CSC 207 Assignment3
Due date: 30th November 2015 @ 11:59pm

1. Introduction

Welcome to the final CSC 207 programming assignment. You are free to use any IDE that you wish. This assignment is intended to help you practice basic Java Programming skills, navigate a bit of Java API and understand regular expressions better.

2. Problem Statement

The main problem statement is more of a text analysis and you will learn how to read the contents of a webpage by building a Java application. The assignment requires you to write a program, which reads and analyzes the contents of the Google Scholar Page of an author.

Google Scholar is a freely accessible web search engine that indexes the full text of scholarly literature across an array of publishing formats and disciplines. For this assignment, you need to extract information from the Google Scholar publication page of a number of authors. The format of the link towards the Google Scholar publication page of an author would be like:

http://scholar.google.ca/citations?user=X

where X is the user id. You can search for different authors' names and replace this field with their id.

HTML: When you read a webpage, you are actually reading its HTML contents. If you do not know anything about HTML, this wiki page can help you. You don't need to know HTML in order to complete this assignment.

In this assignment, you need to build a web scrapper for Google Scholar pages. Your program will read a list of Google Scholar URL'S, extract specific information and return the following information. At the end of the handout you can find a screenshot *(see last page of this handout)* with the location of all these information in a Google Scholar publication page of an Author:

i. Name of the Author.

ii. Number of All Citations.

iii. Number of i10-index after 2009.

iv. The title of the first three publications.

v. The total number of citations of the first five publications.

vi. Number of co-authors on the right side bar. The only essential information needed is the number of all of the co-authors as they appear on the right side bar. You should not follow the URL under "View all co-authors" link. Zero co-authors are also acceptable.

vii. The total number of co-authors and all the co-authors' names in alphabetical order – one name per line. Co-authors from each Author should be stored in a data structure (check Java Collection). Every time your program finds a co-author, the name of the co-author should be added in your data structure. You should avoid duplicates. If the name of the co-author already exists in your structure, you should not add it again. After visiting all the links from the input file, your program should print the total number of co-authors followed by a list of all the co-authors name in alphabetical order. You should print one co-authors name per line.

The program should receive two arguments at its command line:

- The first argument, which is mandatory, is a single string of HTML files, where a comma separates each separate HTML file. The entire argument is surrounded by double quotes.

  Each Google Scholar URL is of the form:
  http://scholar.google.ca/citations?user=X

  However for this assignment you MUST use the local copies of HTML pages that have been provided to you in your SVN repos. This is because the actual HTML pages on the web, change quite often and this will also break your unit tests. You are required to make your assignment work only with the syntactical structure of the local files provided. **You do not need to make your assignment work with live Google web pages.**

- The second argument, which is optional, is a filename to which the program output will be sent. If this argument was not specified, the program output will be sent to the standard system console.

Notes:

- You have to use the Java Pattern and Matcher class for this assignment.
- You also need a Java Collection.
- You will not use any libraries for parsing etc. You have to implement this functionality through regular expression in your code.
- You will find the following links useful as you create your regular expression queries.
  http://www.regexplanet.com/advanced/java/index.html
  http://regexpal.com/

Task1:

a) Check out your Assignment3 from your repository. You are free to create any additional classes (i.e. separate java files) if you wish, but all your source files will reside in the src folder in appropriate packages and your tests inside the test package.

Your SVN URL is of the following form:

- https://142.1.44.22/svn/csc207h/UTORID/207Assignment3
  Where UTORID is your actual UTORID.

