HW2 Assignment

CS572 Course Assignments
Last Modified: December 22, 2018

Homework 1: Comparing Search Engine Results

- Search Engine Comparison Exercise
- Grading Guidelines
- Homework #1 Due Jan. 28th

Homework 2: Web Crawling

- [Instructions for Installing Eclipse and crawler4j]
- [Flowchart for Crawler4j]
- [Web Crawler Exercise]
- [Grading Guidelines]
- Homework #2 Due Feb. 20th

1. Java programming
   - I assume all of you know how to program in Java!
2. Eclipse Software Development Environment
3. crawler4j, an open source java web crawler
4. a crawl and analysis of a web site and an analysis of the crawl

What is Eclipse?

- Eclipse started as a proprietary IBM product (IBM Visual age for Smalltalk/Java)
  - Embracing the open source model IBM opened the product up
- Open Source
  - It is a general purpose open platform that facilitates and encourages the development of third party plug-ins
- Best known as an Integrated Development Environment (IDE)
  - Provides tools for coding, building, running and debugging applications
- Originally designed for Java, now supports many other languages
  - Good support for C, C++
  - Python, PHP, Ruby, etc…

Prerequisites for Running Eclipse

- Eclipse is written in Java and will thus need an installed JRE (Java Runtime Environment) or JDK (Java Development Kit) in which to execute
  - JDK recommended

Obtaining Eclipse

- Eclipse can be downloaded from…

  http://www.eclipse.org/downloads/packages/release/photon/r/eclipse-ide-java-developers
- Eclipse comes bundled as a zip file (Windows) or a tarball (all other operating systems)

Installing Eclipse

- Simply unwrap the zip file to some directory where you want to store the executables
- The document

"Instructions for Installing Eclipse and crawler4j"

- located at

http://www-scf.usc.edu/~csci572/2018Fall/hw2/Crawler4jinstallation.pdf

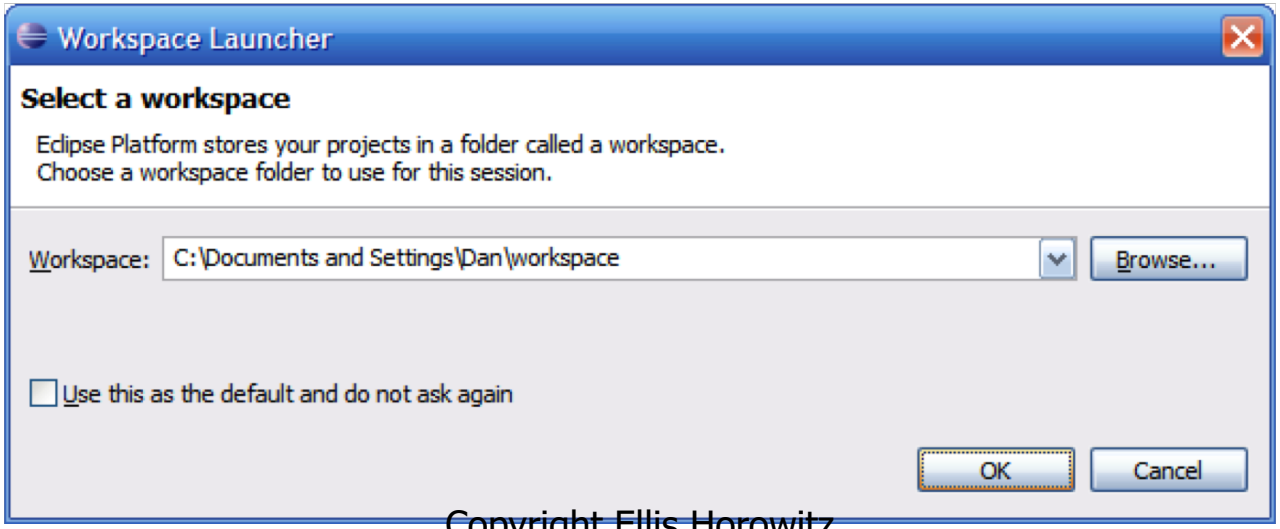
describes the installation for Windows and Macs

Launching Eclipse

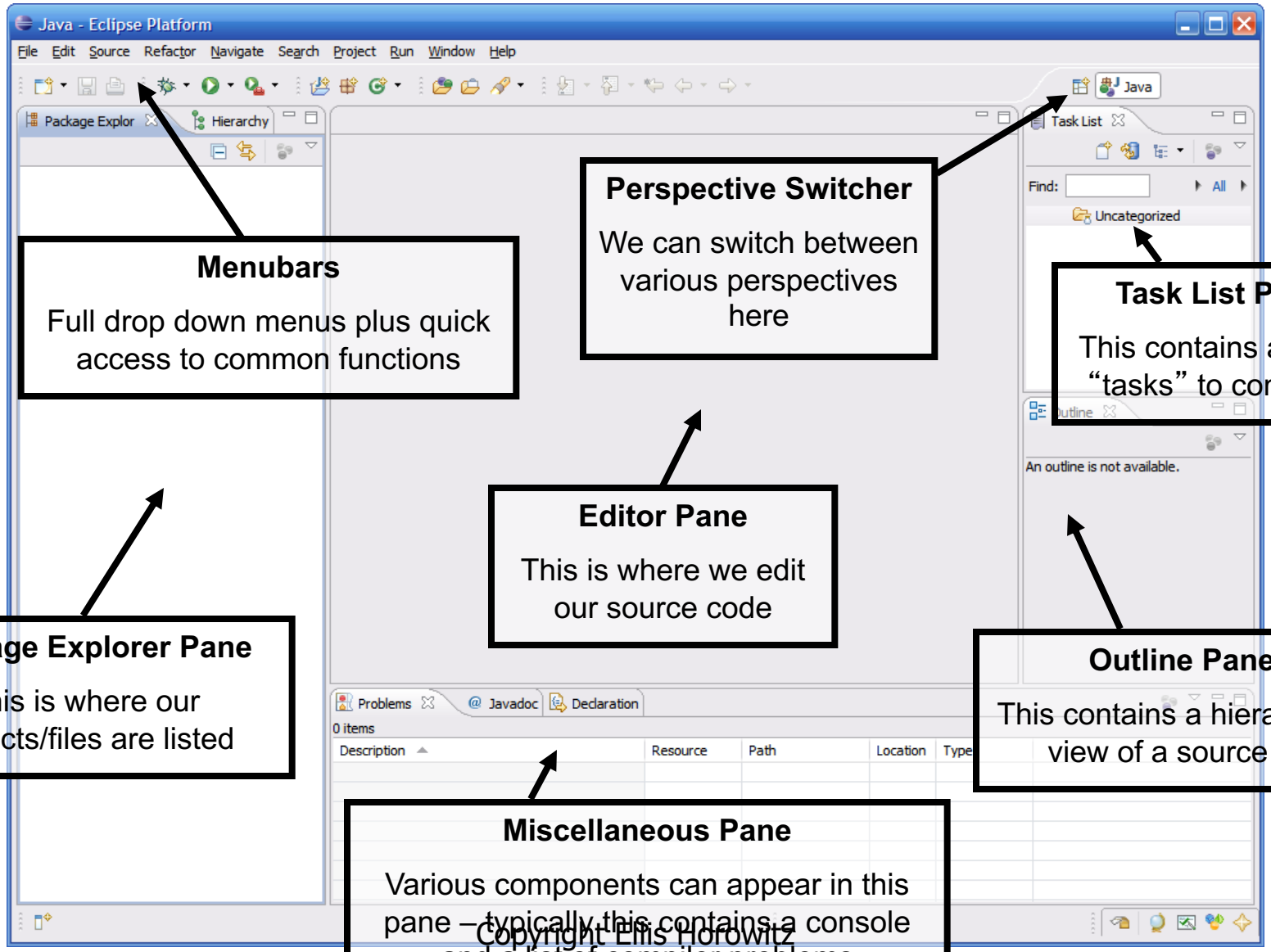
- Once you have the environment setup, go ahead and launch eclipse
- You should see a splash screen such as the one below…

Selecting a Workspace

- In Eclipse, all of your code will live under a *workspace*
- A *workspace* is nothing more than a location where we will store the source code and where Eclipse will write out preferences
- Eclipse allows you to have multiple workspaces – each tailored in its own way
- Choose a location where you want to store your files, then click OK



