CSCI 3901 Final Project

Due date: 11:59pm Thursday, April 20, 2023 in git.cs.dal.ca. The repository is at https://git.cs.dal.ca/courses/2023-winter/csci3901/project/xxx.git where "xxx" should be replaced by your timberlea login id. The repository is already created for you, so you should just need to clone it.

Goal

The course project is your opportunity to demonstrate all of the concepts from the course in one body of work.

Project Structure

The final project will have you apply your problem solving and development skills. The solution will require you to bring together

- Abstract data types and data structures
- Java implementation
- Basic algorithms
- Software development techniques including version control, testing, debugging, and defensive programming
- Good software program design
- Database design and use

This project is _not_ expected to include a user interface.

Problem

Publications are a staple of academia. We rely on publication citations to give credit to others for the work that they have done. We also rely on citations to gauge how impactful a research is, based on how much other people build on their work and on whether an author is the first author of a paper or how far down the list of authors someone's name is.

You will create a system to help a research community use and understand a common body of publication information.

The project

Create a class called "PublicationLibrary" that lets us manage information about publications. We restrict the class to only manage journal and conference publications, but your design should be ready to add other forms of publications like books, white papers, or web-based publications in the future. For this project, we will only worry about citations whose eventual output format is the IEEE format, so you can use that formatting guide to determine what information is mandatory in a citation and which information is optional.

The class can use internal data structures, databases, or files to store information, however you choose to design the class.

The information managed by the class should survive between execution of programs that use the PublicationLibrary class.

Here is the minimum information that you will need for the project.

The information we typically capture for a journal publication is the set of authors, the paper title, the journal name, the page range for the publication, the volume and issue number, and the month and year of the publication. The information for a conference publication is the set of authors, the paper title, the conference name, location and year, and the page range for the publication. Each publication venue (journals and conferences) have some organization that organizes the publication, an area of research that the journal or conference covers, and an editor or organizer with their contact information. The organizations then have their own contact information and home office.

Unfortunately, definitions of research area can vary. Some can be very narrow, like "computational geometry" while others can be broad, like "theory", which includes computational geometry. Your solution should be ready to handle a research area being identified as a part of some larger research area.

Every publication in the publication library will have an alphanumeric string associated with it. That string will be what authors use to cite the paper in their writing. A separate tool then converts these citation strings into actual references for the paper.

Ultimately, we want to be able to answer the following questions or tasks:

- How many citations does an author have?
- Which are the seminal papers in a given research area?
- What are the references in a given article?
- Who lies within X publication authorships from me, as potential grant collaborators?
- In what areas does an author do research?
- Given a paper, convert the citation strings into actual IEEE citations.

Your task is to create classes that will gather the information for this system and resolve these queries.

Methods for PublicationLibrary

You can add exception throwing, as appropriate, to the given methods.

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

Add a publication to the library. All the publication information is in the Map where the Map keys are the type of information and the Map values are the actual information. Expected Map keys include authors, title, journal, pages, volume, issue, month, year, conference, location, although not all of these keys will appear for each publication.

The author value consist of a comma-separated list of author full names (no abbreviations).

Return true if the publication has been added and false if the publication is not added to the library.

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

For the publication whose publication identifier is "identifier", record that that paper referenes all of the publications whose publication identifiers are in "references".

Since the library is a growing entity, we may call addReferences multiple times for one research paper as its references are added to the library and get publication identifiers. When called repeatedly, later calls may or may not include previously-reported references.

Return true if the references have been added and false if the references are not added to the library.

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

Add a publication venue, like a journal or a conference to the library. All the venue information is in the Map where the Map keys are the type of information and the Map values are the actual information. Expected Map keys include publisher, editor, editor_contact, location, and conference_year, although not all of these keys will appear for each publication.

Return true if the venue has been added and false if the venue is not added to the library.

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

Add a publisher to the library. All the publisher information is in the Map where the Map keys are the type of information and the Map values are the actual information. Expected Map keys include contact_name, contact_email, and location.

Return true if the publisher has been added and false if the publisher is not added to the library.

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

Add a research area to the library. The research area may be a subset of zero or more other research areas, as provided by the parentArea set.

