CS1003 W09 Practical 160026526 Tutor: Ruth Letham 1 April 2018

• Overview:
The aim of the practical was to write a Java program which is able to interact with search API of DBLP, a service which prides extensive information on major computer science publications. The program should be able to take 3 values on the command line interface which are search, query and cache. Search should take either author, publication or venue, query should take a string which will be used to execute the search and cache should take a path to a cache directory. The ordering for the 3 values should be interchangeable as long as their corresponding values do not change order. Error messages should also be produced in a case where the command line arguments are invalid. The program should print specific outputs for each search query based on the specification and the queries should be modified to follow the specified searches. Each of the XML responses should be saved in a cache file and an encoder should be used to encode the URL into a file name. The program should always check its cache before making a request to the server to avoid repetition.

• Design:
Creating the searches
To begin the practical, it is clear that the search will require either venue, publication or author to specify the type of search. With this in mind, three Java classes are created which are called SearchVenue, SearchPublication and SearchAuthor. The coding for each of the classes should be fairly similar when creating the XML files. SearchVenue is created first as it is the simplest out of the three in which it only requires the printing of the venue name.

Creating the SearchVenue/SearchPublication/SearchAuthor classes
A search method is created which initially took an int search, int query and Path cache. String[] args was added later in which was used to get arguments from the commandline. When the method was first created, the args[query] had spaces in the search which was likely to disrupt the search and thus the spaces were replacd with “%20” to follow DBLP’s search API. The method then creates a new URL which takes a string combination of “http://dblp.org/search/” with the args[search], the API query and “&h=40&c=0” which takes 40 hits and set the maximum number of completion terms to 0 as according to the specification. A DocumentBuilderFactory dbf is created which is used to create the DocumentBuilder db. A document is then created by parsing the URL to the document builder. Since the file is required in XML, a TransformerFactory tf is used to create the Transformer t. The URL is then encoded to convert the URL to a valid file name using the URLEncoder with the encoding being UTF-8 as it supports many languages. A File called apiCall is then created using the cache Path along with encodeURL and “.xml” which creates the xml file in the cache file. DOMSource is then used to create an input source from XML input and StreamResult is used to create a stream result from the file. The Transformer t is then used to transform the XML input into the file. This entire sequence is also used when constructing the SearchPublication and SearchAuthor XML files. Since these are similar, they are condensed into another method file called Settings.

Creating the Settings file
String[] args, Document doc and URL url are privated at the start as specific method will be made to individually get them. The private prevents them from being changed directly from other methods. The makeCache method is created first. It takes a File apiCall and throws two exceptions. This method will be used to make the cache file and thus a TransformerFactory and Transformer are created. To transform a source and result are required and as such the DOMSource will be created on a XML document called doc and a FileWriter is written for the apiCall file. The result will be a file so StreamResult uses the file created by FileWriter. Finally, the transformer transforms the

source to the result. Moving on a method will be required to create the document and this is called buildDoc. This takes a String[] args, int search, int query and Path cache and throws exceptions. The code from creating the searches is then added here, specifically the replacement of whitespaces, creating of URL, encoding of URL and creating of the apiCall file. Since the program should always check with the cache file, an if statement is created that checks if the apiCall exists and if it does it will use an overloaded buildDoc method. The method then makes a DocumentBuilderFactory and a DocumentBuilder and a document is made by parsing the URL. The apiCall is then used as a File for the makeCache method. Moving on to the overloaded buildDoc method, the method takes a apiCall file and parses it into the DocumentBuilder. By doing this, the program checks if the file exists before doing the DBLP search. A getDoc and getUrl methods are made in the event the document or URL is required.

Moving back to SearchVenue

A try catch block is created in the search method which first creates a new settings object and document and cache file are created using the buildDoc method. A NodeList venue is constructed which get elements by the tag name “venue” from the document created previously. To find the names of each venue, a for loop is made which prints each name of the venues. The catches then catch exceptions.