Eclipse IDE Components



Menubars

Full drop down menus plus quick access to common functions

Perspective Switcher

We can switch between various perspectives here

Task List Pane

This contains a list of "tasks" to complete

Package Explorer Pane

This is where our projects/files are listed

Editor Pane

This is where we edit our source code

Outline Pane

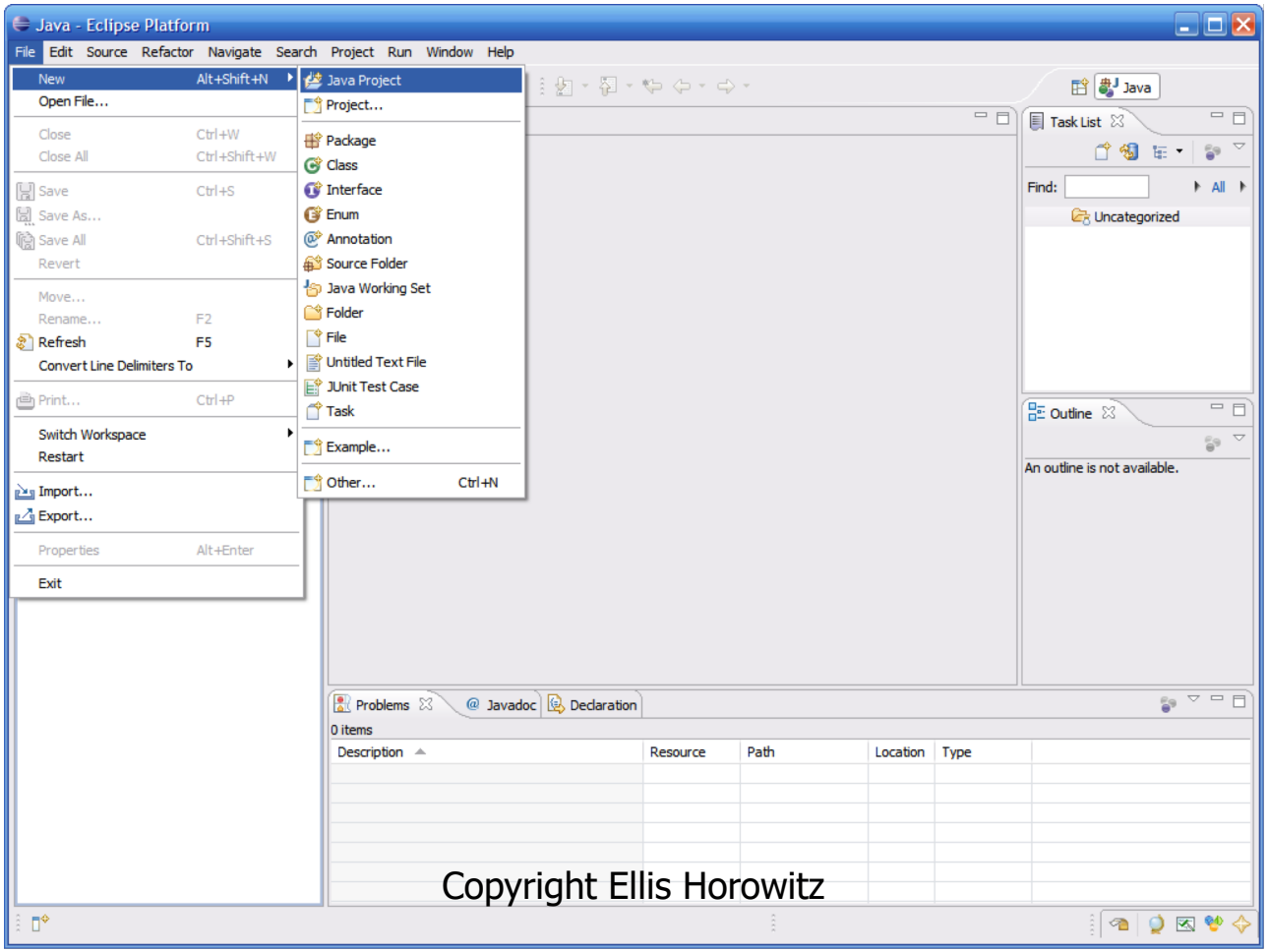
This contains a hierarchical view of a source file

Miscellaneous Pane

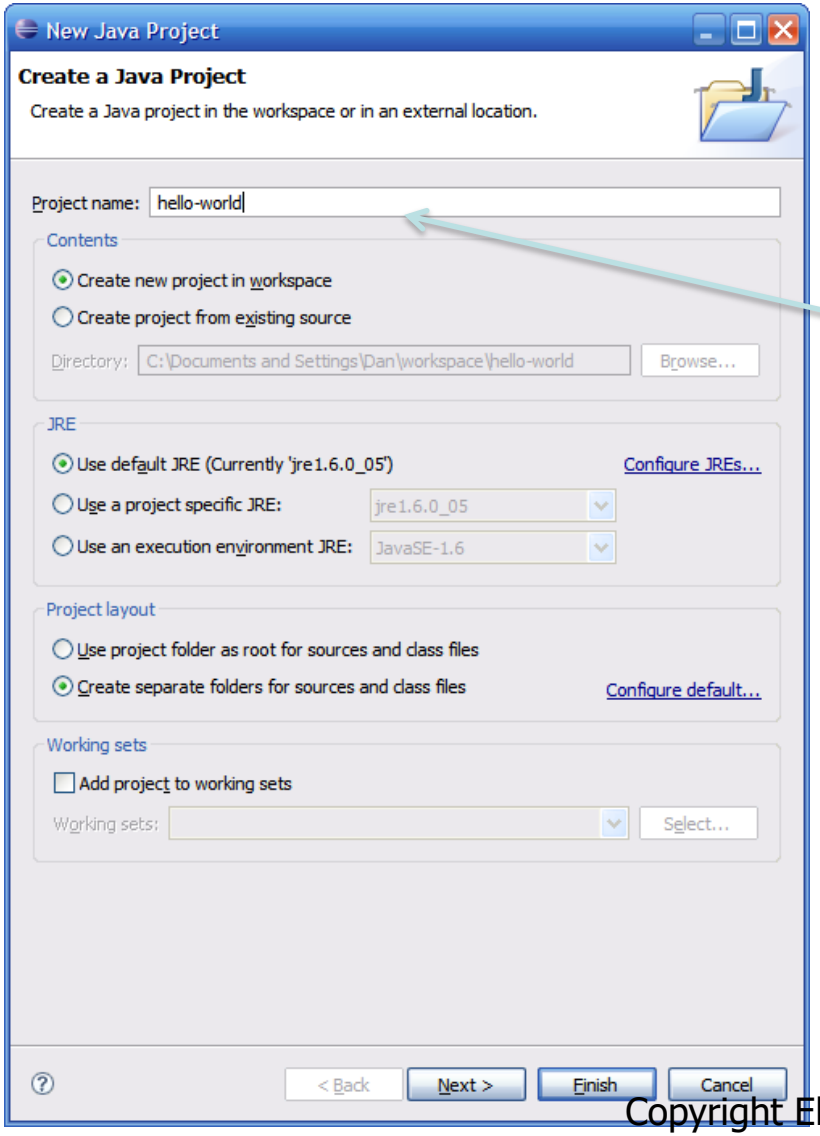
Various components can appear in this pane – typically this contains a console and a list of compiler problems

Creating a New Project

- All code in Eclipse needs to live under a project
- To create a project: File → New → Java Project



Creating a New Project (continued)