- Instructions to checkout Assignment3 is the same as all your previous Assignment(s).
  You may want to refer back to Assignment0 slides on how to do this.

b) If you are using Eclipse, you will need to change the Run Configuration as follows:

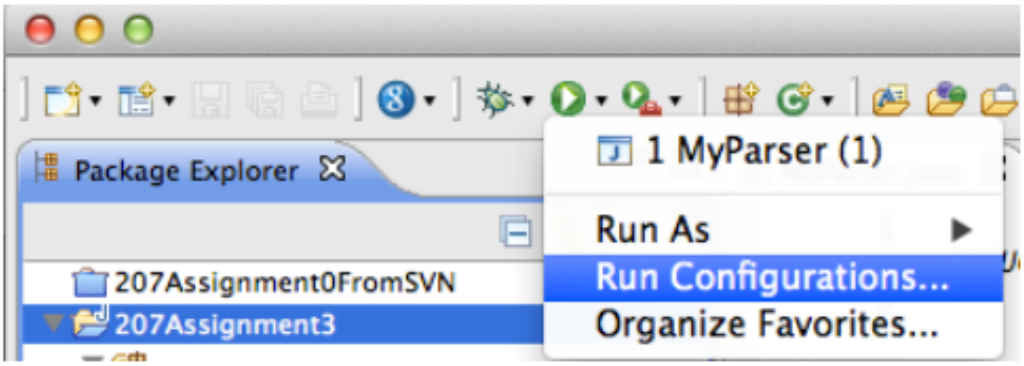


Figure1: First open up the run configuration as seen in the above screenshot.

Figure 2: Open the Arguments tab, and add the two arguments (each argument is separated by a white space): 1) list of local HTML where each file is separated by a comma. 2) Name of the output
file (in the above screenshot it is called outFile, this argument is optional).

c) Without making any change to the source code, when you run your project now from Eclipse, you should see the following output.