SearchPublication

A try catch block is created and a new settings object is created which is used to buildDoc. After reading the publication search it can be seen that the parent node is info and thus a NodeList info is created using getElementsByTagName. A for loop is then created and individual nodes are extracted from info and the element nodes are called element. Since the task requires the title and the number of authors, the program will print the names by searching by the tag name “title” under elements and print the number of authors by first searching by tag name “authors” under elements and getting the length. The catches then catch exceptions.

SearchAuthor

A try catch block is created and a new settings object is created which is used to buildDoc. After reading the publication search it can be seen that the parent node is info and thus a NodeList info is created using getElementsByTagName. A for loop is then created and individual nodes are extracted from info and the element nodes are called element. A second API call is required for this as URL’s in the XML file are to be used to redirect to a new URL to find total publications and number of co-authors. A new URL is created which takes the URL’s from the elements and a the same process from “Creating the SearchVenue/SearchPublication/SearchAuthor classes”. A person NodeList is created from the elements by tag name “dblpperson” . A second for loop is create and the Nodes for node2 are from from the person NodeList and the element nodes are found. The authors names are printed from the element after getting by the tag name “author”. To find the number of publications, elements from the element2 are counted using getLength after the elements are found by the tag name “title”. To find the number of co-authors, elements from the element2 are counted using getLength after the elements are found by the tag name “co”. The catches then catch exceptions.

Writing the main method

To begin, a for loop is created to loop around the argument length. The first thing to check are the corresponding values as this helps keep the array from going out of bounds. This is done with and if statement and it checks if arg[i] does contains “--”. A switch case is made inside the if statement and if args[i-1] is used as it gives us either “--search”, “--query” or “--cache”. The first case is if args[i-1] is “--search” and thus we set search = i and initialise search = 0 at the beginning. The second case is if args[i-1] is “--query” and thus we set query = i and initialise query = 0 at the beginning. The

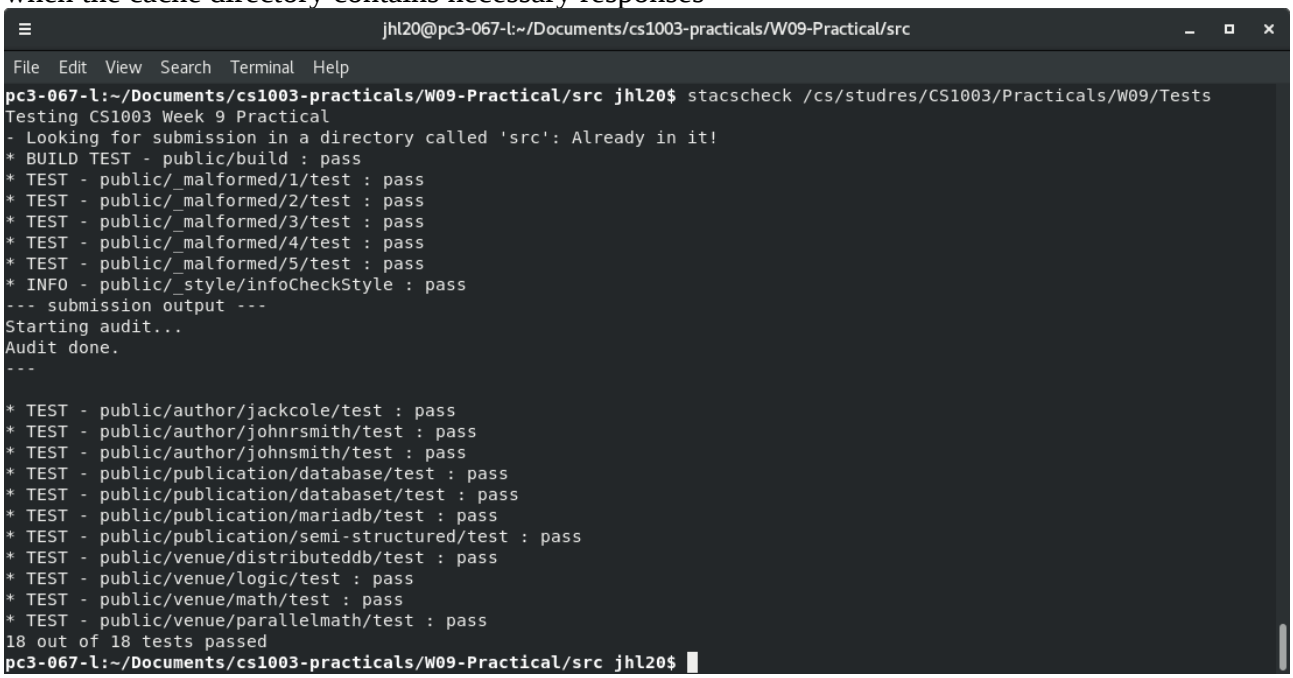
third case is if args[i-1] is “--cache”. A default case is used to print an error message and exit the system. A second if statement is written and this checks for when args[i] equals “--cache” and if it does then a Path cache is initialised as null in the beginning and the cache will be set to the Path.get of args[i+1] which is the corresponding value of “--cache” and the if loop and for loop are ended. Now errors have to be taken into consideration and a if statement checks if query equals 0 and if it does then an error message is printed and the program ends. The second if statements checks if search = 0 and if it does then an error message is printed and the program ends. The finak is statement checks if the cache file does not exists and if it does not exist then an error message is printed saying that the directory doesn’t exist and the program terminates. A public static final int PUBL = 4 is written before the main method and this will be explained in the next section.