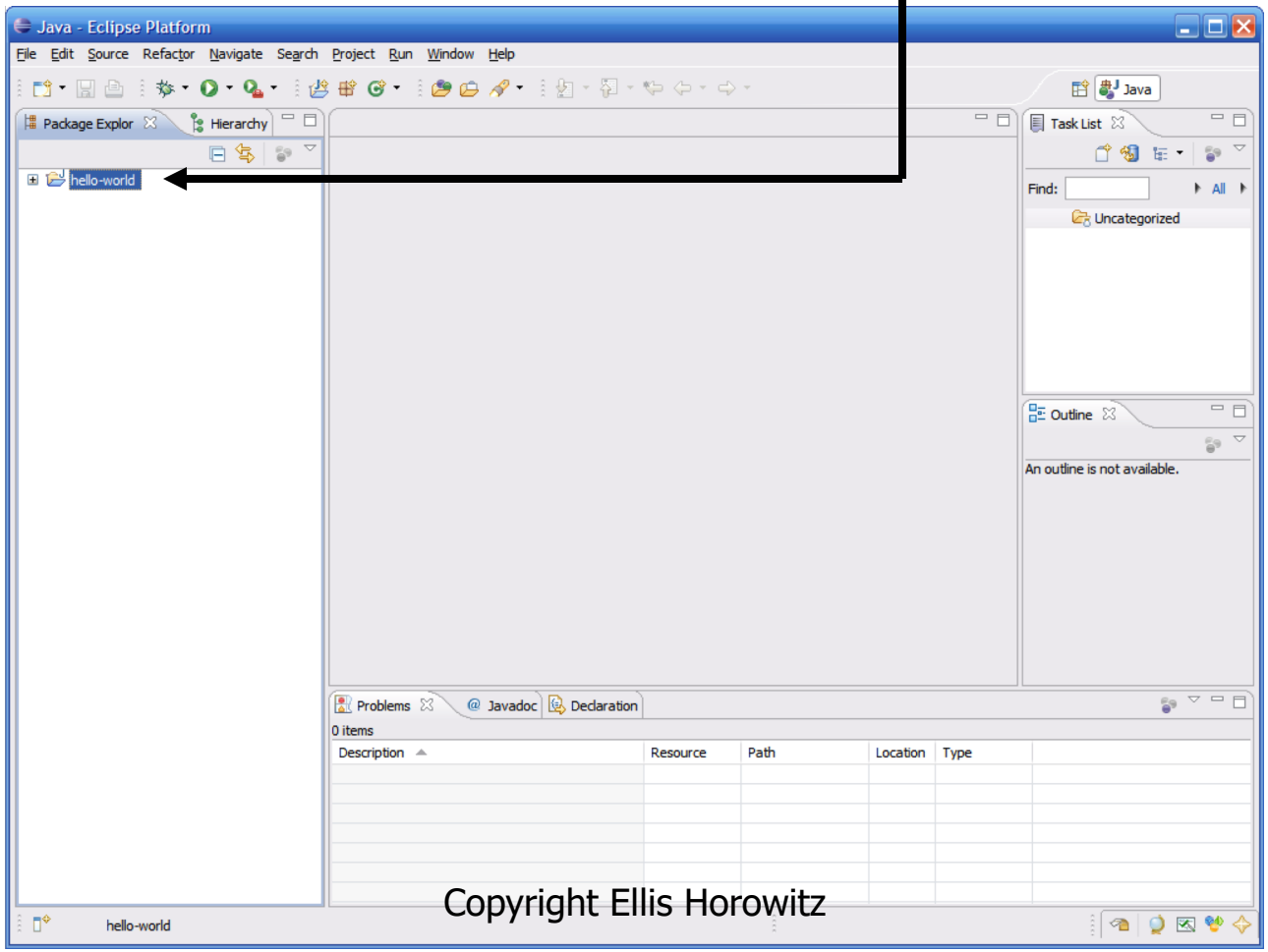


- Enter a name for the project, then click Finish

Hello-world Project

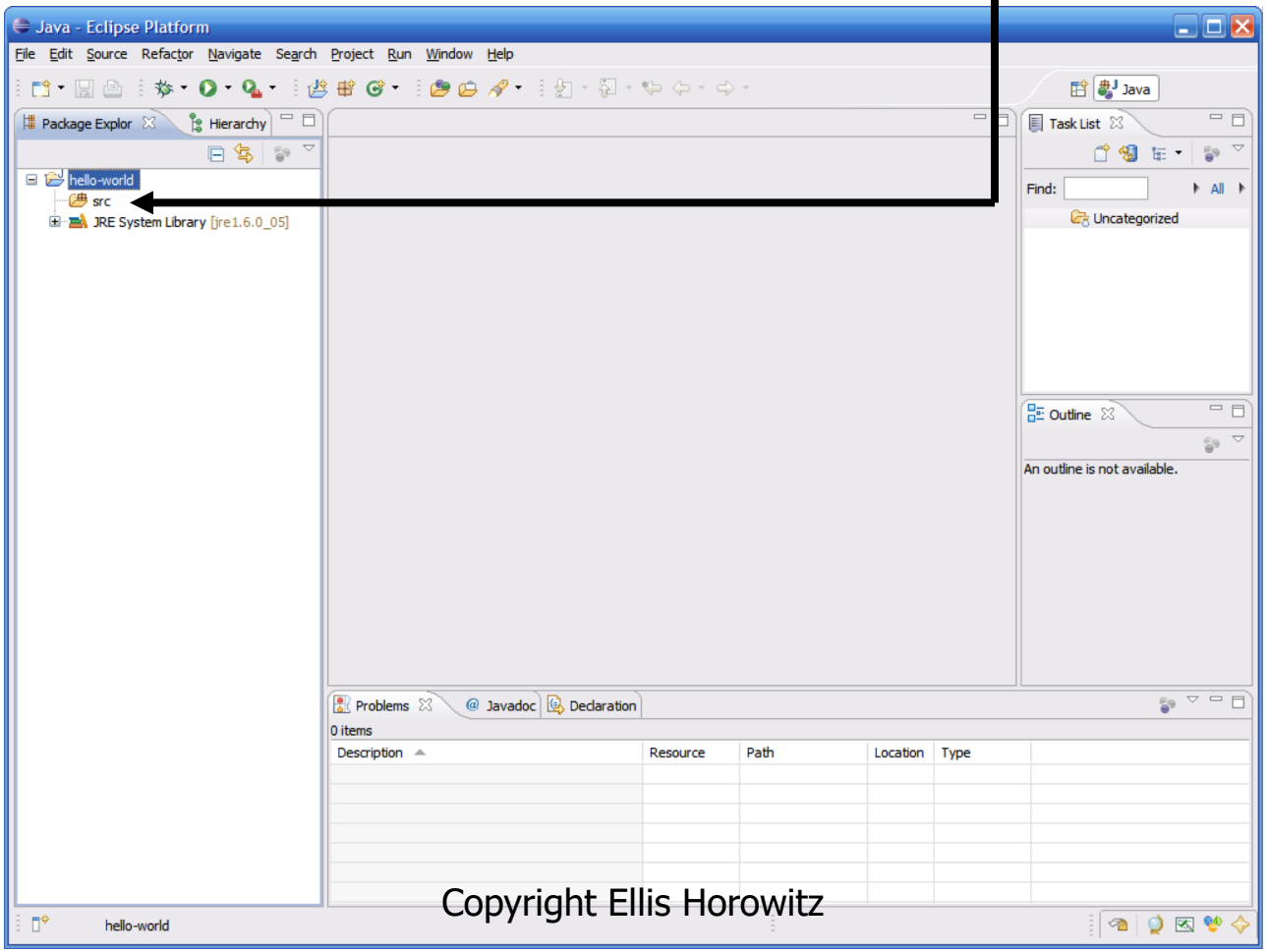
Creating a New Project (continued)

