HOMEWORK 7- Fall 2015

HOMEWORK 7 - due Tuesday, December 1st no later than 6:00PM

REMINDERS:

- **Be sure your code follows the coding style for CSE214.**
- **Make sure you read the warnings about academic dishonesty. *Remember, all work you submit for homework or exams MUST be your own work.***
- **Login to your grading account and click "Submit Assignment" to upload and submit your assignment.**
- **You may use (and are encouraged to use) any Java API Data Structures you like to implement this assignment.**
- **You may use Scanner, InputStreamReader, or any other class that you wish for keyboard input.**

The World Wide Web is an open source information space where documents (*pages*) and other web resources are identified by URLs and can be accessed via hyperlinks. The system can be described as a directed graph (or 'web') of pages in which a link from page A to page B corresponds to a directed edge from node A to node B. Each document can be abbreviated as a handful of important keywords describing what the document is about. Given the vast amount of information available on the web, searching for the most relevant pages containing a keyword in an efficient manner is an incredibly important operation. In this assignment, you will create a basic search engine to generate a sorted list of web pages using a simplified version of Google's PageRank algorithm.

For this assignment, you will build a directed graph of WebPage objects by reading two files: `pages.txt` and `links.txt` (instructions on how to do so are included further below). Once the graph is constructed, you will run a basic search engine that prompts the user to enter a keyword. You will then perform the PageRank algorithm, and display the pages to the user in tabular form sorted by descending PageRank.

Required Classes

The following sections describe classes which are required for this assignment. Each section provides a description and the specifications necessary to complete each class. If you feel that additional methods would be useful, feel free to add them during your implementation as you see fit. However, all the variables and methods in the following specifications must be included in your project.

1. WebPage

Write a fully-documented class named `WebPage` which represents a hyperlinked document. The `WebPage` class should contain a String member variable for its URL, an int to represent its position in the adjacency matrix, an int to represent the rank (described later in the specs) of this WebPage, as well as a Collection of Strings containing the keywords describing this this page.

You should also implement a `toString()` method, which should print the URL of the document as well as all of the associated keywords in a neat tabular form.

- **`public WebPage()` - constructor (you may include a constructor with parameters)**
- **A `String` member variable:**
  - **`private String url`**
- **Two `int` member variables:**
  - **`private int index`**
  - **`private int rank`**
- **A `Collection<String>` member variable:**
  - **`private Collection<String> keywords`**
  - **Note : When we say `keywords` must be a "Collection" of Strings, we mean that it can be any data structure you like that implements the `Collection` interface - e.g. `ArrayList, Vector, Array, LinkedList`, etc.**
- **`public toString()` - returns string of data members in tabular form.**
  - **Note 1: Since we cannot determine the "Links" portion of this WebPage, we will substitute it for a dummy String. For example, we can use "***" (or anything that is unique). A sample result would be:**

```
0   | google.com         |***| search, knowledge, tech
```

In the WebGraph class, when we want to print, we will determine all the Links for the URL "google.com" and can use the String.replace() method to replace your unique String (e.g. *) with the correct values.**

2. WebGraph

Write a full-documented class named `WebGraph` that organizes the `WebPage` objects as a directed graph. The class should contain a Collection of `WebPages` member variable called `pages` and a 2-dimensional array of integers (adjacency matrix) member variable called `links`. The matrix should be of size `MAX_PAGES x MAX_PAGES`, where `MAX_PAGES` is a static integer constant defined as 40. Note that although there will be `MAX_PAGES` rows and `MAX_PAGES` columns in the matrix, the *logical* size of the matrix is restricted to the size of `pages`. This means that if there are only 6 elements in the `pages` Collection, then only rows/columns 0 to 5 should be considered part of the matrix, and rows/columns 6 to `MAX_PAGES-1` are out of bounds. If a page is added, then row 6 and column 6 will become available to the matrix, whereas if a page is removed, row 5 and column 5 will become unavailable.

Entries in the adjacency matrix will correspond to directed edges in the graph structure. For example, if `links[i][j]` is set to `1`, then that means there is a hyperlink (directed edge) from `pages[i]` to `pages[j]` (note that the bracket notation for `pages` is simply used for convenience and does not imply that `pages` has to be an array). Remember that since the graph is directed, an entry of 1 at `links[i][j]` does NOT imply that `links[j][i]` is also 1. Please review the following example for an overview of how the graph structure should work.