Implementing the searches into the main method

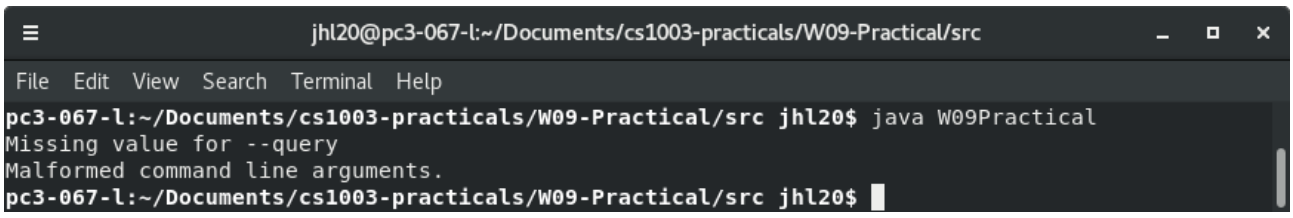
To do this, a switch case is created that checks the args[search]. For the first case, if it is venue, then a new object for SearchVenue, sv, is created. Then the method search is ran. If args[search] is publication then the second case runs and we first change the get a substring of args[search] using PUBL and this is used to shorten publication to publ as that is how DBLP has sorted their publications. A new object for SearchPublications sp is created and the search method is ran. Finally when args[search] is author then the third case runs and an object SearchAuthor sa is created and the search method is used. Finally a default is added for good practice.

• Testing:

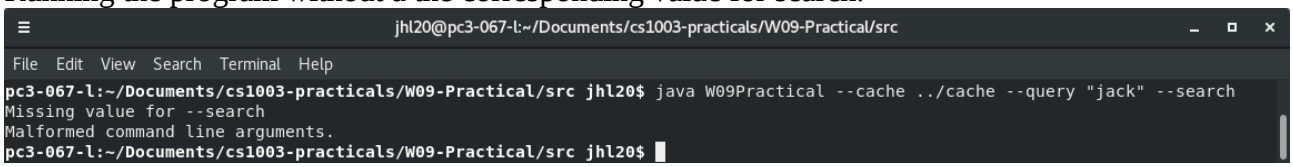
The first test to ensure that the program works according to the specification is the autochecker which is provided in the specification. The program passed all 18 of the autochecker tests. This is when the cache directory contains necessary responses



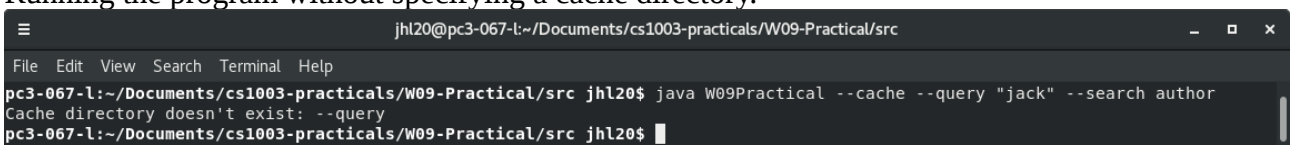
Running the program without any arguments.



