Improving the ‘Spider’

First Question: How much work is there?

Once you have a completed working spider, let’s examine how much work it has to do. Try some experiments in which you continue using increasing values of maxUrls in the Spider. Please note that you can provide this value in its constructor. Add a method to the Spider that enables you to ask how many pages are still left to work on in the ‘work’ queue. You may also want to add a method to know how many pages have been finished.

Change the RunSpider class to run some experiments with different values of maxUrls by executing several Spiders. For each value of maxUrls, report on how much work is left to do. How quickly is our Spider overloaded with work?

Multiple Spiders to the rescue

Now let’s examine how we can use multiple spiders working at the same time on this problem. Your instructor will take a moment to explain how we will use a technique called threads to run many spiders at the same time, each of who will access the work, finished, and urlCounter queue. Then you will try this out below.

There is now a new lab.concurrentSpider package in our shared space. Examine the RunThreadedSpider class. Note that we now use a Java class called a Thread to begin running multiple instances of the Spider in many Threads. The Spider is now in a class called ConcurrentSpider, and implements an interface called Runnable.

A key feature of concurrently running Spiders is that they must ***share the same data structures in order to work together.*** To do this, we need to place the data structures they are working on in one class and create one instance of that class in RunConcurrentSpider. Then each new ‘Runnable’ ConcurrentSpider will receive a reference to that class of shared data structures.

First try: share our original data structures

We are gong to try this process in 2 steps, so you will first look at the ‘first version’, where we will try to share the original data structures designed for the single Spider. Examine the RunConcurrentSpider1 and ConcurrentSpider1 classes. Create the class called SharedSpiderData1 and move the data structures into it from your original Spider class from the lab.spider package. Make getters for each data structure (work, finished, urlCounter).

Finish the ConcurrentSpider1 class by filling in the code needed in run() and the processPage() method-- this should be much like you did it for the ‘sequential’ single Spider case. You should be able experiment with your new ConcurrentSpider1 by running RunThreadedSpider1. Try running it several times without changing anything. Can you tell if you get the same results each time?

Notes:
The following classes are needed for this first try:
AllWordsCounter
ConcurrentSpider1
HTMLHelper
RunThreadedSpider1
TestHTTPHelper
WordsCount

You will make the SharedSpiderData1 class.

Second try: concurrent data structures

Your instructor will discuss an important improvement in class and share come more code with you.

Following that discussion, now use the new Java Concurrent Data Structures from the package java.util.concurrent. Begin with the file SharedSpiderData to see the types of shared, thread-safe data structures we will use for this version of the multi-threaded crawler.

Finish the classes called ConcurrentSpider and RunThreadedSpider.