Directed Graph with Adjacency Matrix.

- **A static constant int variable:**
  - `public static final int MAX_PAGES = 40;`
- **A Collection of `WebPages` member variable (Note: This can be any class which implements the `Collection` interface):**
  - `private Collection<WebPage> pages;`
- **A 2-dimensional adjacency matrix of `ints` member variable:**
  - `private int[MAX_PAGES][MAX_PAGES] edges; // instead of array, you may use ArrayList, Vector, etc.`
- `public static WebGraph buildFromFiles(String pagesFile, String linksFile) throws IllegalArgumentException`
  - ***Brief:***
    - **Constructs a `WebGraph` object using the indicated files as the source for pages and edges.**
  - ***Parameters:***
    - `pagesFile` **- String of the relative path to the file containing the page information.**
    - `linksFile` **- String of the relative path to the file containing the link information.**
  - ***Preconditions:***
    - **Both parameters reference text files which exist.**
    - **The files follow proper format as outlined in the "Reading Graph from File" section below.**
  - ***Postconditions:***
    - **A WebGraph has been constructed and initialized based on the text files.**
  - ***Returns:***
    - **The `WebGraph` constructed from the text files.**
  - ***Throws:***
    - `IllegalArgumentException`**: Thrown if either of the files does not reference a valid text file, or if the files are not formatted correctly.**
  - ***Note:***
    - **See the section "Reading Graph from File" below for information on how to construct the graph using two files.**
- `public void addPage(String url, Collection<String> keywords) throws IllegalArgumentException`
  - ***Brief:***
    - **Adds a page to the `WebGraph`.**

- ***Parameters:***
  - `url` - The URL of the webpage (must not already exist in the `WebGraph`).
  - `keywords` - The keywords associated with the WebPage.

- ***Preconditions:***
  - `url` is unique and does not exist as the URL of a `WebPage` already in the graph.
  - `url` and `keywords` are not `null`.

- ***Postconditions:***
  - The page has been added to `pages` at index 'i' and `links` has been logically extended to include the new row and column indexed by 'i'.

- ***Throws:***
  - `IllegalArgumentException`: If `url` is not unique and already exists in the graph, or if either argument is `null`.

- `public void addLink(String source, String destination) throws IllegalArgumentException`
  - ***Brief:***
    - Adds a link from the WebPage with the URL indicated by `source` to the `WebPage` with the URL indicated by `destination`

  - ***Parameters:***
    - `source` - the URL of the page which contains the hyperlink to `destination`.
    - `destination` - the URL of the page which the hyperlink points to.

  - ***Preconditions:***
    - Both parameters reference `WebPages` which exist in the graph.

  - ***Throws:***
    - `IllegalArgumentException`: If either of the URLs are null or could not be found in `pages`.

- `public void removePage(String url)`
  - ***Brief:***
    - Removes the `WebPage` from the graph with the given URL.

  - ***Parameters:***
    - `url` - The URL of the page to remove from the graph.

  - ***Postconditions:***
    - The `WebPage` with the indicated URL has been removed from the graph, and it's corresponding row and column has been removed from the adjacency matrix.
    - All pages that has an index greater than the index that was removed should decrease their index value by 1.
    - If `url` is null or could not be found in `pages`, the method ignores the input and does nothing.

  - ***Note:***
    - When the page is removed, it's corresponding row and column must be removed from the adjacency matrix. This can be accomplished by copying `links[k][j+1]` to `links[k][j]` and `links[j+1][k]` to `links[j][k]` for $0 \leq k <$ `size(pages)` and $i \leq j <$ `size(pages)-1`

- `public void removeLink(String source, String destination)`
  - ***Brief:***
    - Removes the link from `WebPage` with the URL indicated by `source` to the `WebPage` with the URL indicated by `destination`.

  - ***Parameters:***
    - `source` - The URL of the `WebPage` to remove the link.
    - `source` - The URL of the link to be removed.

  - ***Postconditions:***
    - The entry in `links` for the specified hyperlink has been set to 0 (no link).
    - If either of the URLs could not be found, the input is ignored and the method does nothing.