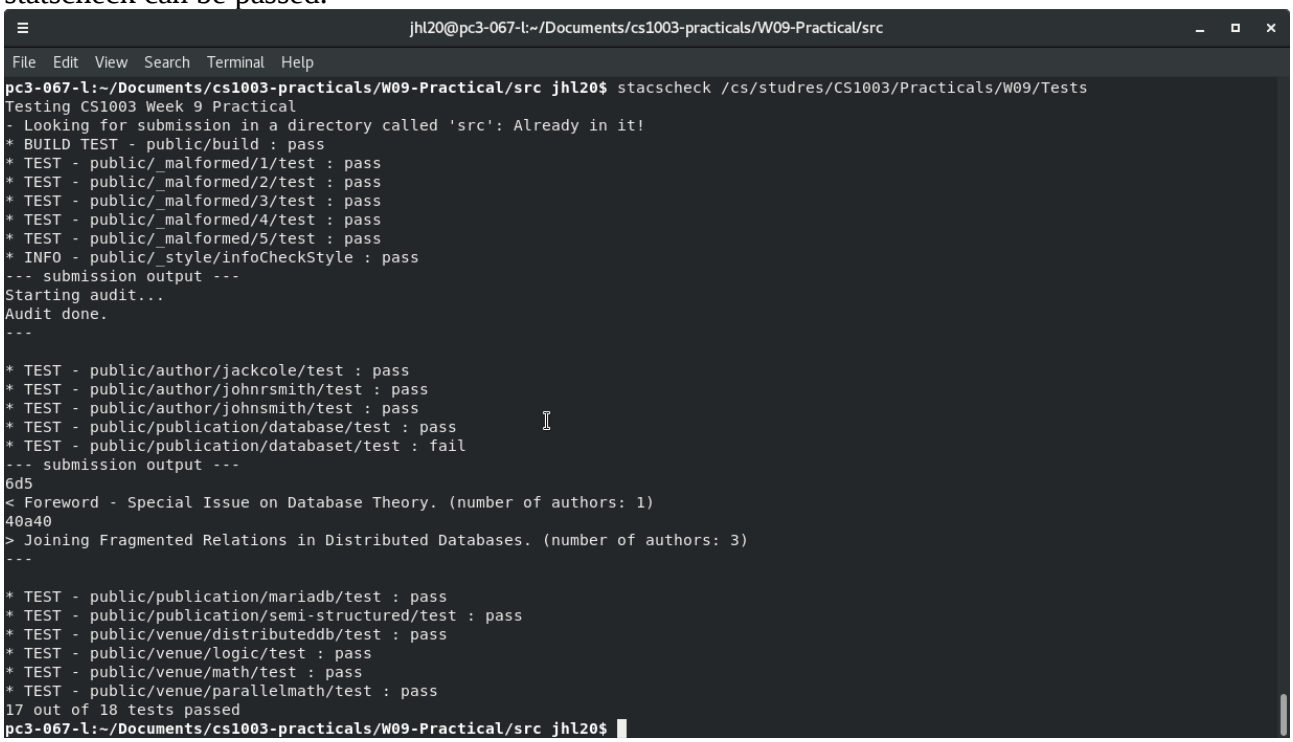
Running the program without a the corresponding value for search.



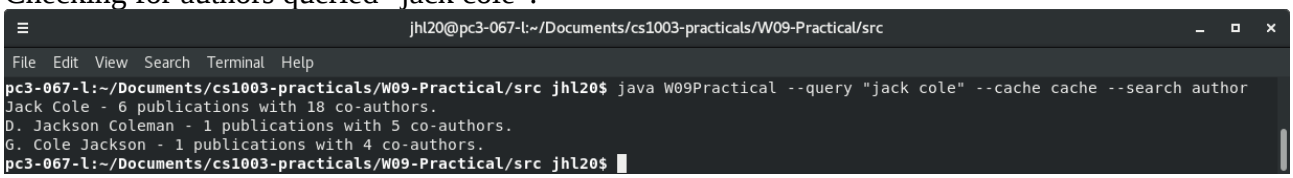
Running the program without specifying a cache directory.