- The newly created project should then appear under the Package Explorer



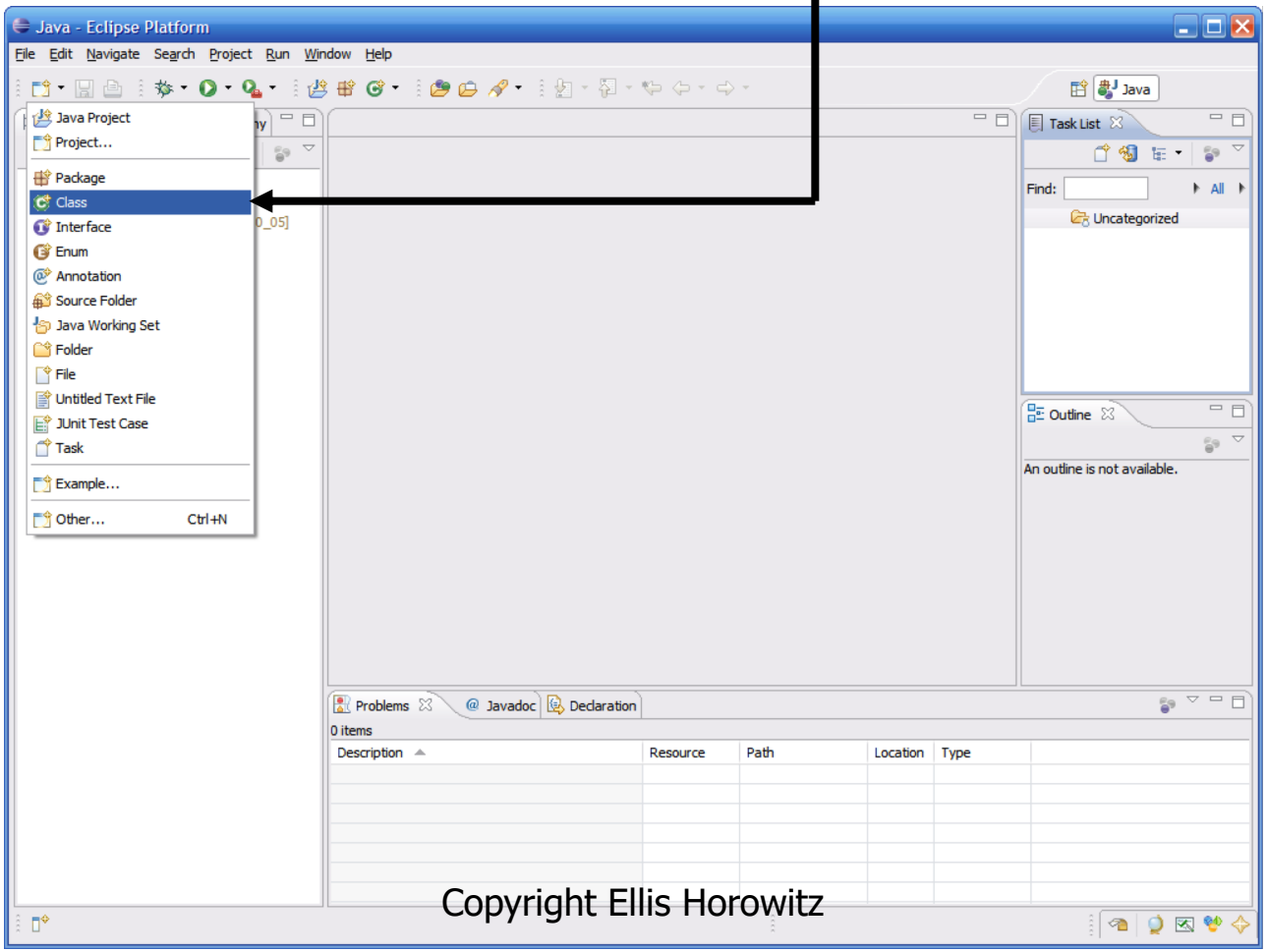
The src folder

- Eclipse automatically creates a folder to store your source code in called src

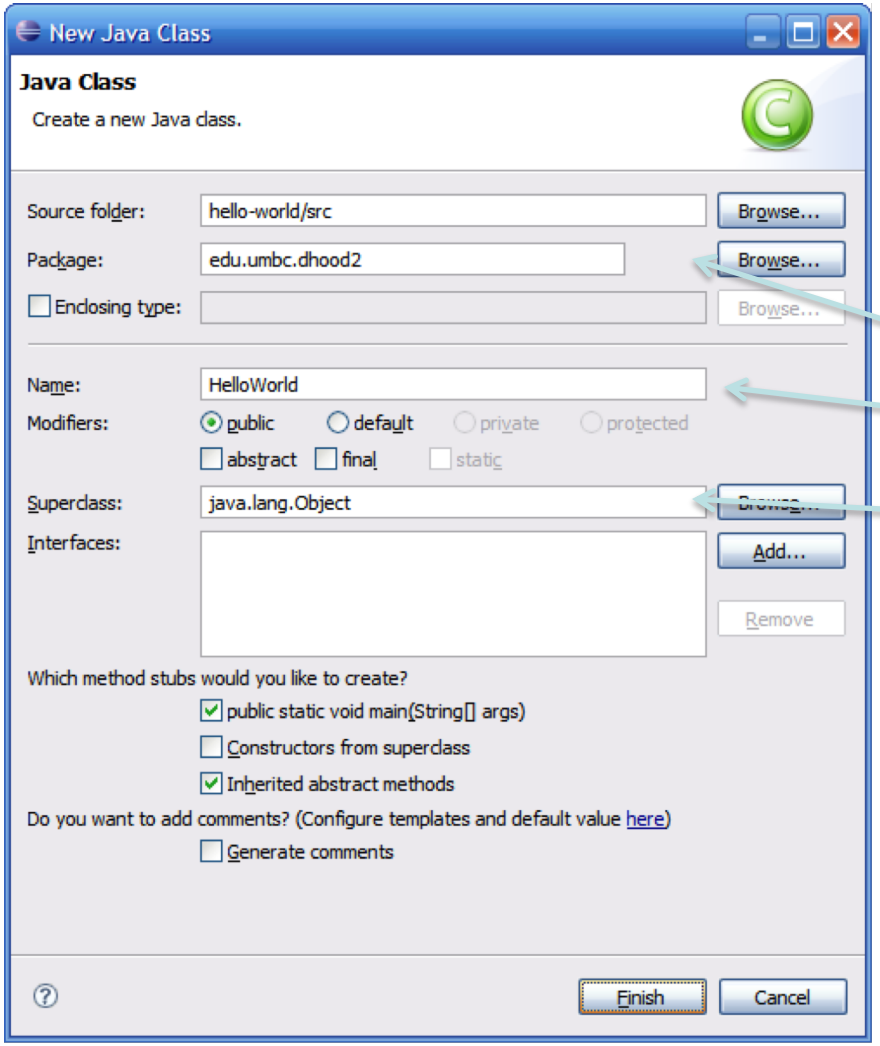


Creating a Class

- To create a class, simply click on the New button, then select Class



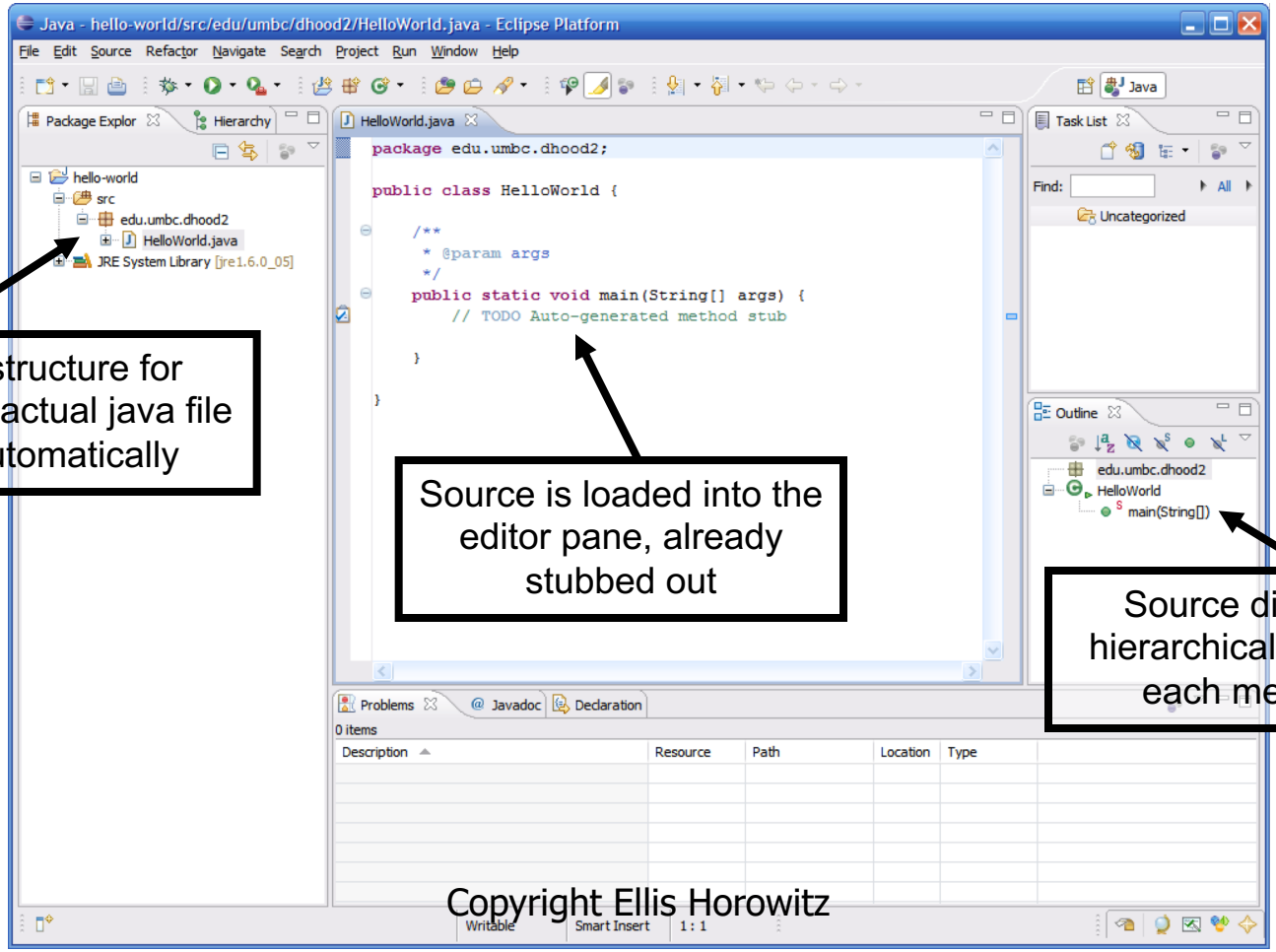
Creating a Class (continued)



