Add Publisher data
    - Add Venue data
    - Add Publication Data
    - Add Research-area data
    - Add author – publisher connection
    - Add references to the publications which connect to the Authors.
2. get methods and other methods to manipulate the data.

For the users:

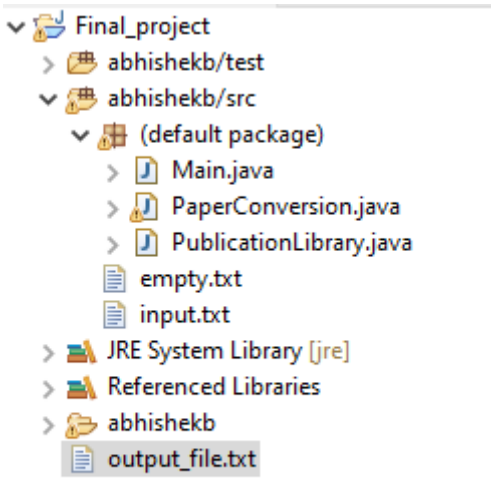
1. Upload the paper to convert in the specific folder provide the name of the input file and provide the name of the output file they needed to convert that input file in.

System design:



Files and Data

In this code, I used 3 classes to preform required task in such a manner that it can store and process the data.



This code mainly focused on the PublicationLibrary class which stores and get the data from the database and PaperConversion class which take input and output file name to process the final result for the user.

Work-flow

I divided the workflow into 3 steps:

1. Make a database design to store data.
2. Decide on the flow of the data. i.e. What to add first and what can be the relation between the tables.
3. Make insertion methods to add data to the database.
4. Make get methods to get the required data from the database.
5. Make user accessed class to enter input and output file name in the console to process it to desired output.

Assumption

- Data is present in the database.
- Author's full name will be unique.
- Reference will be in the form of \cite{lastName_year}
- No data redundancy will be added to the database.
- Users have to upload a document to a specific folder to convert it.
- IEEE format assumptions.

Explanation: (step by step)

PublicationLibrary class

The class has several instance variables defined:

- "username" and "password" that store the credentials to connect to the database
- "connect" of type Connection, which is used to establish a connection with the database
- "statement" of type Statement, which is used to execute SQL queries

The class also has a constructor that establishes a connection to the database using the credentials stored in "username" and "password".

The methods in the class are:

"**getConnection()**": returns a Connection object after establishing a connection to the database. It sets the database to be used in the connection.

"addPublication(String identifier, Map<String, String> publicationInformation)": inserts a new publication record into the Publication table in the database. It takes two arguments: the unique identifier of the publication and a Map of its information.

"addAuthor(String identifier, String fullName): inserts a new author record into the Author table in the database. It takes two arguments: the unique identifier of the author and their full name.

"addPublicationAuthor(int publicationId, int authorId)": inserts a new publication-author record into the Publication_Author table in the database. It takes two arguments: the unique identifiers of the publication and the author.

"addReferences(String identifier, Set<String> references): inserts a new reference record into the Reference table in the database. It takes two arguments: the unique identifier of the publication and a set of citation strings.

"addVenue(String venueName, Map<String, String> venueInformation, HashSet<String> researchAreas): inserts a new venue record into the Venue table in the database. It takes three arguments: the name of the venue, a map of its information, and a hash set of its research areas.

"addPublisher(String identifier, Map<String, String> publisherInformation): inserts a new publisher record into the Publisher table in the database. It takes two arguments: the unique identifier of the publisher and a map of its information.

"addArea(String researchArea, Set<String> parentArea): inserts a new research area record into the Research_Area table in the database. It takes two arguments: the name of the research area and a set of its parent areas.

"getPublications(String key): retrieves a publication record from the Publication table in the database. It takes one argument: the unique identifier of the publication. It returns a map of the publication's information.

“public int authorCitations(String author)”: This method takes an author name as input and returns the number of citations for all the publications that the author has been associated with.