At the time of writing the report, the DBLP website has changed and only 17 of the 18 checks from statscheck can be passed.



Checking for authors queried “jack cole”.



Checking for authors queried “jack”.

```
jhl20@pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src
File  Edit  View  Search  Terminal  Help
pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src jhl20$ java W09Practical --cache cache --query "jack" --search author
Jack Aaronson - 1 publications with 0 co-authors.
Jack Abad - 1 publications with 3 co-authors.
Jackson Abascal - 2 publications with 1 co-authors.
Roselyn Abbiw-Jackson - 2 publications with 3 co-authors.
Daniel Abbw-Jackson - 1 publications with 2 co-authors.
Jackson Adams - 1 publications with 1 co-authors.
Jack R. Adams-Webber - 3 publications with 6 co-authors.
Jack Addis - 1 publications with 5 co-authors.
Jack F. Adolph - 1 publications with 2 co-authors.
Jackeline Agorreta - 1 publications with 6 co-authors.
Jacob Jacky Aharon - 1 publications with 4 co-authors.
Jack Aiston - 2 publications with 2 co-authors.
Jacky Akoka - 66 publications with 47 co-authors.
Jackson Akpojaro - 1 publications with 2 co-authors.
Jack Aldrich - 1 publications with 8 co-authors.
Jack B. Aldrich - 1 publications with 2 co-authors.
Jack A. Alexander-Webber - 1 publications with 12 co-authors.
Jack A. Alford Jr. - 1 publications with 1 co-authors.
Jack Allen - 2 publications with 38 co-authors.
Jack Andersen - 8 publications with 4 co-authors.
Jack B. Andersen - 1 publications with 4 co-authors.
Jack Bukkoldt Andersen - 1 publications with 2 co-authors.
Jack Anderson - 1 publications with 3 co-authors.
Jackie Andrade - 1 publications with 1 co-authors.
Jackie Antig - 1 publications with 3 co-authors.
Jack H. Arabian - 5 publications with 6 co-authors.
Jackson Archer - 1 publications with 6 co-authors.
Jackie Archibald - 2 publications with 5 co-authors.
Jack Armitage - 2 publications with 7 co-authors.
Jackson Armstrong - 1 publications with 3 co-authors.
Jack Asavanant - 3 publications with 10 co-authors.
Jack Ashby - 1 publications with 5 co-authors.
Jackie Assa - 9 publications with 20 co-authors.
Jack Atherton - 2 publications with 2 co-authors.
Jack Atkin - 1 publications with 1 co-authors.
Jacky Au - 2 publications with 8 co-authors.
Jack S. Avrin - 1 publications with 0 co-authors.
Jack Baczynski - 10 publications with 9 co-authors.
Jack Baker - 1 publications with 3 co-authors.
Jack W. Baker - 1 publications with 1 co-authors.
pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src jhl20$
```

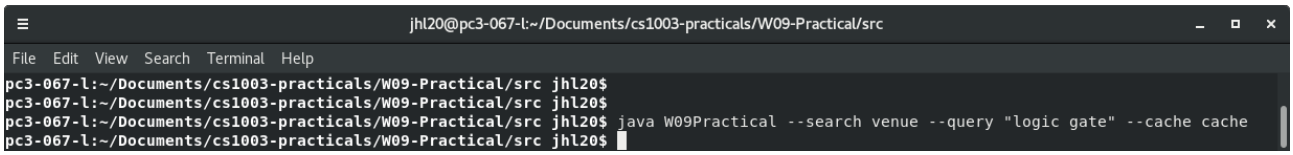
Checking for publications queried “mariadb”.

```
jhl20@pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src
File  Edit  View  Search  Terminal  Help
pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src jhl20$ java W09Practical --search publication --query "mariadb" --cache cache
An NVM Aware MariaDB Database System and Associated IO Workload on File Systems. (number of authors: 7)
Full automated continuous integration and testing infrastructure for Maxscale and MariaDB. (number of authors: 6)
NVM aware MariaDB database system. (number of authors: 5)
pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src jhl20$
```

