Web Crawler Comments

There are many possible ways to code up a web crawler. The following is one suggested approach for this assignment (this was my approach to my WebCrawler program).

According to the assignment requirements, you should have a `WebElement` interface.

The three classes that are to implement that interface are: `WebFile`, `WebImage`, and `WebPage`. Each of these classes should have some getter methods, and a save method, along with whatever other methods you decide are needed. The `WebPage` class should also have the `crawl()` method.

Among other things, the `WebPage` class will have it's own list of images, its own list of files, and its own list of pages linked to from that page. (I say "list", but they could be vectors, arraylists, arrays, etc. - however you prefer to implement the lists). So, the constructor for a WebPage will, possibly among other things, instantiate new vectors (or arraylists, or arrays, etc.) for each list of images, files & pages. The `crawl()` method will save all the images, files, and pages to their respective lists for that page. The getter methods for each of those lists will return that vector/arraylist/array.

The `WebCrawler` class is the engine for the whole program (the main method resides here). In this class, you'll have some variables including a `DownloadRepository`, which is where the images that will be written to disk will be held, and also the files that will be held to disk will be held. The command line arguments from `args[]` will be taken care of, and then `crawl()` will be called with the url passed into `args[0]` using that as the start page. You may loop depth_#_of_times saving for each page that is crawled the images and files (from it's own list of images and list of files) to the download repository. Each other page link on the given page being crawled will need to be saved for the next level of crawling, making sure duplicates are not saved (you don't want to crawl the same page twice or get stuck in a circular crawl). After all the crawling is done and all the saving is done, and the download repository has all the images from each page and all the files from each page, then those images and files can be written to disk.

The `DownloadRepository` class is to be implemented as a singleton - you only want one repository to hold all the images and all the files from each page that is crawled. This class will have methods to save the images from each page's list of images to the repository; same for the files; as well as a method to write the files to disk; and whatever other needed methods you may come up with. I also have a `Repsitory` interface that the `DownloadRepository` implements.