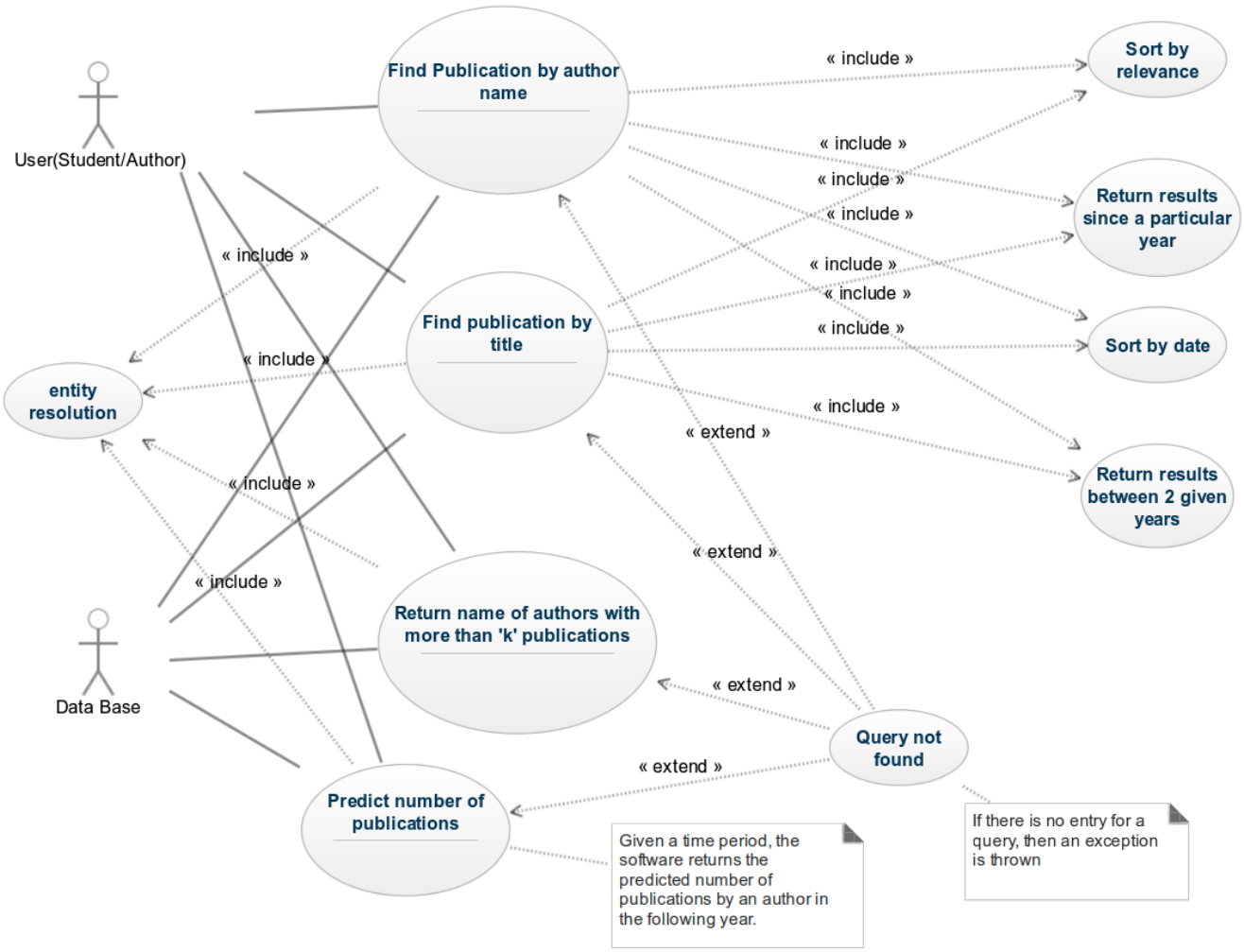
Lab-7
DBLP Project Report
-Vedant Nanda(2015114)
-Arpan Mondal(2015132)

Use Case Diagram:



Use Case Descriptions:

1) **Name:** Entity Resolution
   Entry condition: The database has been loaded onto the memory.
   Exit condition: Authors with different aliases have been grouped together as one.
   Event flow:
   1. The whole database(dblp.xml) is traversed.
   2. For every author name we check whether it is similar any existing author names using the <www> tag.
   3. If yes then that object is added to an arraylist containing all the aliases.
   4. If not a new arraylist is created.
   Exceptions: None

2) **Name:** Finding publications by author name
   Participating actor: Database, User.
   Entry condition: The database has been loaded onto the memory.