Checking for venue queried “logic”.

```
jhl20@pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src
File  Edit  View  Search  Terminal  Help
pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src jhl20$ java W09Practical --search venue --query "logic" --cache cache
Logic in Databases (LID)
Workshop on Logic Programming Environments (WLPE)
Logics of Programs
Learning Language in Logic
Logical and Computational Complexity (LCC)
Logic at Work
Logic, Action, and Information
Logic and Machines
Annual Conference for Computer Science Logic (CSL)
International Symposium on Games, Automata, Logics and Formal Verification (GandALF)
International Conference on Theorem Proving in Higher Order Logics (TPHOLs)
International Workshop on Meta-Programming in Logic (META)
Fuji International Symposium on Functional and Logic Programming (FLOPS)
Logic of Theory Change
Logical Aspects of Computational Linguistics (LACL)
European Society for Fuzzy Logic and Technology (EUSFLAT)
Workshop on Rewriting Logic and Its Applications (WRLA)
ERCIM Annual Workshop on Constraint Solving and Constraint Logic Programming (CSCLP)
Temporal Logic in Specification
Conférence francophone sur les Architectures Logicielles (CAL)
International Conference on Topology, Algebra and Categories in Logic (TACL)
Constraints in Computational Logics (CCL)
International Conference on Logic Programming and Non-Monotonic Reasoning (LPNMR)
International Conference on Algebraic and Logic Programming (ALP)
Logic in Computer Science (LICS)
Joint International Conference and Symposium on Logic Programming (JICSLP)
Advances in Modal Logic (AIML)
International Workshop on Fuzzy Logic and Applications (WILF)
Workshop on Curry and Functional Logic Programming (WCFLP)
International Workshop on Classical Logic and Computation (CL&C)
Logic Programming and Automated Reasoning (LPAR)
International Workshop on the Implementation of Logics (IWIL @ LPAR)
Logic Programming and Knowledge Representation (LPKR)
International Workshop on Quantum Physics and Logic (QPL)
Workshop on Logics for Intelligent Agents and Multi-agent Systems (WLIAMAS @ IAT)
Logic Programming Summer School (LPSS)
Symposium on Programming Language Implementation and Logic Programming (PLILP)
APPIA-GULP-PRODE Joint Conference on Declarative Programming (AGP)
International Conference/Workshop on Inductive Logic Programming (ILP)
Infinity in Logic and Computation (ILC)
pc3-067-l:~/Documents/cs1003-practicals/W09-Practical/src jhl20$
```

Checking for venue queried “logic gate”. There should be none.



• Evaluation:
In this practical, the Java program was required to interact with DBLP. The program is suppose to take 3 values which are “--search”, “--query” and “--cache” along with the corresponding values which makes a total of 6 arguments required. The 3 main arguments should also be interchangeable as long as the corresponding arguments stay with them. The program should print the names of venue when the search is venue, print title and number of authors when the search is publication and name of author, number of publications and number of co-authors when the search is author. A cache file should also store all the previously searched queries as XML files and the program should check with these cache files before proceeding to make a request from the DBLP server. The program should also be able to pass the autochecker without flows. The program successfully carried out the requirements set by the practical specification and therefore is deemed as successful. However, some information on DBLP has changed over the last few days and thus when the autochecker is ran only 17 of the 18 tests pass. Although this is the case, when using the old cache files, the program passes all 18 of the autochecker tests and therefore works.

• Conclusion:
In conclusion, the program was successful in interacting with DBLP and was able to take in different combinations of commandline arguments. The program also was able to deal with cache files and checks before requesting from the server. A few of the difficulties came when implementing the search classes into the W09Practical class but were eventually fixed as arguments were not used at first which caused the program to fail. Other than that, due to the duplication of code in all three search classes a fourth class called settings was used to combine the similar methods to reduce the clutter. None of the extensions were attempted as using the JSON format would have proved to be a greater challenged as it is unfamiliar. With more time, it could have been attempted. Method overloading also proved to be slightly challenging and old notes had to be revisited.