CPSC 1060: Introduction to Programming in Java

Project 2: Extracting Links from Web Pages

Introduction

Web pages are text files written using HTML (Hypertext Markup Language), a structured language used to specify how pages should be laid out and presented to users by Web browsers. Web pages can contain links (*hyperlinks*) to images, documents, Web pages, and other files stored on the Web. The location of a file is indicated using a *URL* (*Uniform Resource Locator*), and Web browsers use these to retrieve the files and to navigate from one Web page to another. The World Wide Web, at least in part, consists of a network of linked Web pages and other files.

For this project, you are to create a Java program called **`LinkExtractor`** that prompts the user for a URL, accesses the Web page associated with that URL, parses the page's HTML, and extracts hyperlinks present in the HTML. The links will point to images, documents, and other pages on the Web (the different types will be indicated by the file extension contained in the URL). The program should allow the user to select the sort of links to search for, and the links that are found should be displayed to the user. The process of prompting the user for a URL, accessing the appropriate page, and printing out extracted links should be repeated until the user decides to quit.

The general logic for the program is shown in Figure 1, and your submission must follow it. Furthermore, the output of the program must be consistent with the details shown in the examples at the end of these instructions.

You should examine the Figure and the examples in order to determine how your program should work. Only after you have done this should you begin implementing your program. You may find it helpful to write down (using *pseudocode*) the various steps you'll need to implement before beginning to actually write your program code.

Learning Objectives

- Apply your knowledge of loops, decision statements, nested statements, variables, assignments, expressions, inputs, outputs, and algorithm design/implementation. This assignment emphasizes the use of loops and nested loops in a structured programming language.
- Testing and debugging source code.

Program Requirements

1. The program should prompt the user to enter a URL or else type "Q" to quit as shown in the examples. If the user enters an invalid URL, then the program should output **`"Input was unrecognized"`**.

   It should then prompt the user again, and repeat this process until the user enters a valid URL or else decides to quit the program.

2. For the sake of the program and to make things simple, a valid URL is one that begins with the prefix **`"http://"`** or **`"https://"`**.

3. When determining whether to quit, the program should accept any string beginning with the letter "Q", and it should not be sensitive to case (e.g., "**`quit the program`**" as input would cause the program to terminate).

4. After the user has entered a valid URL, the program should prompt the user to enter an integer in the range 1-5 that indicates the type of link to search for. You may assume that an integer value is entered, though not necessarily one in the range 1-5. If the user enters a number not in that range, then the program should re-prompt the user until a valid number is entered.

5. A class called **`Fetch`** has been provided for you. Invoking the method **`fetchURL(strURL)`**, where **`strURL`** is the string URL entered by the user (e.g., **`http://www.uga.edu`**) will cause the Web page associated with the URL to be accessed by the program and returned as a **`String`** object. The method may be invoked as shown below.

```
// Retrieve a web page and return its contents as a string.
String webpage = Fetch.fetchURL(strURL);
```

You must not modify the class **`Fetch`** in any way. To use the **`Fetch`** class, place a copy of **`Fetch.java`** in the same source folder as your **`LinkExtractor.java`**. Also, you must have working internet connection to use the **`Fetch`** class.

6. If there is a problem with the invocation of **`fetchURL`** (for instance, if the URL is not associated with a valid Web page), then the method will return a string that starts with **`$error$`** followed by a Java Exception message. Your program should check to see if such an error occurred. If so, it should print out the message following **`$error$`** and then return to the beginning of the program (prompting the user to enter a URL).