Exit condition: All publications have for a given author name have been found.
Event flow:
1. Entity resolution is done on the whole database.
2. For a given author name publications corresponding to the cluster of that author are separated and returned.
Exceptions: No author matching the input name exists.

3) **Name:** Finding publications by title
Participating actor: Database, User.
Entry condition: The database has been loaded onto the memory.
Exit condition: All publications corresponding to the given title have been found
Event flow:
For a given title, publications matching the given title are retrieved and returned.
Exceptions: No publication matching the input title exists.

4) **Name:** Finding authors with more than K publications
Participating actor: Database, User.
Entry condition: The database has been loaded onto the memory.
Exit condition: All authors having more than k publications have been found
Event flow:
For a given value of k author names of authors having more than k publications are returned.
Exceptions: No author has more than k publications or k is not an integer.

5) **Name:** Predicting number of publications for an author given publications till a specified year
Participating actor: Database, User.
Entry condition: The database has been loaded onto the memory.
Exit condition: The number of publications for the next year have been predicted based on the data given till a particular year.
Event flow:
For a given author, using the data up to a particular year we predict the number of publications made by that author in the next year based on some probabilistic estimations.
Exceptions: The author entered does not exist.

6) **Name:** Query not found/Inappropriate query
Participating actor: Database,User
Entry condition: The database has been loaded onto the memory and user enters a query to be searched.
Exit condition: Search query returned no results.
Event flow:
For a given input, all records have been searched and no record matched the input query.

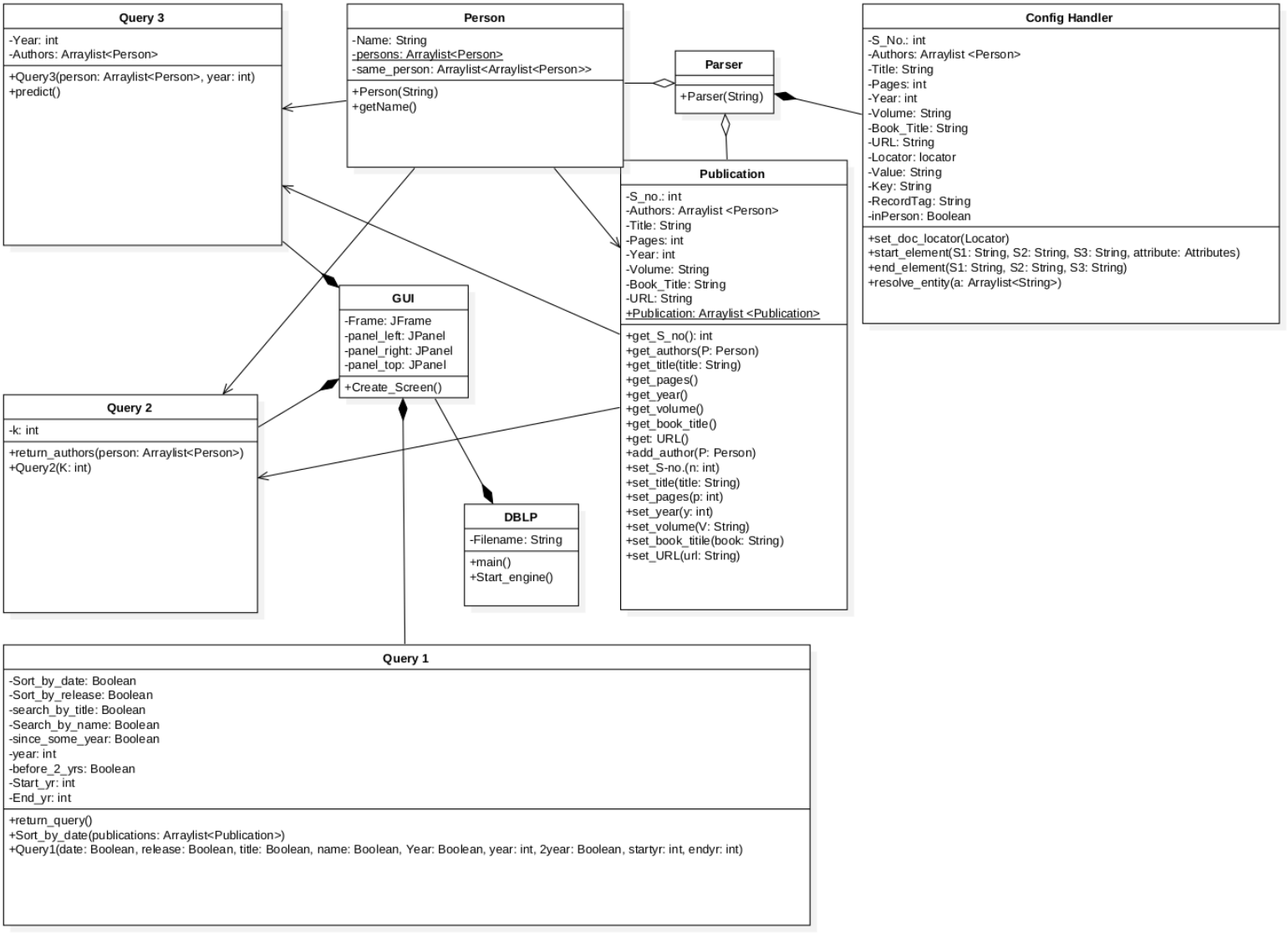
7) **Name:** Sort by relevance
Participating actor: Database
Entry condition: The database has been loaded onto the memory and user enters a query to be searched.
Exit condition: Search query returned no exception and the returned results have been sorted by relevance.
Event flow:
For a given input, all records have been searched and the matching results are sorted by their relevance.