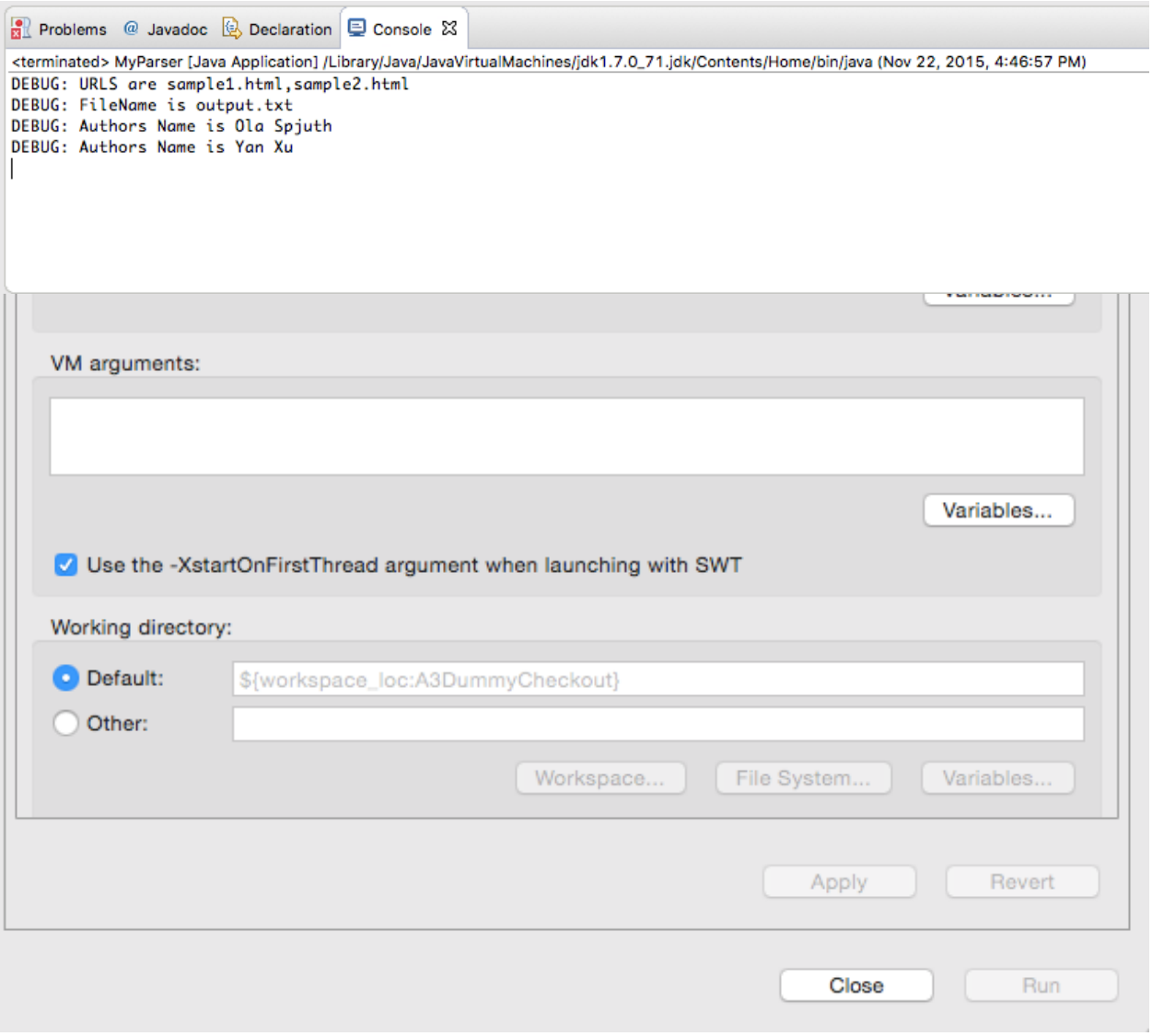


Figure 3: Output of the starter code that I have provided in the SVN repo.

Task 2:
Write your Java HTML Parser program. Your main function should be in a MyParser.java file. You can have other .java files as well, depending on your design. Write the code and regularly commit the changes to the repository. Upon your progress commit ONLY the source code of your application to your repository (this will ensure that you don't commit any IDE files).

You will need to make sure that your program is following the SRP principle across your entire design. This implies that you will create many more classes (as per SRP principle).

Task 3 (optional and Bonus) (20%):

1. (10%) Prepare a build.xml that can be executed with Apache Ant. If you attempt this, your build.xml file must reside inside Assignment3 folder. You only need to create a build.xml file for this. **Do not commit anything else besides this file in your Assignment3 folder**. You may want to refer to the workshop material on Apache Ant (material on Blackboard). Your build.xml must create the bin folder (same location as where src is). The bin folder will contain all the .class files of the .java files (this also includes the test files). The default target when called from Apache Ant, should perform all the above steps and then execute the main function inside the MyParser.java by passing in sample1.html and sample2.html as command line arguments. You can also safely assume that we will have the required junit.jar and hamcreast-core.jar file injected in your Assignment3 folder during testing. YOU MUST ENSURE THAT YOU ARE REFERRING TO THE ABOVE JUNIT JAR FILES WITH THE ABOVE NAMES ONLY. **You will only receive the bonus marks if everything works as expected when executed on the cslinux machine, otherwise there is no partial mark.**

2. (10%). Make your parser also work with live Google Pages. All live Google pages will begin with https://scholar.google.ca/citations?user=… (as an example here is one sample page https://scholar.google.ca/citations?user=oSJgw2cAAAAJ&hl=en&oi=ao). If you attempt this bonus section, then keep in mind that when we pass in https://… as arguments instead of sample1.html, then this implies your code must work with live Google pages. At any given time you can safely assume that all the arguments are either live Google pages OR local html pages (format of the html page is similar to the sample files provided). Also, the live pages are very different from the local sample provided. This implies that your regular expression for the live pages will be different from the local html pages. You will only receive the bonus marks if everything works as expected when executed on the cslinux machine, otherwise there is no partial mark.

If you attempt any of the bonus section, you must submit a bonus.txt file explaining if you are attempting both the bonus section or any one of them. If this file is not present, we will assume you have not attempted any bonus section and the auto marker will ignore the bonus marking.

Marking:

See Blackboard for the marking scheme.

==========

You can use *Eclipse* OR *command line* to compile and run your assignment. I am providing two examples. Example1 is using Eclipse without the outputFile argument AND example2 is using command line that includes outputFile argument.