- This brings up the new class wizard
- From here you can specify the following...
  - Package
  - Class name
  - Superclass
  - Whether or not to include a main
  - Etc…
- Fill in necessary information then click Finish to continue

The Created Class

- As you can see a number of things have now happened…

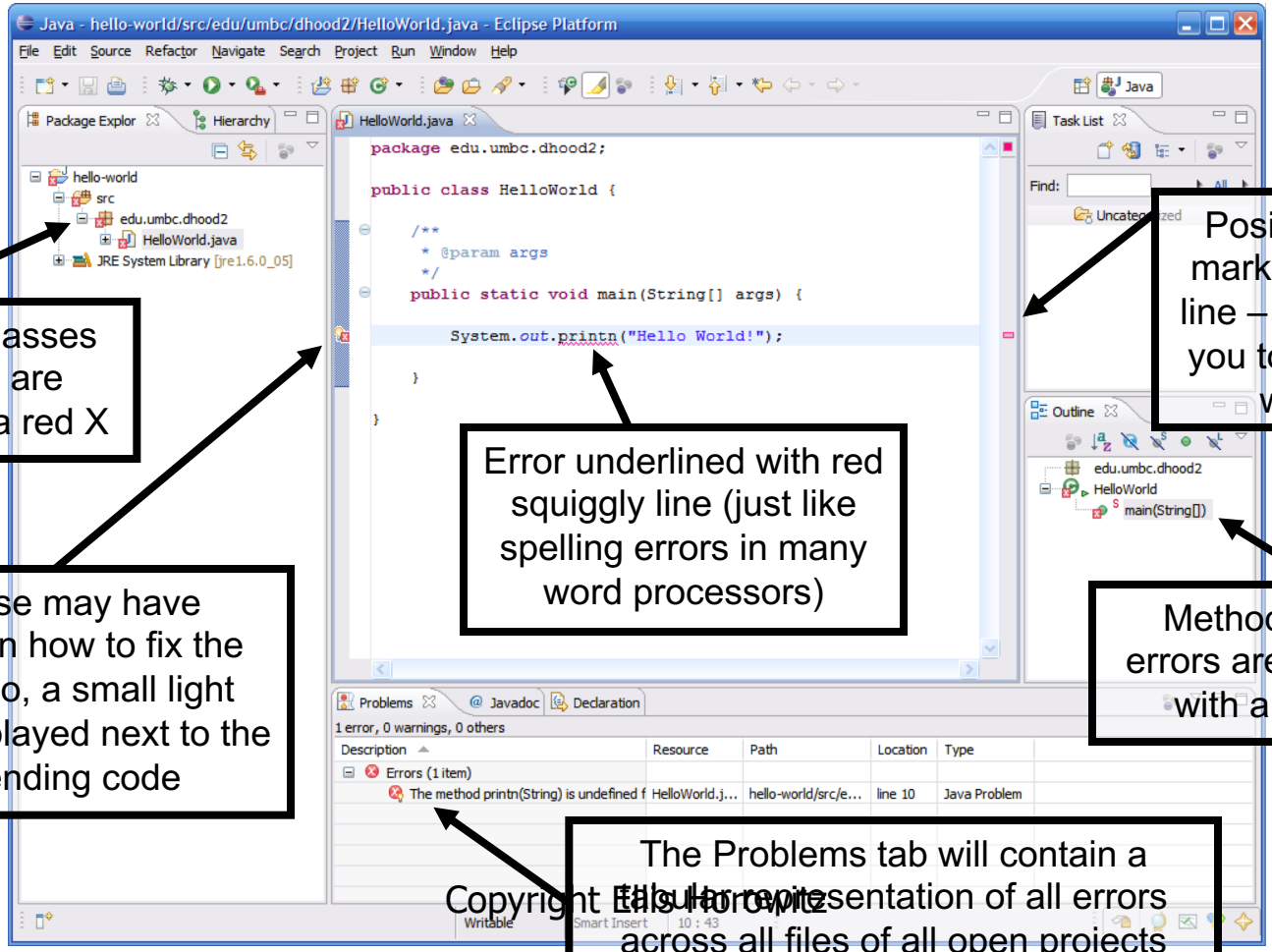


Compiling Source Code

- One important feature of Eclipse is that it automatically compiles your code in the background
- This means that errors can be corrected when made
  - We all know that iterative development is an excellent approach to developing code, but going to shell to do a compile can interrupt the normal course of development
  - You no longer need to go to the command prompt and compile code directly

Example Compilation Error

- This code contains a typo in the println statement…



Packages/Classes with errors are marked with a red X

Position in file is marked with a red line – 1 click allows you to jump to line with error

Often Eclipse may have suggestions on how to fix the problem – if so, a small light bulb will be displayed next to the line of offending code

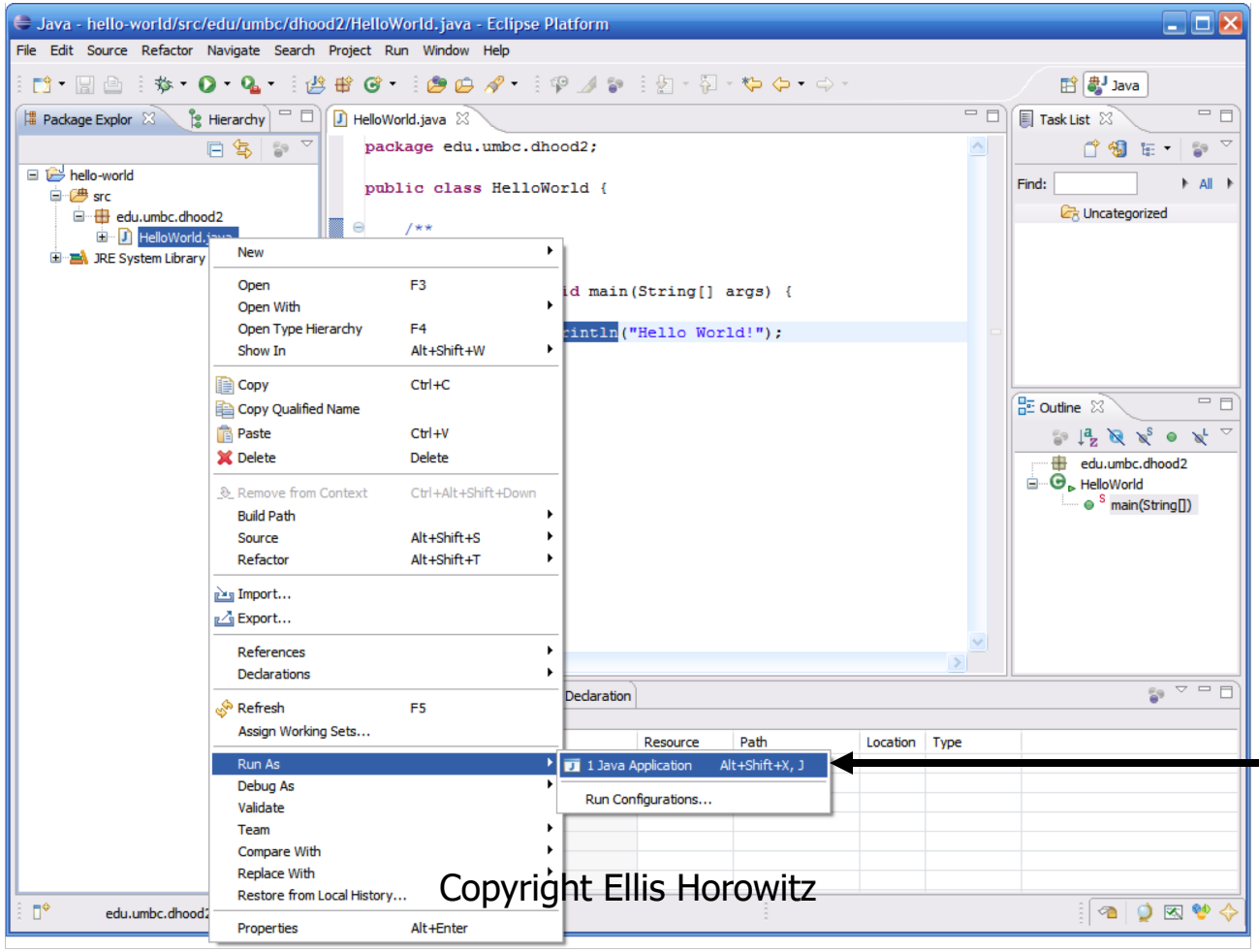
Error underlined with red squiggly line (just like spelling errors in many word processors)