8) **Name:** Sort by date
   Participating actor: Database
   Entry condition: The database has been loaded onto the memory and user enters a query to be searched.
   Exit condition: Search query returned no exception and the returned results have been sorted by date.
   Event flow:
   For a given input, all records have been searched and the matching results have been sorted by date.

9) **Name:** Return results since a particular year
   Participating actor: Database
   Entry condition: The database has been loaded onto the memory and user enters a query to be searched.
   Exit condition: Search query returned no exception and the returned results have been filtered since after the given year.
   Event flow:
   For a given input, all records have been searched and the matching results have been filtered out based on given year.

10) **Name:** Return results between 2 years
    Participating actor: Database
    Entry condition: The database has been loaded onto the memory and user enters a query to be searched.
    Exit condition: Search query returned no exception and the returned results have been filtered to be between 2 particular years.
    Event flow:
    For a given input, all records have been searched and the matching results have been filtered out based on given years.

Class Diagram:



Explaination:

- **DBLP**
  The main class that starts and initiates everything. It has a main function which calls the function start_engine(). This functions starts the DBLP application by creating an object of GUI class.
- **GUI**
  This class has create_screen() function which creates a Jframe and adds Jpanels to the GUI. Based on the actions on these elements, it accordingly creates objects of Query1, Query2 or Query3.
- **Query1**
  The constructor sets the boolean variables. Based on these variables' values, the function return_query() returns an ArrayList of Publications. There is another function sort_by_date() which sorts and returns a given arraylist of Publications based on date.
- **Query2**
  Given k, the function return_author() reurns an arraylist of the authors with more than k pubications.

- **Query3**
  An arraylist of authors is initialized which has the names of authors for whom we need to predict. The function predict() returns an integer arraylist which is the predicted number of publications for each author entry in the corresponding author arraylist.
- **Person**
  This class represents an author. A static arraylist of Person contains all the authors in the database. The static variable same_person is an arraylist of arraylist which contains all authors that are same but have different names in different places.
- **Publication**
  This class represents every publication in the database.It has all the attributes needed to display a publication as private variables and their corresponding setters and getters as public functions. The static variable publications is an arraylist of all publications in the database.
- **Parser**
  This is the API endpoint which takes a string filename as input and parses the xml and stores the formatted data in the arraylists persons,same_person and publications.
- **ConfigHandler**
  This is an inner class of Parser. The functions start_element() and end_element() specify the opening and closing tags, data between these tags is stored in the appropriate class variables. The function resolve_entity() looks for the tag<www> and puts all authors with the same name in an arraylist which is added to the same_person arraylist(thus making same_person an arraylist of arraylists).