- `public void updatePageRanks()`
  - ***Brief:***
    - Calculates and assigns the PageRank for every page in the `WebGraph` (see the PageRank Algorithm section for further detail).
    - **Note:** This operation should be performed after ANY alteration of the graph structure (adding/removing a link, adding/removing a page).

  - ***Postconditions:***
    - All `WebPages` in the graph have been assigned their proper PageRank.

- `public void printTable()`
  - ***Brief:***
    - **Prints the `WebGraph` in tabular form (*see sample I/O for more information*).**

3. SearchEngine

Write a fully-documented class named `SearchEngine` which will initialize a `WebGraph` from the appropriate text files and allow the user to search for keywords in the graph. To keep things interesting, the class should provide functionality to add/remove pages to/from the graph, as well as alter the hyperlinks between pages in the graph.

On start-up, the `SearchEngine` should construct a new `WebGraph` using the files indicated by the static constant Strings `PAGES_FILE` and `LINKS_FILE`. The user should then be presented with a menu allowing them to add a page, remove a page, add a link, remove a link, print the graph, search for a keyword, or quit the program. The appropriate operation should be handled depending on the selection.

- **Two constant static `String` variables:**
  - `public static final String PAGES_FILE = "pages.txt"`
  - `public static final String LINKS_FILE = "links.txt"`

- **A `WebGraph` member variable:**
  - `private WebGraph web`

- `public static void main(String[] args)`
  - ***Brief:***
    - **Provide a menu prompt and implement the following menu options:**

- `(AP) - Add a new page to the graph.`
- `(RP) - Remove a page from the graph.`
- `(AL) - Add a link between pages in the graph.`
- `(RL) - Remove a link between pages in the graph.`
- `(P) - Print the graph.`
  - `(I) Sort based on index (ASC)`
  - `(U) Sort based on URL (ASC)`
  - `(R) Sort based on rank (DSC)`
- `(S) - Search for pages with a keyword.`
- `(Q) - Quit.`

Note 1: You should make sure that the graph is sorted by `index` before making any changes to the structure (failing to do so will lead to unintended bugs). Also, be sure to update all of the PageRanks for pages after any alterations by using the `updatePageRanks()` method for the graph.
Note 2: After choosing the print option, the user should be given another set of menu options. Display the results based on the secondary menu option selected.

4. Comparator

You will write helper classes that will help you with the sorting. Review the sample code (later in the specs) to understand how it is implemented and used. Implement the following classes:

```
// sort numerically ASCENDING based on index of the WebPage
public class IndexComparator implements Comparator

// sort alphabetically ASCENDING based the URL of the WebPage
public class URLComparator implements Comparator

// sort numerically DESCENDING based on the PageRank of the WebPage
public class RankComparator implements Comparator
```

SAMPLE COMPARABLE CODE:

```
import java.util.*;

/*
 * An example of type abstraction that implements Comparable
 * and Comparator interfaces.
 *
 * The Comparable/Comparator interfaces provide a standard means
 * for communication with yet unknown types of objects.
*/
```

```
public class CollectionsTester {
         public static void main(String[] args) {
           ArrayList staff = new ArrayList();

           staff.add(new Employee("Joe",100000, 177700010));
           staff.add(new Employee("Jane",200000, 111100010));
           staff.add(new Employee("Bob",66666, 1999000010));
           staff.add(new Employee("Andy",77777, 188800010));

           Collections.sort(staff);                                    // Sort by salary
           System.out.println("Lowest paid employee: "+staff.get(0));    // Prints Bob

           Collections.sort(staff, new NameComparator());              // Sort by alphabetical order
           System.out.println("First employee in list: "+staff.get(0));  // Prints Andy

           Collections.sort(staff, new IdComparator());                // Sort by ID number
           System.out.println("Employee with lowest ID: "+staff.get(0)); // Prints Jane
         }
}

public class Employee implements Comparable {
        private String name;
        private int salary;
        private int id;
        public Employee(String initName, int initSal, int initId) {
                id     = initId;
                name = initName;
                salary = initSal;
        }
        public String getName(){ return name; }
        public int getSalary() { return salary; }
        public int getId(){ return id; }
        public void setSalary(int newSalary) {
                salary = newSalary;
        }
        public int compareTo(Object o) {
                Employee otherEmp = (Employee)o;
                if (this.salary == otherEmp.salary)
                        return 0;
                else if (this.salary > otherEmp.salary)
                        return 1;
                else
                        return -1;
        }
        public String toString() {
                return name + ", $" + salary + ", "+ id;
        }
}

public class NameComparator implements Comparator {
        public int compare(Object o1, Object o2) {
                Employee e1 = (Employee) o1;
                Employee e2 = (Employee) o2;
                return (e1.getName().compareTo(e2.getName()));
        }
}

public class IdComparator implements Comparator {
       public int compare(Object o1, Object o2) {
           Employee e1 = (Employee) o1;
           Employee e2 = (Employee) o2;
           if (e1.getId() == e2.getId())
               return 0;
           else if (e1.getId() > e2.getId())
               return 1;
           else
               return -1;
       }
```

```
}
```