Return true if the research area has been added and false if the area is not added to the library.

Map<String, String> getPublications (String key)

Return a map of all the information the library currently stores on a specific publication, as identified by the publication key. The articles that this paper references will be returned with a Map key of "references" and a comma separated string of all the publication identifiers cited by the article.

int authorCitations (String author)

Report how many publications directly cited the given author in their publications. If one publication cites two different papers of the author then count each citation separately.

The author's name will be the full name of the individual.

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

A seminal paper is one that has had a huge influence in a research area. In the case of this method, a seminal paper is one that has few references of its own in the current research area, but has many other papers in the research area referencing it.

Report the publication identifiers of the seminal research papers in the given area. The papers you return must not cite more than paperCitation papers in the given area and must have at least otherCitations papers citing it directly.

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

Report the full names of all people whose author distance to the given author is at most the "distance" parameter value. These people represent people who the given author might be able to easily reach out to for collaboration opportunities, like investigating a new research problem or writing a research grant.

The author distance is the length of the shortest chain that links one author to another through co-authorship on papers. Suppose that we have the following sets of authors on 5 papers: {A, B, C}, {B, D}, {B, E}, {C, F}, {F, E}. So, A published a paper with B and C. Then the author distance of A with themself is 0, A's author distance to C is 1 (co-authored a paper), and A's author distance to E is 2 (from A to B, to E, which is shorter than A to C to F to E).

This method is a generalization of the Erdős number[1] in mathematics or the Bacon number[2] in Hollywood movies.

---

[1] https://en.wikipedia.org/wiki/Erd%C5%91s_number

[2] https://en.wikipedia.org/wiki/Six_Degrees_of_Kevin_Bacon#:~:text=Bacon%20numbers,-A%20map%20of&text=The%20Bacon%20number%20of%20an,Kevin%20Bacon%20the%20actor%20is.

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

Report the research areas in which the given author has published at least "threshold" papers. When a research area is part of a larger research area, include both the specific research area and the broader research areas in the returned list.

Paper conversion

You will also write a main() method in a class of your choice that will convert citations in a paper into actual IEEE references.

Your main method will accept the name of an input file and then of an output file.

The input file is a text file that contains the text of a research paper. In that file, you can find citations in the form of either \cite{aaa} or \cite{aaa, bbb, ccc} that indicate that one or multiple articles are being cited at the given location and whose article keys are aaa, bbb, and ccc.

You will transform these citations into IEEE references. So, \cite{aaa} will become [1] in the output text (assuming that paper aaa is the first item in the reference list) and you will append text to the whole paper with [1] as having the IEEE formatted reference entry for paper aaa. In making that IEEE reference entry, you can assume that all but the last string in an author's name is a first name to be abbreviated and that you just pick the first letter of all the first names (plus the full last name) in the IEEE author list.

An entry of \cite{aaa, bbb, ccc} is then output as [1, 2, 3], assuming that article aaa is reference [1], article bbb is reference [2], and article ccc is reference [3].

Output the resulting article, with the appended references to the given output file.

Assumptions

- Author full names will be unique, will not be hyphenated names, and are case insensitive.
- We don't remove data from the library.
- The phrase \cite will not appear as regular text in the article. Any notion of \cite will be an actual reference.
- Any \cite entry will not span multiple rows in the input file.
- All \cite entries will be properly opened and closed with {} brackets and will not be nested.

Notes

- Include the type of exception handling that you feel is most appropriate for your circumstances. Document that exception handling mode and be consistent with it.

Milestones

Only the final submission is graded. There is an intermediate milestone for April 13:

- April 13 23: Blackbox tests and external documentation of data structures, code design, and key algorithms.

The milestone material is due in the project git repository, in the docs folder.

Marking scheme

- Meeting intermediate milestones (10%)
- Documentation (10%)
- Overall design and coding style (15%)
    - Design, including the quality and consistency of the decisions being made in the design 10%
    - Coding style and proper use of version control to do a multi-stepped implementation 5%
- Working implementation, including efficient processing (50%)
    - Building the database 5%
    - Reporting methods 15%
    - Conversion of a file to include references 20%
    - Robustness and error reporting 5%
    - Efficiency 5%
- Final test plan (15%)