```
********************** WebImage.Java *************************

package cu.cs.cpsc215.project1;

import java.io.IOException;
import java.util.ArrayList;
import org.jsoup.Connection.Response;
import org.jsoup.Jsoup;
import org.jsoup.helper.HttpConnection;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

/**
 * WebPage (imp WebElement):
        Constructor()
                Pass an address and save said address

        Crawl(<Depth>)
                Multiple ways I could do this
                after time thinking about it, I feel the simplest method would be
                        to increment/decrement depth
                1) Create ULR object and data structures needed
                        Data Structure choice is vast for this project
                        play around with a few to find the best
                                so far: ArrayList, Map, or Tree of some sort
                2) Check if depth is < 0
                        if so, you're done, return page
                3) Save string of ULR passed in, to ULR object
                4) BufferedReader that poop
                        find we elements
                        Sort them
                5) Check if reading is empty, then end?
                6) decrement depth

                What does this still need to be functional?
                        A way to "parse" what the BufferedReader gives?
                                Use JSoup?
                                        JSoup is OK, but keep experimenting…
                                Make a Parser?
                                        More Control of what's being passed.
                                        Make a parserBeReadin(), It can't be that hard
                Getters()
                        Still undecided on Data Structure I will use.
                        Once decided, just return the Data together. Easy.
                Sorter()
```

```
                        part of step 4 above
                        A way to sort through to see if an type
                                        of element as been read
                        pages will require you to enter and download within
                                same method of reading in,
                                just with <Depth> involved
                        basic alg:
                        if(WebPage)
                                add Web page to it's own Data for that type
                        if(WebImage)
                                add Image to it's own Data
                        if(WebFile)
                                add File to it's own Data?
                                or file contents?
 * @author cfitt
 *
 */

public class WebPage implements WebElement {

        String url;
        ArrayList<WebPage> pages = new ArrayList<WebPage>();
        ArrayList<WebImage> images = new ArrayList<WebImage>();
        ArrayList<WebFile> files = new ArrayList<WebFile>();
        ArrayList<String> visited = new ArrayList<String>();
        private static Elements media;
        private static Elements links;


        public WebPage(String url) {
                this.url = url;
        }

        public void crawl(int depth){
                if (depth < 0)//End Recursion Case
        return;

                System.out.println("Crawling " + url + " with a depth of " + depth);
                try { //Start Parse for page
                   Document doc = Jsoup.connect(url).get();
                   links = doc.select("[href]");//All Links
                   media = doc.select("[src]");//All Image Types

                   if(doc != null) { //If not a blank page, or down?
                           for (Element src : media) {
                                String imgtopass = src.absUrl("src");
```

```
                                    if (src.tagName().equals("img") &&
!visited.contains(imgtopass)) {
                                            WebImage newImage = new
WebImage(imgtopass);
                                   images.add(newImage);
                                   visited.add(imgtopass);
                                    System.out.println(imgtopass + " was downloaded");
                                }
                             }
                             /**
                              * this next part was tricky
                              *
                              * I was running into problems differentiating between
files/pages
                              * finally, after parousing the Jsoup Cookbook, I found a neat
recipe
                              *              after including I had a bunch of new problems,
                              *              but this current version works.
                              *
                              * Method:
                              *              1) check if the link is useable
                              *              2) set the content type
                              *              3) Use the content type passed to determine
                              *                             if link is a page or file
                              *              4) Pass pages into pages, which could be used
later if depth permits
                              *              5) pass files into folders, from
DownloadRepository
                              */
                             HttpConnection testLink = null;
                      Response testResponse = null;

                             for (Element link : links) {
                                    //test if the link is another use-able link
                                    String baselinkUrl = link.absUrl("href");
                                    baselinkUrl = baselinkUrl.trim();
                          if(!baselinkUrl.contains("#")) {
                            try {
                              testLink = (HttpConnection) Jsoup.connect(baselinkUrl);
                              testLink.ignoreContentType(true);
                              testLink.ignoreHttpErrors(true);
                              testResponse = (Response) testLink.execute();
                            } catch (Exception e) {
                              System.out.println(e.getLocalizedMessage());
                            }
                            String contentType = null;
```

```
                        if(testResponse != null) {
                          contentType = testResponse.contentType();
                        }
                        if(contentType == null) {continue;}
                        if(contentType.toString().equals("text/html")) {
                         if(!visited.contains(baselinkUrl)) {
                                  WebPage newPage = new WebPage(baselinkUrl);
                                  pages.add(newPage);
                                  visited.add(baselinkUrl);
                                  System.out.println(baselinkUrl + " Added Page");
                         }
                        } else if(!visited.contains(baselinkUrl)) {
                                  String fileUrl = baselinkUrl;
                                  WebFile newFile = new WebFile(fileUrl);
                                  files.add(newFile);
                                  visited.add(baselinkUrl);
                                  System.out.println(fileUrl + " Added File");
                        }
                      }
                         }
                 }
              } catch (IOException e){
                       System.out.println("Page Not Found");
                       return;
              }
              DownloadRepository downloadRepository =
DownloadRepository.getInstance();
    downloadRepository.addWebElements(this);
    for (WebPage page : pages)
                       page.crawl(depth - 1);
      }
      @Override
      public void saver(String savePath) {
   //unused in pages, since they hold elements, and are not elements
      }

      public ArrayList<WebImage> getImages(){
              return images;
      }

      public ArrayList<WebFile> getFiles(){
              return files;
      }

      public ArrayList<WebPage> getWebPages(){
              return pages;
```

```
            }
    }
```