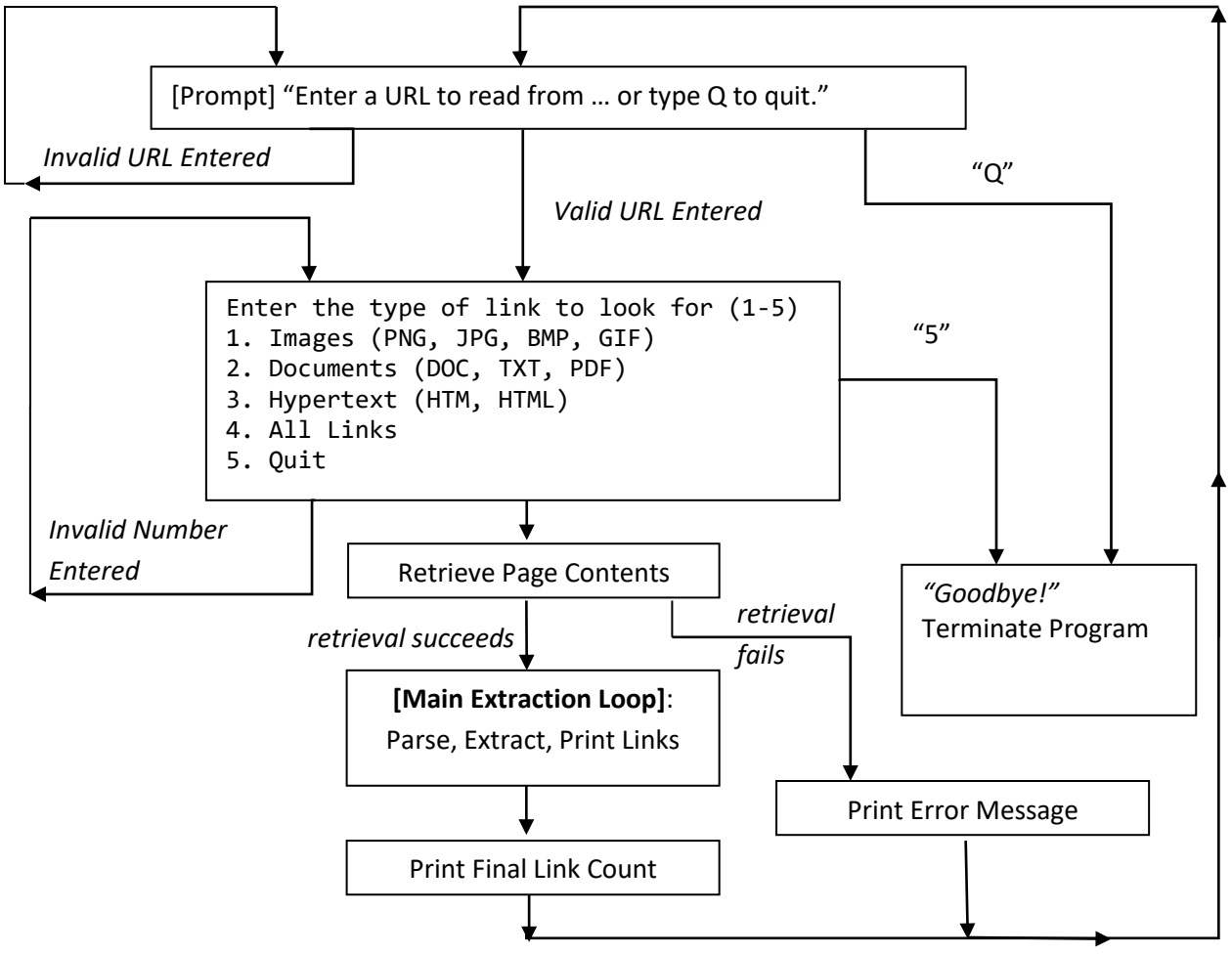
7. After the page has been successfully retrieved, the program should read through the text contained in it, looking for links. A hyperlink in a web page typically has the following form (and you may assume the following format for your project).

```
<a href="http://www.google.com">This is a link to Google</a>
```

The text to extract and display to the user is the value of the **`href`** attribute (**`http://www.google.com`**).

8. The value of the **`href`** attribute in a Web page might be enclosed in single quotes (**`'http://…com'`**) rather than double quotes. For the project, you may assume that all links will use double quotes.

Figure 1 – General algorithm for the program



9. As indicated in the figure above, the program should identify links of particular types (images versus documents, for instance). The types are indicated by so-called file extensions. E.g., the **`.jpg`** in

`tiger.jpg` indicates that the file is an image. Your program should output the URLs matching the user-specified extensions. You may assume that the extensions appear at the end of the URL. However, extensions are not case-sensitive (e.g. .JPG, .jpg, .JpG are valid image file extensions).

10. You should keep track of the total number of links your program has discovered. This number should be printed out after the page has been parsed, as indicated in the examples.
11. After the program has extracted the links and printed them out, it should prompt the user for another URL. That is, the program should continue to process Web pages until the user decides to quit.
12. As a safety measure, **`fetchURL`** can only be invoked 20 times before it causes the program to abort. When we test your program, we will not input more than 20 URLs for a single run of your program.