Methods with errors are marked with a red X

The Problems tab will contain a tabular representation of all errors across all files of all open projects

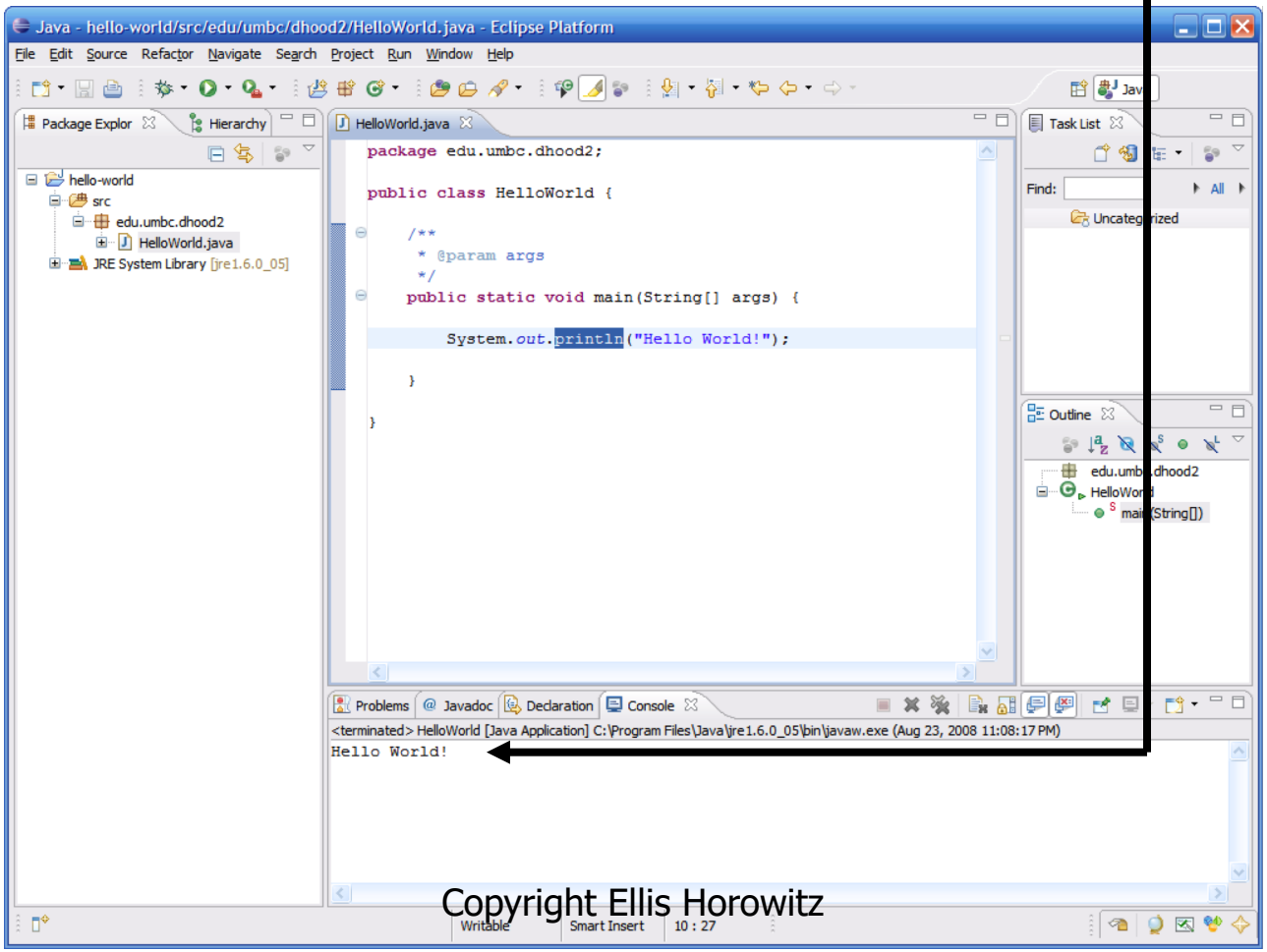
Running Code

- An easy way to run code is to right click on the class and select Run As → Java Application



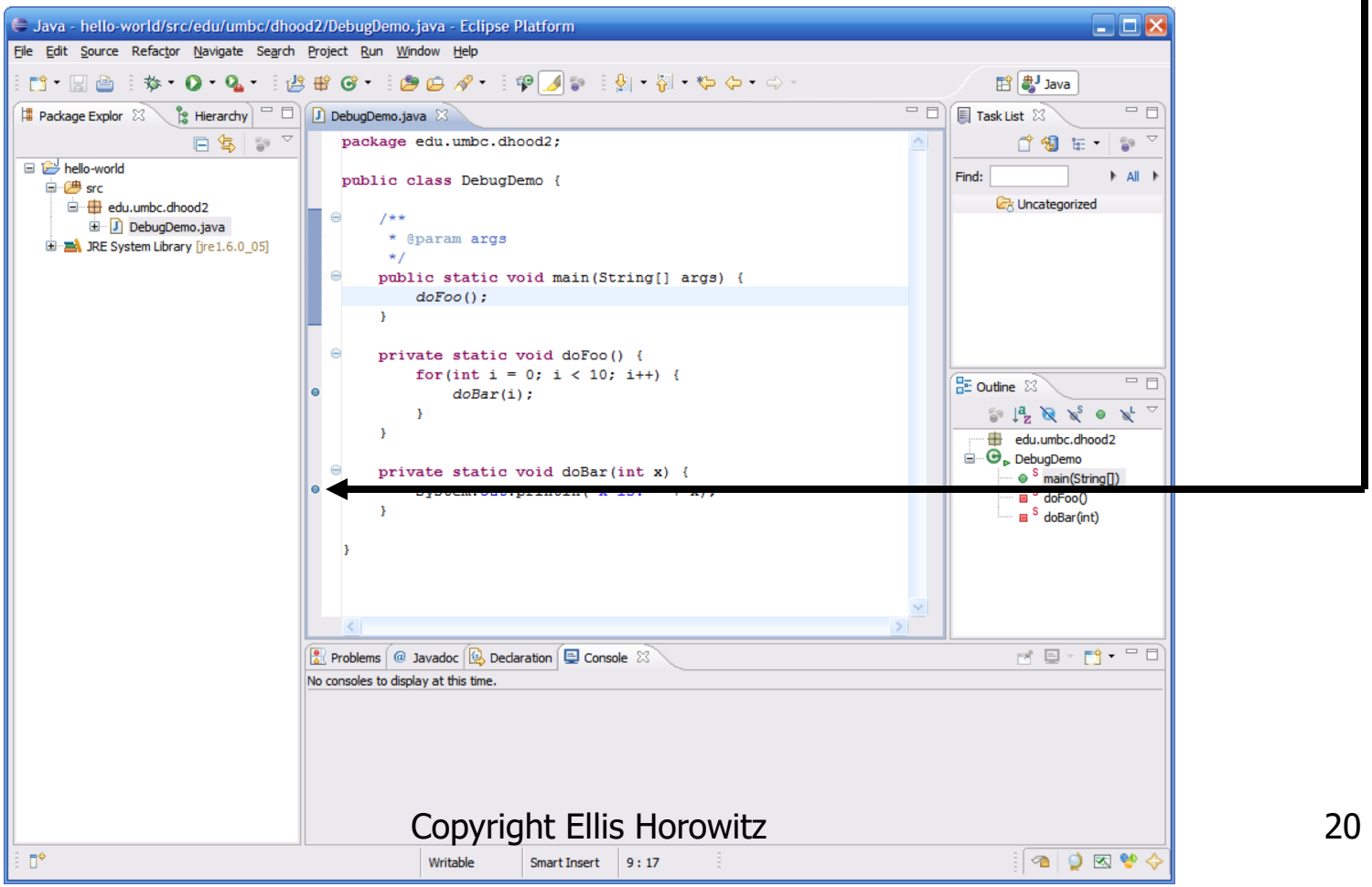
Running Code (continued)

- The output of running the code can be seen in the Console tab in the bottom pane



Debugging Code

- Eclipse comes with a pretty good built-in debugger
- You can set break points in your code by double clicking in the left hand margin – break points are represented by these blue bubbles



