COMP 1406Z
Fall 2022 – Course Project

Project Background

In the 1405 portion of the course, you implemented a web crawler and search engine in Python. One of the main goals for the 1406 project is to re-implement this functionality in Java using object-oriented programing. The class design and overall implementation details for the 1406 project will be your responsibility. Along with the crawl and search functionality from the 1405 project, your 1406 project must also include:

1. A GUI application that allows the user to perform searches and see search results. This must be implemented using the MVC paradigm. The overall design of the view is flexible. It does not have to look fancy, but it should be organized well and easy to use. The view component of the GUI application must include at least:
    a. A text field for entering a search query
    b. A search button (alternatively, you can automatically update the search results when the text in the search field or the PageRank boost option is changed).
    c. A way for the user to indicate whether PageRank boost should be used or not (e.g., a radio button element).
    d. A list of the top 10 search results for the current query, sorted by their overall search score (from highest to lowest). Each search result must show the name of the page and the current search score for that page.
2. A class that implements the ProjectTester interface included on the project's Brightspace page. This will allow some automated tests to be run on your project. Note that this class must only support the required methods and will almost certainly create instances of your other classes (e.g., a crawler, a search engine, etc.) to perform the computations. In a way, you can look at this class as some 'glue' between your classes and an automated tester. Each method defined in this interface describes a bit about how that method should work. This is especially important for the search method, which differs slightly from the 1405 requirements.
3. A project report that includes the following:
    a. Instructions for running your GUI (i.e., which class is the controller).
    b. A list of functionality you have completed successfully and a list of the functionality that does not work.

c. An outline of each of your classes/interfaces/etc. that explains the main responsibility of each file in your project.
d. Discussion about the overall design of your system. You should discuss how you have applied the OOP principles in your project and how this has improved the overall quality of your implementation (think about modularity, extensibility, etc.). You should also include a description of how you have saved your data in files. You do not have to provide formal runtime complexity analysis but can discuss what steps you took to improve the efficiency of your application. Within the report, don't solely describe *how* you did something. You should also focus on explaining *why* you did it this way. Include arguments to support your decisions (e.g., why is this a good approach? what are its advantages? why is it better than alternatives? etc.).

Constraints and Assumptions

1. You must submit a valid IntelliJ project. You are free to work in whichever IDE you choose during development but will have to move your project into IntelliJ and ensure it works properly before submitting.
2. You must not modify any of the provided classes or interfaces.
3. The crawler must save all required information to files. It should be possible to run a crawl, stop the program, and run several searches later without re-running the crawler.
4. All files must be saved within the project using relative paths. You must also use the OS-specific file separator to ensure your code works when moved to another computer.
5. You may use any built-in Java/JavaFX functionality. You may not use any other external libraries.
6. (**The remainder of the assumptions and constraints are the same as those provided in the 1405 project specification**) You may assume the HTML is well-formatted (e.g., each <a> or tag has a matching closing or tag).
7. You may assume that there will be no additional HTML inside of the <a>… or … tags.
8. You may assume that all links include an "href" property and the link location is contained inside quotation marks after href= inside of the link definition.
9. You may assume that the title for a page will be the text in between the <title> and </title> tags within the page.
10. Each URL will contain either an absolute or a relative URL. All absolute URLs will start with "http://". Links that do not start with "http://" will be relative links and you

will have to compute the full URL using a combination of the link address and the current page URL. You may assume that all relative links will start with "./".

11. You may assume that all words within a page will be contained between a start tag and end tag. Words may be separated by spaces or new-line characters (i.e., "\n").
12. While all pages in the test data are named N-x.html, **you may not assume** that this is generally true for all pages (i.e., pages may have any name).

Test Files

Test files like those provided for the 1405 project are included in the test-resources.zip file. These test files will interact with your class that implements the ProjectTester interface. In the individual test files, you will have to change the line that instantiates an instance of the ProjectTester interface to create an instance of your own tester (or name your own implementation ProjectTesterImp). There will be six different datasets that you can test your implementation on (tinyfruits, fruits, fruits2, fruits3, fruits4, and fruits5). If you want to test your program manually, you can start your crawl at the following links to crawl the pages associated with each dataset:

1. http://people.scs.carleton.ca/~davidmckenney/tinyfruits/N-0.html
2. http://people.scs.carleton.ca/~davidmckenney/fruits/N-0.html
3. http://people.scs.carleton.ca/~davidmckenney/fruits2/N-0.html
4. http://people.scs.carleton.ca/~davidmckenney/fruits3/N-0.html
5. http://people.scs.carleton.ca/~davidmckenney/fruits4/N-0.html
6. http://people.scs.carleton.ca/~davidmckenney/fruits5/N-0.html

The tinyfruits dataset contains only 10 interlinked web pages and is a good starting point to test your crawler/search functionality. The remainder of the datasets contain 1000 interlinked web pages and will take longer to crawl/test.

Submission

Submit a single .zip file named “project.zip” to Brightspace. This zip file should contain your IntelliJ project folder (which should include all required resources) and your analysis report in PDF format. Ensure that your file is a .zip and has the proper name. Download your submission afterward and check the contents to ensure everything uploaded as expected. You should verify that your project still works correctly after being extracted from the zip file. Consider running the tests again on the downloaded zip to ensure you get the expected results.