Reading Graph from File:

pages.txt will contain information related to a single `WebPage` on each line in the following format:

```
// name keyword1 keyword2 ... keywordN
    google.com search knowledge tech
    stonybrook.edu university seawolf knowledge journalism
    cnn.com news current world journalism
    facebook.com social friends tech
    youtube.com video music cats tech
    wikipedia.org knowledge encyclopedia
    stackoverflow.com forum knowledge social
```

links.txt will contain a hyperlink on each line in the following format:

```
// source destination
    google.com stonybrook.edu
    google.com cnn.com
    google.com facebook.com
    google.com youtube.com
    google.com wikipedia.org
    google.com stackoverflow.com
    stonybrook.edu google.com
    stonybrook.edu cnn.com
    cnn.com facebook.com
    facebook.com google.com
    facebook.com cnn.com
    facebook.com youtube.com
    youtube.com facebook.com
    stackoverflow.com wikipedia.org
```

When constructing the graph, you should first insert all of the web pages into the `pages` Collection in the `WebGraph` object. Then, after all pages have been inserted into the graph, read each of the links and add them to the `links` adjacency matrix. This will ensure that you do not attempt to add and link between pages before adding either of the pages to the graph.

You may use the following code to open an input stream for reading from a text file:

```
    FileInputStream fis = new FileInputStream(fileName);

    InputStreamReader inStream = new InputStreamReader(fis);

    BufferedReader reader = new BufferedReader(inStream);
```

Once the stream is open, you can read one line at a time as follows:

```
     String data = reader.readLine();
```

You can also create the reader using a scanner:

```
     Scanner scanner = new Scanner(new File(fileName));
```

Check the documentation for BufferedReader or Scanner.

Page Rank Algorithm

PageRank is an algorithm developed by Larry Page (of Google fame) to attempt to order web pages based on relevance. The idea is very simple - the more hyperlinks pointing to a page, the more "popular" that page is in the web graph, and the more relevant the page is (note that this is an extremely simplified version - the actual algorithm involves many more complex operations to implement this basic idea). In this assignment, you will implement PageRank in it's most basic form - the rank of a WebPage is equal to the number of directed edges pointing *towards* the page in the `WebGraph`.

When a user enters a keyword, the program should look through all the `WebPages` in the `WebGraph` to find which ones are tagged with the keyword, and add them to a list. After all of the `WebPages` have been added, they should be sorted according to their PageRank in a descending fashion (ties can be broken arbitrarily). Note that the PageRank of `WebPage` 'p' can be easily calculated by adding all of the entries in the column 'i' of the adjacency matrix `links`, where 'i' is the index of 'p' in the `pages` Collection. For example, the PageRank of "wikipedia.org" in the graph below would be calculated by summing the 5'th column of the adjacency matrix, which would yield a PageRank of 2.

PageRank of "wikipedia.org" is 2.

After all the pages have been added and sorted, the URLs should be printed to the console. To make sure that the algorithm is working correctly, you should print out the total ranking (e.g. '1' for the highest PageRank, '2' for the second highest, etc.) as well as the actual PageRank value along with the URLs in tabular form.

GUI EXTRA CREDIT (OPTIONAL):

In addition to performing all the functions described above, the GUI must have clickable headers on all lists which sort by that column when clicked. Clicking again on that column should reverse the sorting order.