End of Eclipse Tutorial

Tools for Surface Web Crawling

- **Command line for issuing http requests**
  - wget, pre-installed in Ubuntu
    - get a single page
    - wget http://www.example.com/index.html
    - support http, ftp etc., e.g.
    - wget ftp://ftp.gnu.org/pub/gnu/wget/wget-latest.tar.gz
  - curl, OSX pre-installed also supports http requests
- **Simple crawling programs**
  - Crawler4j, written in Java
  - Scrapy: http://scrapy.org, written in Python
- **Large-scale crawling programs**
  - Heritrix, crawler for archive.org
  - Nutch, Apache Software Foundation

How To Get a Web Page in Java

```
import java . net .*;
import java . io .*;
public class URLReader {
public static void main(String [] args) throws Exception { } }
  URL oracle = new URL(”http://www.oracle.com/”);
  BufferedReader in = new BufferedReader (
  new InputStreamReader(oracle.openStream()));
  String inputLine ;
  while (( inputLine = in . readLine ()) != null)
     System . out . println ( inputLine );
     in . close ();
   }
}
```

- After you create a URL, you can call the URL's openStream() method to get a stream from which you can read the contents of the URL.
- The openStream() method returns a java.io.InputStream object

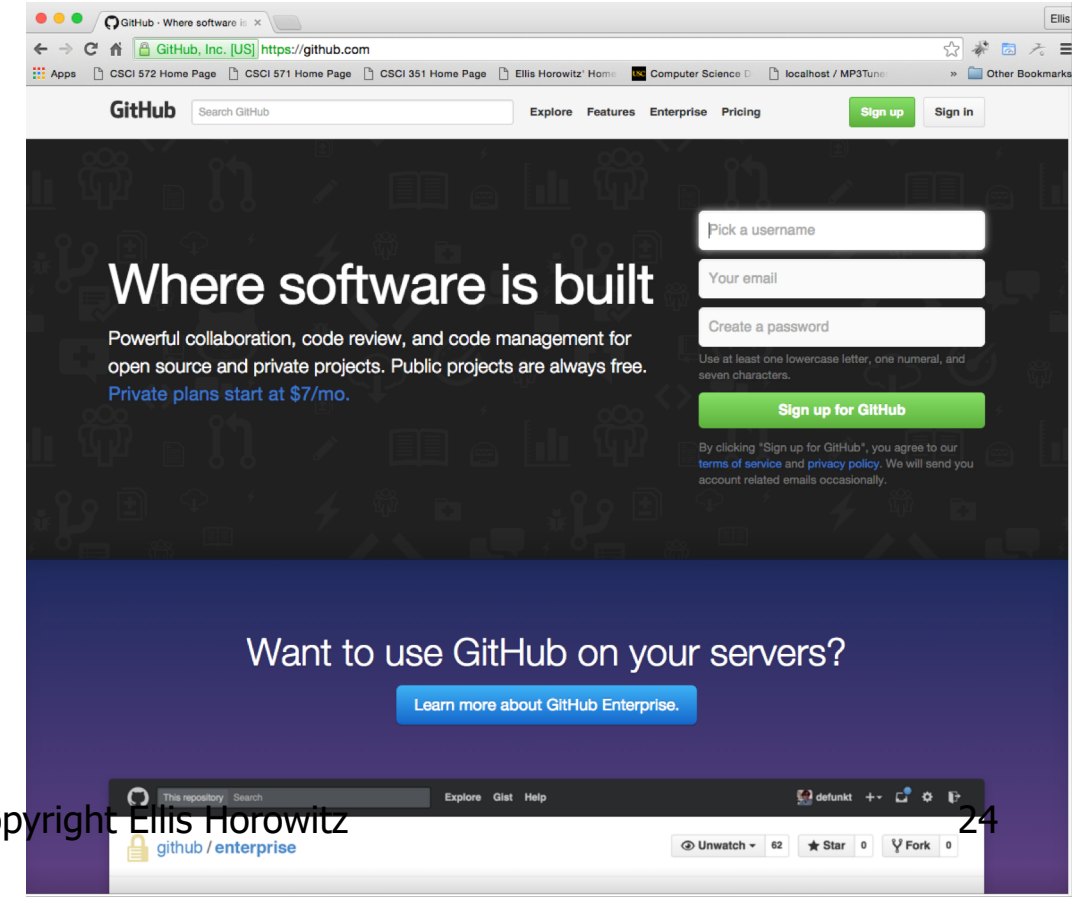
Instructions for Installing Crawler4j

- download crawler4j from github
  - **GitHub** is a web-based repository hosting service for software. Originally the Git system offered distributed revision control and source code management (SCM) functionality, but on the command line; GitHub offers a web interface and some additional features.
  - As of June 2018, GitHub