Example1:

When your complete program is executed from Eclipse (as seen in Figure 2) or from the command line with no output file parameter, the following will be printed out on **stdout** of your console. For your convenience the exact output is also provided on Blackboard. (See output.txt)

---

1. Name of Author:
Yan Xu
2. Number of All Citations:
263
3. Number of i10-index after 2009:
9
4. Title of the first 3 publications:
1- Face-tracking as an augmented input in video games: enhancing presence, role-playing and control
2- Art of defense: a collaborative handheld augmented reality board game
3- Sociable killers: understanding social relationships in an online first-person shooter game
5. Total paper citation (first 5 papers):
158
6. Total Co-Authors:
14

---

1. Name of Author:
Ola Spjuth
2. Number of All Citations:
437
3. Number of i10-index after 2009:
12
4. Title of the first 3 publications:
1- Bioclipse: an open source workbench for chemo-and bioinformatics
2- The LCB data warehouse
3- XMPP for cloud computing in bioinformatics supporting discovery and invocation of asynchronous web services
5. Total paper citation (first 5 papers):
239
6. Total Co-Authors:
15

---

7. Co-Author list sorted (Total: 29):
Abigail Sellen
Adam Ameur
Andrew D Miller
Antony John Williams
Blair MacIntyre
Christoph Steinbeck
Deepak Jagdish
E.D. Mynatt
Egon Willighagen
Elsa Eiriksdottir
Erika Shehan Poole
Greg Turk
Iulian Radu
Janna Hastings

John Stasko
Jonathan Alvarsson
Komorowski Jan
Kurt Luther
Nina Jeliazkova
Noel M. O'Boyle
Rajarshi Guha
Sam Adams
Samuel Lampa
Sean Ekins
Thore Graepel
Valentin Georgiev
Xiang Cao
Youn-ah Kang
gilleain torrance

Example2 (using command line):

If you were to specify the `outfile` argument (See how this is done in Eclipse in Figure2), then the entire output of your program that was earlier generated on stdout, will now be written to `outfile`.

First compile the program:

To compile Assignment3 (starter code, as provided by the instructor) this is what you do:

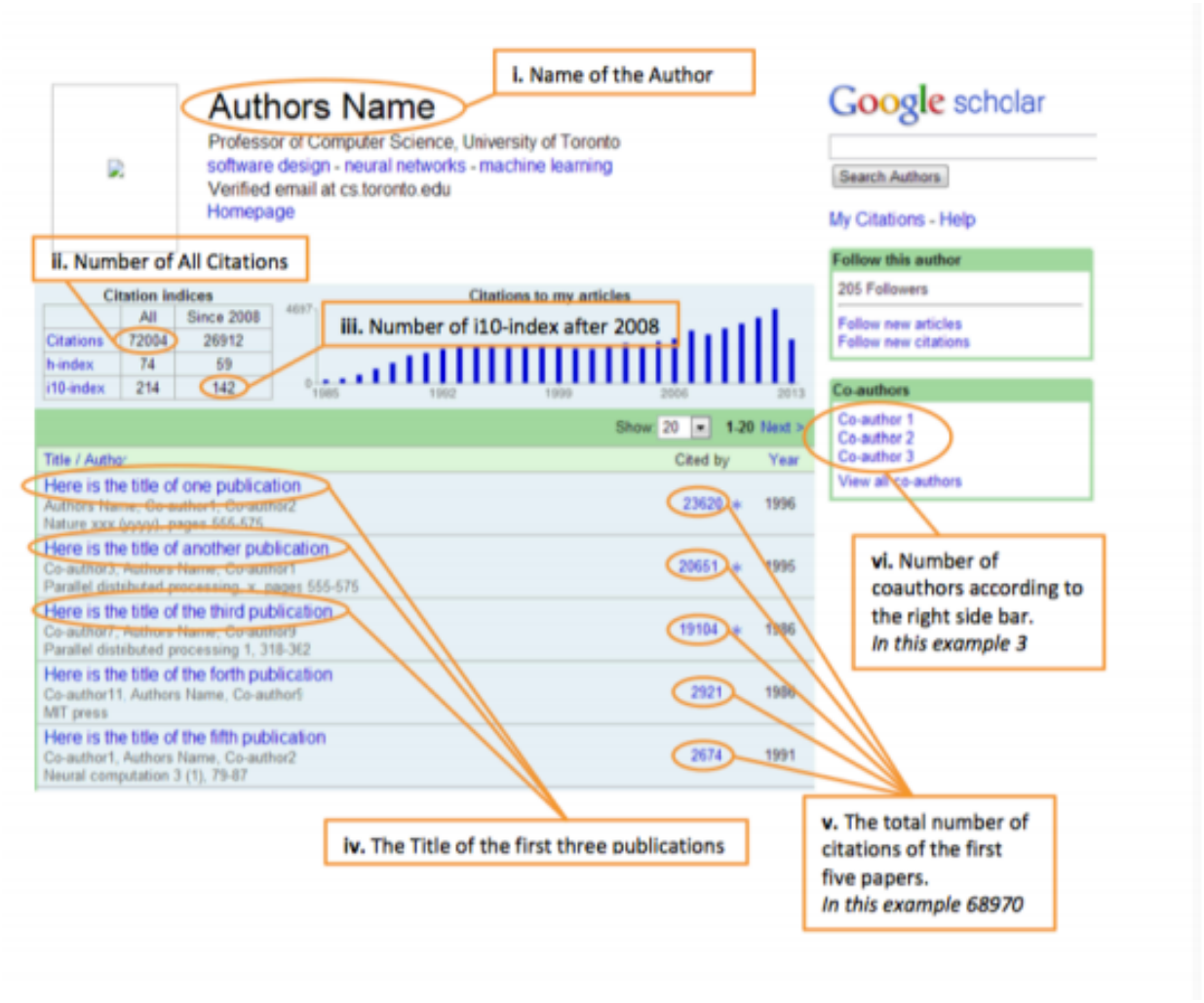
```
javac -d bin/ src/driver/MyParser.java
```

—Assuming that you have created bin folder already before running the above program.
—Assuming that the src folder is in the current working directory.
—Assuming that there is only one .java file i.e. MyParser.java to compile. If you have have more .java files, you will enter them accordingly.
—The javac command stands for java compiler.

Second, run the program

```
java -cp bin/ driver.MyParser sample1.html,sample2.html outfile
```

—The -cp option basically stands for class path and all it means is class path. It points to the directory where you class files that were compiled from the javac command resides.
—The argument driver.MyParser is the class that happens to have the main function in order for the program to run.
—At this point your file `outfile` will be populated with the content. There will be no output on stdout or on the console.


i. Name of the Author
Authors Name
Professor of Computer Science, University of Toronto
software design - neural networks - machine learning
Verified email at cs.toronto.edu
Homepage
ii. Number of All Citations
Citation indices
All
Since 2008
Citations
72004
26912
h-index
74
59
i10-index
214
142
iii. Number of i10-index after 2008
Citations to my articles
Google scholar
Search Authors
My Citations - Help
Follow this author
205 Followers
Follow new articles
Follow new citations
Co-authors
Co-author 1
Co-author 2
Co-author 3
View all co-authors
vi. Number of coauthors according to the right side bar. In this example 3
Show: 20
1-20 Next >
Title / Autho:
Cited by
Year
Here is the title of one publication
Here is the title of another publication
Here is the title of the third publication
Here is the title of the forth publication
Co-author11, Authors Name, Co-author1
MIT press
Here is the title of the fifth publication
Co-author1, Authors Name, Co-author2
Neural computation 3 (1), 79-87
23620
20651
19104
2921
2674
1996
1995
1986
1990
1991
iv. The Title of the first three publications
v. The total number of citations of the first five papers. In this example 68970