The Spider Lab: one crawler

The World Wide web is aptly named when you consider the URL links found in pages. One page can have many links in it that take a viewer to another page, which has more links, and so on, forming a very large cyclic graph of interconnected pages. In this lab you will be finishing some code for a web crawler, or spider, that will start with a 'seed' URL to a web page and read it to find links to other pages. Those links will be placed on a *queue* for further processing (we'll call this the work queue). When the initial page is processed, it is placed on another data structure to indicate that it has been visited. This process is repeated for the next page whose link is on the work queue. The code you will be given uses a Java library for parsing html files and looking for links (java.net.URL).

Here are the files in the package ***lab.spider***, which you will use as your staring point:

AllWordsCounter.java
  // contains a 'dictionary' to hold counts of how often a URL is encounterd
HttpHelper.java
  // contains methods to read html pages and extract links; also can detect whether a URL is an image
RunSpider.java   // has main()
Spider.java   // the workhorse and the one you will be changing
TestHttpHelper.java   // JUnit test class
TestSpider.java   // JUnit test class
WordCount.java   // small helper class that holds a word and a count

Start by creating lab.spider in your own repository and copying these files.

The Spider.java class is the one that you should work on for this assignment. The RunSpider class contains main() and uses it. As the code stands now it doesn't really do anything if you run it.

Examine the code in the files. Begin by creating a class diagram that shows which classes 'use' or 'have' one of the other classes.

Your task is to finish the Spider class by doing the following:

1. Complete the processPage method. When it works, one of the TestSpider unit tests should pass.
2. Complete the crawl() method. When it works, both TestSpider unit tests should pass.

Note that there are comments in these methods to help assist you.

Once your unit tests pass, you should be able to run the code, which is currently 'hard-coded' to start at macalester.edu, and see it produce the URLs found when crawling, along with how many times it saw them.

Try this:

- Experiment with this variable found in Spider: maxurls   If you double it, how many new urls were encountered? You might want to make a method that would answer this for you.
- Experiment with the BEGNNING_URL variable found in RunSpider by choosing some other pages of interest to you as starting points.