Further Requirements

13. Your program must implement the general logic indicated in Figure 1 and use the additional implementation details found in the examples at the end of these instructions.
14. Your program must use loops to re-prompt the player whenever an input is invalid, and it must use loops to parse the text of the web page. It should only terminate if the user types a string beginning with "Q" (or if the **`fetchURL`** method has been invoked more than 20 times).
15. Your program cannot use a **break** statement. The use of a break statement for this project will result in a grade of zero.
16. Your program cannot use a **continue** statement. The use of a continue statement for this project will result in a grade of zero.
17. Your program can only use the following classes/objects and their methods: System, System.out, String, Scanner, and Fetch (the classes/objects found in the Fetch class do not count since that code is already provided for you and that class should not be modified in any way). Using any other classes/objects than those aforementioned may result in a grade of 0.
18. Your program must include a comment between the import statements and the class declaration. Copy the comments below, and fill in the file name, your name, the submission date, and the program's purpose.

```
/*
* [Class name here].java
* Author:  [Your name here]
* Submission Date:  [Submission date here]
*
* Purpose: A brief paragraph description of the
* program. What does it do?
*
* Statement of Academic Honesty:
*
* The following code represents my own work. I have neither
* received nor given inappropriate assistance. I have not copied
* or modified code from any source other than the course webpage
* or the course textbook. I recognize that any unauthorized
* assistance or plagiarism will be handled in accordance
* with the policies at Clemson University and the
* policies of this course. I recognize that my work is based
* on an assignment created by the School of Computing
* at Clemson University. Any publishing or posting
* of source code for this project is strictly prohibited
* unless you have written consent from the instructor.
*/
```

All instructions must be followed for full credit to be awarded.

Project Submission

After you have thoroughly tested your program with all the examples provided plus your own examples, submit **`LinkExtractor.java`** (and only that file) to our course lab website under this project's assignment.

Project Grading

All projects are graded out of a possible 100 points. Programs not submitted before the deadline will receive a grade of zero. Programs that do not compile, use a break statement, or use a continue statement will also receive a grade of zero. Points might be deducted for not following instructions. You must make absolutely certain your program compiles before submitting, and you must thoroughly test your program with different inputs to verify that it is working correctly.

This project will be graded for both correctness and style:

Style [20pts]

- 5 points for including the class comment required for all projects.
- 5 points for helpful in-line comments.
- 5 points for using well-named variables, conforming to Java naming conventions, and naming your file correctly (spelling and case).
- 5 points for proper and consistent code indentation and readability.

Correctness [80pts]

- 80 points for correct output on various tests cases.

Examples

Your program should be consistent with the examples below. All input/output should be formatted as shown below except for word wrapping for long lines of output (long lines of output should appear on a single line in your program). Also note that the italicized and underlined ***Example*** headings should not be part of your program's output. Each example represents a single run of a correctly working program.

Example 1

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test1.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 4

Searching http://slamarc.people.clemson.edu/cpsc1060/files/test1.html
Searching for: all links

1. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/file1.doc
2. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/babbage.txt
3. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/AdaLovelace.pdf
```

```
4. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Alan Turing -
Wikipedia, the free encyclopedia.pdf
5. http://farm5.staticflickr.com/4054/4387582220_ae690a67df_o.jpg
6. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Meow%27s_first_weigh-in.jpg
7. http://upload.wikimedia.org/wikipedia/commons/3/3c/Parts_of_a_shark.svg
8. http://www.clemson.edu/index.html
9. http://en.wikipedia.org/wiki/Ren%C3%A9_Magritte

Number of Links Found: 9

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> Q
Goodbye!
```

Example 2

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> quit
Goodbye!
```

Example 3

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test1.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 2

Searching http://slamarc.people.clemson.edu/cpsc1060/files/test1.html
Searching for: documents

1. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/file1.doc
2. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/babbage.txt
3. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/AdaLovelace.pdf
4. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Alan Turing -
Wikipedia, the free encyclopedia.pdf

Number of Links Found: 4

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test1.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 3
```

```
Searching http://slamarc.people.clemson.edu/cpsc1060/files/test1.html
Searching for: hypertext