Sample Input/Output:

// Comment in green, input in red, output in black

```
// Load WebGraph from file. PageRanks should be calculated and initialized.
Loading WebGraph data...
Success!

Menu:
    (AP) - Add a new page to the graph.
    (RP) - Remove a page from the graph.
    (AL) - Add a link between pages in the graph.
    (RL) - Remove a link between pages in the graph.
    (P)  - Print the graph.
    (S)  - Search for pages with a keyword.
    (Q)  - Quit.

Please select an option: P

    (I) Sort based on index (ASC)
    (U) Sort based on URL (ASC)
    (R) Sort based on rank (DSC)

Please select an option: I

Index     URL                PageRank  Links                Keywords
-------------------------------------------------------------------------------------------------
  0    | google.com          |    2    | 1, 2, 3, 4, 5, 6  | search, knowledge, tech
  1    | stonybrook.edu      |    1    | 0, 2              | university, seawolf, knowledge, journalism
  2    | cnn.com             |    3    | 3                 | news, current, world, journalism
  3    | facebook.com        |    3    | 0, 2, 4           | social, friends, tech
  4    | youtube.com         |    2    | 3                 | video, music, cats, tech
  5    | wikipedia.org       |    2    |                   | knowledge, encyclopedia
  6    | stackoverflow.com   |    1    | 5                 | forum, knowledge, social

// Menu not printed in sample I/O.

Please select an option: S
Search keyword: knowledge

Rank    PageRank    URL
-----------------------------------------
  1   |    2     | google.com
  2   |    2     | wikipedia.org
  3   |    1     | stonybrook.edu
  4   |    1     | stackoverflow.com

// Menu not printed in sample I/O.

Please select an option: AL
Enter a source URL: wikipedia.org
Enter a destination URL: stonybrook.edu

Link successfully added from wikipedia.org to stonybrook.edu!

// Menu not printed in sample I/O.
```

```
Please select an option: AL
Enter a source URL: stackoverflow.com
Enter a destination URL: stonybrook.edu

Link successfully added from stackoverflow.com to stonybrook.edu!

// Menu not printed in sample I/O.

Please select an option: P

// Sub-menu not printed in sample I/O.

Please select an option: I

Index     URL                PageRank  Links                Keywords
-------------------------------------------------------------------------------------------------
  0    | google.com          |    2    | 1, 2, 3, 4, 5, 6  | search, knowledge, tech
  1    | stonybrook.edu      |    3    | 0, 2              | university, seawolf, knowledge, journalism
  2    | cnn.com             |    3    | 3                 | news, current, world, journalism
  3    | facebook.com        |    3    | 0, 2, 4           | social, friends, tech
  4    | youtube.com         |    2    | 3                 | video, music, cats, tech
  5    | wikipedia.org       |    2    | 1                 | knowledge, encyclopedia
  6    | stackoverflow.com   |    1    | 1, 5              | forum, knowledge, social

// Menu not printed in sample I/O.

Please select an option: S
Search keyword: knowledge

Rank    PageRank    URL
-----------------------------------------------
  1   |     3     | stonybrook.edu
  2   |     2     | google.com
  3   |     2     | wikipedia.org
  4   |     1     | stackoverflow.com

// Menu not printed in sample I/O.

Please select an option: AP
Enter a URL: tumblr.com
Enter keywords (space-separated): blog art social friends

tumblr.com successuflly added to the WebGraph!

// Menu not printed in sample I/O.

Please select an option: AL
Enter a source URL: facebook.com
Enter a destination URL: tumblr.com

Link successfully added from facebook.com to tumblr.com!

// Menu not printed in sample I/O.

Please select an option: P

// Submenu not printed in sample I/O.

Please select an option: I

Index     URL                PageRank  Links                Keywords
-------------------------------------------------------------------------------------------------
  0    | google.com          |    2    | 1, 2, 3, 4, 5, 6  | search, knowledge, tech
  1    | stonybrook.edu      |    3    | 0, 2              | university, seawolf, knowledge, journalism
  2    | cnn.com             |    3    | 3                 | news, current, world, journalism
  3    | facebook.com        |    3    | 0, 2, 4, 7        | social, friends, tech
  4    | youtube.com         |    2    | 3                 | video, music, cats, tech
  5    | wikipedia.org       |    2    | 1                 | knowledge, encyclopedia
  6    | stackoverflow.com   |    1    | 1, 5              | forum, knowledge, social