“public Set<String> seminalPapers(String area, int paperCitation, int otherCitations)”: This method takes a research area name and two integer values as input, and returns a set of publication IDs for the papers that meet the following criteria:

- The publication belongs to the given research area.
- The publication has been cited fewer times than the given paperCitation threshold by other papers within the same database.
- The publication has been cited more than the given otherCitations threshold by other papers within the same database.

“public Set<String> collaborators(String author, int distance)”: This method takes an author name and an integer value as input, and returns a set of collaborator names for the given author within the specified distance of collaboration. For example, if distance is 1, the method should return all authors who have collaborated directly with the given author. If distance is 2, the method should return all authors who have collaborated with the given author's direct collaborators, and so on.

“public Set<String> authorResearchAreas(String authorName, int threshold)”: This method takes an author name and an integer value as input, and returns a set of research area names for the areas in which the given author has published at least threshold papers. The method also includes all parent research areas of the identified areas recursively.

All the methods use SQL queries to interact with the database. The SQL queries are defined as strings and stored in variables. The variables are used to create PreparedStatement objects, which are used to execute the queries with parameters. The results of the queries are stored in ResultSet objects, which are used to extract data and create maps.

Overall, the PublicationLibrary class provides a set of methods that allow users to interact with a database that stores information about academic publications. The methods cover a range of operations, from inserting new records to retrieving existing ones.

PaperConversion class

This class, PaperConversion, is used to convert a LaTeX document containing citation keys into a new LaTeX document with the corresponding IEEE-style references. The main method of this class prompts the user to input the names of an input and output file, and then reads the contents of the input file as a string.

“getAbbreviatedAuthors” method is used to generate a string of abbreviated author names in the format "A. B. Last name, C. D. Last name, ...". The method takes a list of authors as input and iterates through the list, splitting each author's name into first and last names, and then abbreviating the first names and appending them to the output string along with the last name.

"getIEEEReference" method retrieves the publication details and authors for a given citation key from a database and returns an IEEE-style reference string for the publication. The method executes an SQL query to retrieve the publication details and another query to retrieve the authors.

"getAbbreviatedAuthors" method is used to generate an abbreviated author string for the publication, and this string is then used to build the IEEE reference string.

"replaceCitations" method replaces citation commands in the given text with their corresponding reference numbers from the citationMap. The method iterates through the entries in the citationMap and replaces all occurrences of each citation key in the text with the corresponding reference number in brackets.

"main" method is the main entry point of the class. It prompts the user to enter the name of the input and output files, reads the contents of the input file, and uses regular expressions to extract the citation keys from the text. It then iterates over each citation key, retrieves the corresponding IEEE-style reference string using the getIEEEReference method, and stores the citation key and index in the reference list in a citationMap. It then uses the replaceCitations method to replace the citation commands in the text with their corresponding reference numbers and writes the modified text and the reference list to the output file.

Limitations

- The program uses static data for the connection of the database.
- The program works on certain assumptions and will throw errors if the user input is not as expected.
- I have not implemented the paper and journal differences in the given code. I can make a different Enum class for the relation and get the data on the output documentation based on that but due to the time limitation, I was more focused on the end-to-end structure.
- The database is more complex, I can reduce the complexity to avoid redundancy of the data insertions.
- There is two class with static main method one for the users and one for the admin which can be combined into one for better performance.
- Cite entry in input validation has the limitation of citation methods that can be solved based on the format but I was unable to do that at this time that can be implemented in the next iteration with the paper types implementation.
- The data from the ResultSet is added to the appropriate XML elements.
- The XML document is written to a file using the Transformer class.

Overall, the code uses standard Java classes and follows the best practices for connecting to a database, querying it but loosely coupled and dependent on the user. As this can only be handled by the developer who developed it or those who have a strong knowledge of the database and the flow of the code. From the user's perspective, all they have to know is the final location of the input file to be uploaded but has the limitation of types in the paper.