1. http://www.clemson.edu/index.html

Number of Links Found: 1

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> q
Goodbye!
```

Example 4

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> Hello

Input was unrecognized.
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> Q2
Goodbye!
```

Example 5

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://AnUnknownHost

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 4

Searching http://AnUnknownHost
Searching for: all links

java.net.UnknownHostException: AnUnknownHost

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> QQQQ
Goodbye!
```

Example 6

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> AnUnknownHost

Input was unrecognized.
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> AnUnknownHost2

Input was unrecognized.
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://AnUnknownHost/test2.html

Enter the type of link to look for (1-5):
```

```
1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 4

Searching http://AnUnknownHost/test2.html
Searching for: all links

java.net.UnknownHostException: AnUnknownHost

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> Quit
Goodbye!
```

Example 7

```
Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test3.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 1

Searching http://slamarc.people.clemson.edu/cpsc1060/files/test3.html
Searching for: images

1. http://en.wikipedia.org/wiki/File:Battlestar_Galactica_intro.jpg
2. http://en.wikipedia.org/wiki/File:Rotating_earth_%28large%29.GiF
3. http://en.wikipedia.org/wiki/File:BMPfileFormat.png
4. http://en.wikipedia.org/wiki/File:Edsger_Wybe_Dijkstra.jpg
5. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Meow%27s_first_weigh-in.jpg
6. http://dc147.4shared.com/img/fYDBEgfM/s3/Bulldog.bmp

Number of Links Found: 6

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test3.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 2

Searching http://slamarc.people.clemson.edu/cpsc1060/files/test3.html
```

```
Searching for: documents

1. http://www.clemson.edu/academics/integrity/documents/plagiarism-form.pdf
2. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/babbage.txt
3. http://www.irs.gov/pub/irs-pdf/fw9.pdf
4. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/AdaLovelace.pdf
5. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Alan Turing -
Wikipedia, the free encyclopedia.pdf
6. http://www.snee.com/xml/xslt/sample.doc

Number of Links Found: 6

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test3.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 3

Searching http://slamarc.people.clemson.edu/cpsc1060/files/test3.html
Searching for: hypertext

1. http://www.clemson.edu/academics/integrity/index.html
2. http://www.clemson.edu/academics/integrity/index.html
3. http://www.clemson.edu/index.html
4. http://broken123blah.htm

Number of Links Found: 4

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test3.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 4

Searching http://slamarc.people.clemson.edu/cpsc1060/files/test3.html
Searching for: all links

1. http://www.clemson.edu/academics/integrity/index.html
2. http://www.clemson.edu/academics/integrity/documents/plagiarism-form.pdf
3. http://en.wikipedia.org/wiki/File:Battlestar_Galactica_intro.jpg
4. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/babbage.txt
5. http://en.wikipedia.org/wiki/File:Rotating_earth_%28large%29.GiF
6. http://upload.wikimedia.org/wikipedia/commons/3/3c/Parts_of_a_shark.svg
```

```
7. http://en.wikipedia.org/wiki/File:BMPfileFormat.png
8. http://en.wikipedia.org/wiki/File:Edsger_Wybe_Dijkstra.jpg
9. http://www.irs.gov/pub/irs-pdf/fw9.pdf
10. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/AdaLovelace.pdf
11. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Alan Turing -
Wikipedia, the free encyclopedia.pdf
12. http://slamarc.people.clemson.edu/cpsc1060/files/Project2Files/Meow%27s_first_weigh-in.jpg
13. http://www.snee.com/xml/xslt/sample.doc
14. http://www.clemson.edu/academics/integrity/index.html
15. http://dc147.4shared.com/img/fYDBEgfM/s3/Bulldog.bmp
16. http://www.clemson.edu/cecas/departments/computing/
17. http://www.clemson.edu/index.html
18. http://slashdot.org
19. http://broken123blah.htm

Number of Links Found: 19

Enter a URL to read from (include the "http://" or "https://"), or type Q to quit:
>> http://slamarc.people.clemson.edu/cpsc1060/files/test3.html

Enter the type of link to look for (1-5):

1. Images (PNG, JPG, BMP, GIF)
2. Documents (DOC, TXT, PDF)
3. Hypertext (HTM, HTML)
4. All Links
5. Quit
>> 5
Goodbye!
```