```

```
  7   | tumblr.com         |    1    |                    | blog, art, social, friends

// Menu not printed in sample I/O.

Please select an option: S
Search keyword: social

Rank    PageRank    URL
-----------------------------------------------
  1   |     3      | facebook.com
  2   |     1      | tumblr.com
  3   |     1      | stackoverflow.com

// Menu not printed in sample I/O.

Please select an option: S
Search keyword: news

Rank    PageRank    URL
-----------------------------------------------
  1   |     3      | cnn.com

// Menu not printed in sample I/O.

Please select an option: RL
Enter a source URL: google.com
Enter a destination URL: cnn.com

Link removed from google.com to cnn.com!

// Menu not printed in sample I/O.

Please select an option: S
Search keyword: news

Rank    PageRank    URL
-----------------------------------------------
  1   |     2      | cnn.com

// Menu not printed in sample I/O.

Please select an option: RP
Enter a URL: youtube.com

youtube.com has been removed from the graph!

// Menu not printed in sample I/O.

Please select an option: P

// Submenu not printed in sample I/O.

Please select an option: I

// Note that the link numbers have changed for indices > 4.

Index     URL                PageRank  Links                Keywords
-------------------------------------------------------------------------------------------------
  0    | google.com         |    2    | 1, 3, 4, 5         | search, knowledge, tech
  1    | stonybrook.edu     |    3    | 0, 2               | university, seawolf, knowledge, journalism
  2    | cnn.com            |    2    | 3                  | news, current, world, journalism
  3    | facebook.com       |    2    | 0, 2, 6            | social, friends, tech
  4    | wikipedia.org      |    2    | 1                  | knowledge, encyclopedia
  5    | stackoverflow.com  |    1    | 1, 4               | forum, knowledge, social
  6    | tumblr.com         |    1    |                    | blog, art, social, friends

// Menu not printed in sample I/O.

Please select an option: P
```

```
// Submenu not printed in sample I/O.

Please select an option: U

Index     URL                PageRank   Links                Keywords
-------------------------------------------------------------------------------------------------------
  2    | cnn.com            |    2     | 3                  | news, current, world, journalism
  3    | facebook.com       |    2     | 0, 2, 6            | social, friends, tech
  0    | google.com         |    2     | 1, 3, 4, 5         | search, knowledge, tech
  5    | stackoverflow.com  |    1     | 1, 4               | forum, knowledge, social
  1    | stonybrook.edu     |    3     | 0, 2               | university, seawolf, knowledge, journalism
  6    | tumblr.com         |    1     |                    | blog, art, social, friends
  4    | wikipedia.org      |    2     | 1                  | knowledge, encyclopedia

// Menu not printed in sample I/O.

Please select an option: P

// Submenu not printed in sample I/O.

Please select an option: R

Index     URL                PageRank  Links                Keywords
-------------------------------------------------------------------------------------------------
  1    | stonybrook.edu     |    3     | 0, 2               | university, seawolf, knowledge, journalism
  0    | google.com         |    2     | 1, 3, 4, 5         | search, knowledge, tech
  2    | cnn.com            |    2     | 3                  | news, current, world, journalism
  3    | facebook.com       |    2     | 0, 2, 6            | social, friends, tech
  4    | wikipedia.org      |    2     | 1                  | knowledge, encyclopedia
  5    | stackoverflow.com  |    1     | 1, 4               | forum, knowledge, social
  6    | tumblr.com         |    1     |                    | blog, art, social, friends


// Special Cases:

// Menu not printed in sample I/O.

Please select an option: S // Searching for non existent keyword.
Search keyword: dogs

No search results found for the keyword dogs.

// Menu not printed in sample I/O.

Please select an option: AP // Duplicate URLs.
Enter a URL: facebook.com
Enter keywords (space-separated): social friends tech

Error: facebook.com already exists in the WebGraph. Could not add new WepPage.

// Menu not printed in sample I/O.

Please select an option: AL // Non-existent URLs.
Enter a source URL: instagram.com
Enter a destination URL: facebook.com

Error: instagram.com could not be found in the WebGraph.

// Menu not printed in sample I/O.

Please select an option: Q

Goodbye.
```