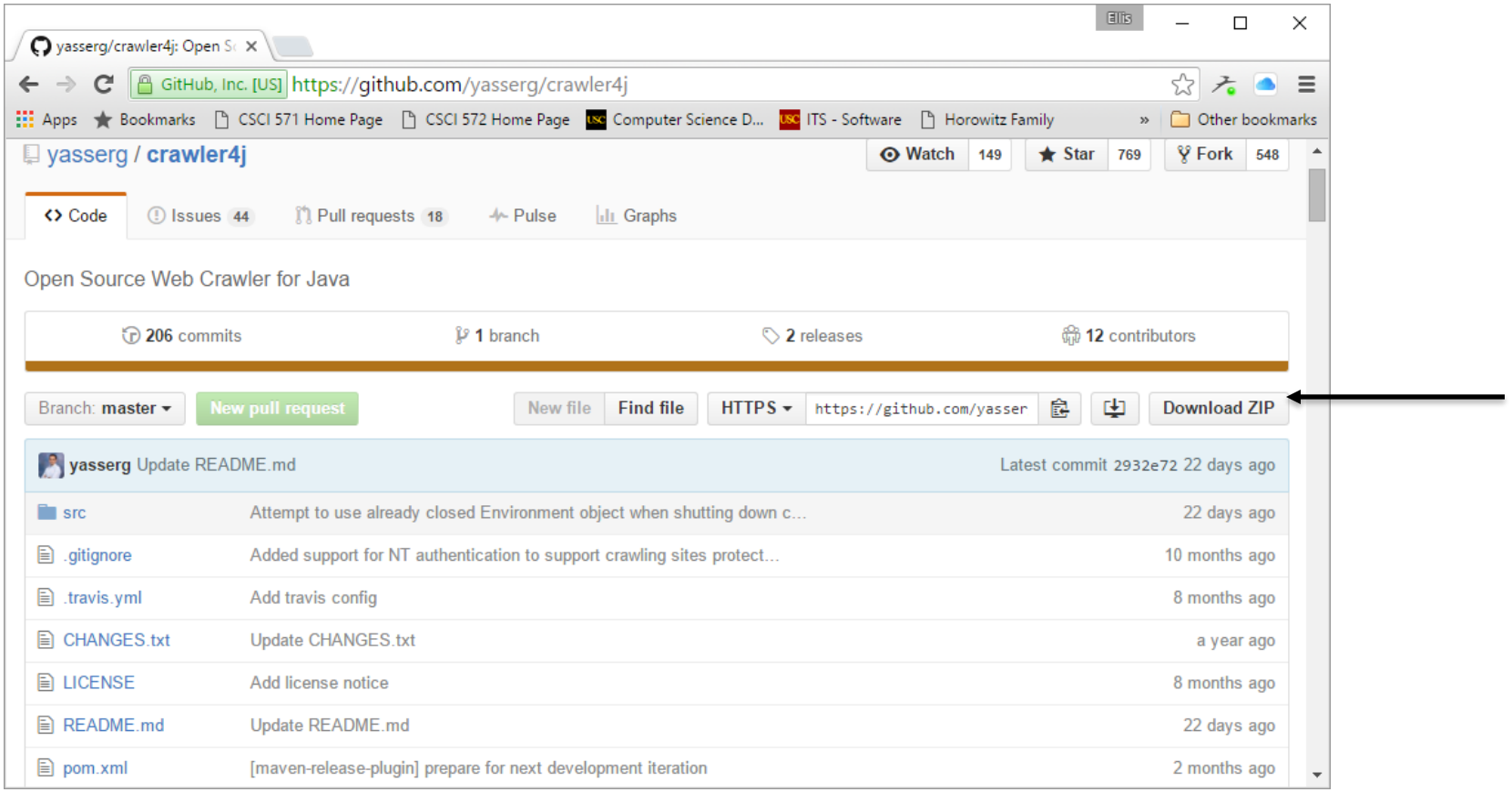
reports having over

28 million users and over

57 million repositories

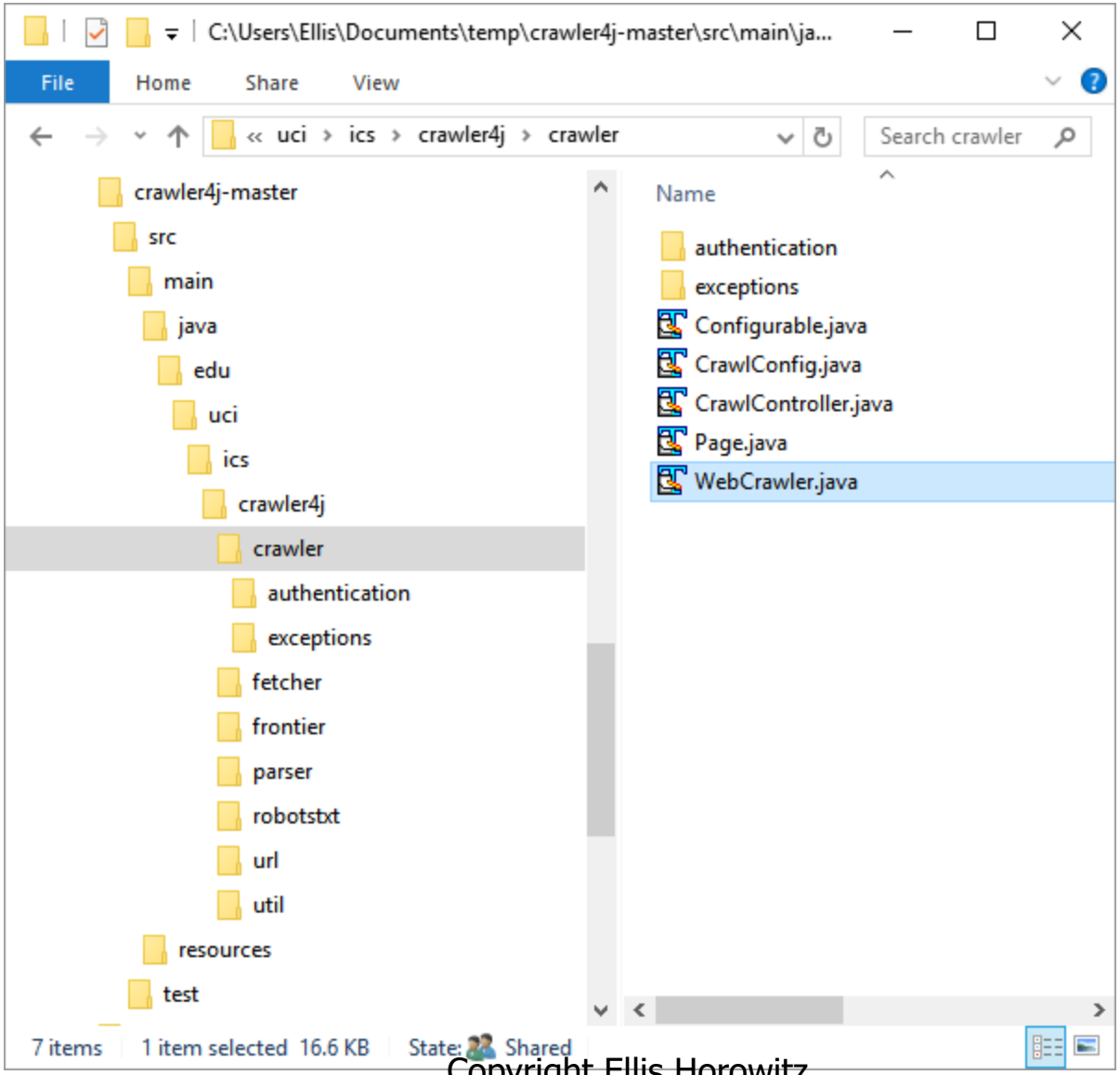


Downloading Crawler4j from GitHub



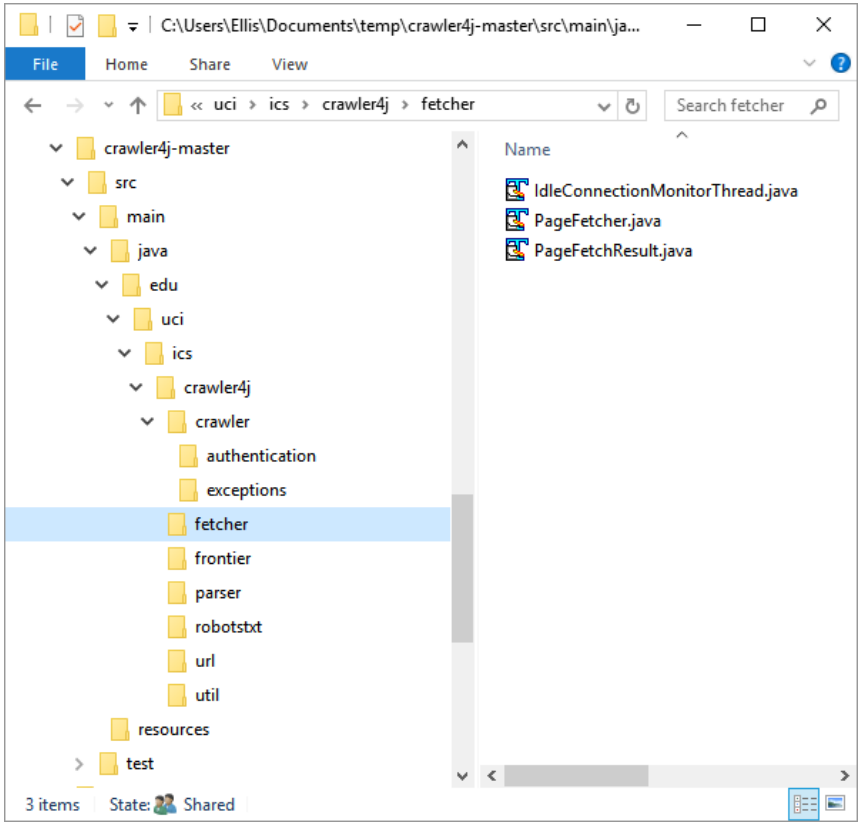
See especially the README file page at
https://github.com/yasserg/crawler4j/blob/master/README.md

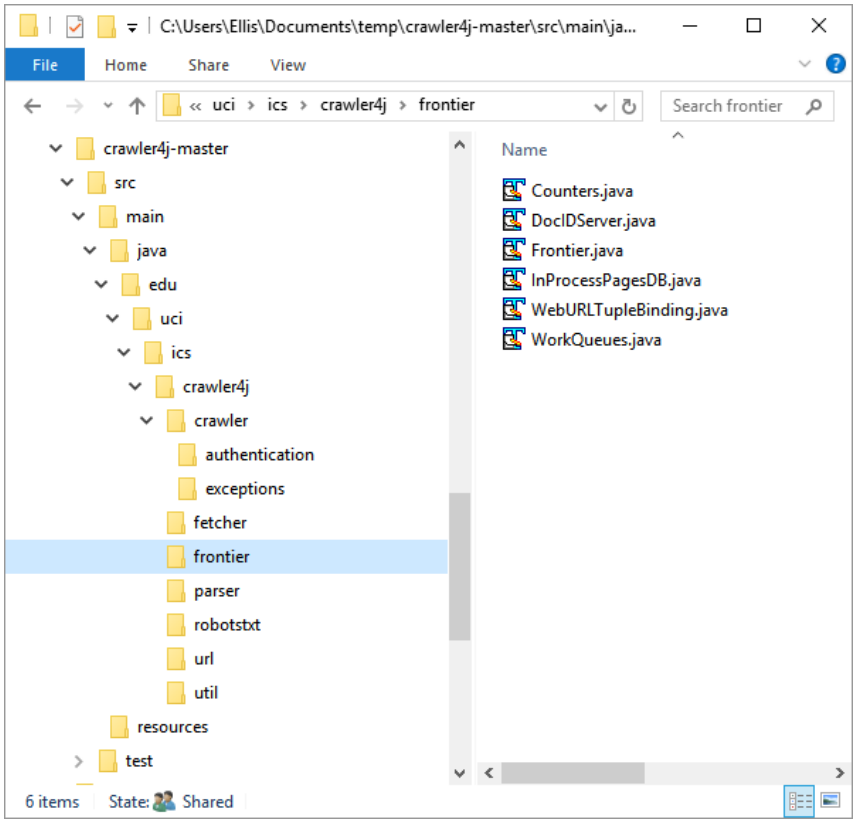
Crawler4j Source Code



Crawler folder, a good place to start; look especially at WebCrawler.java

Crawler4j Source code is Logically Organized into folders

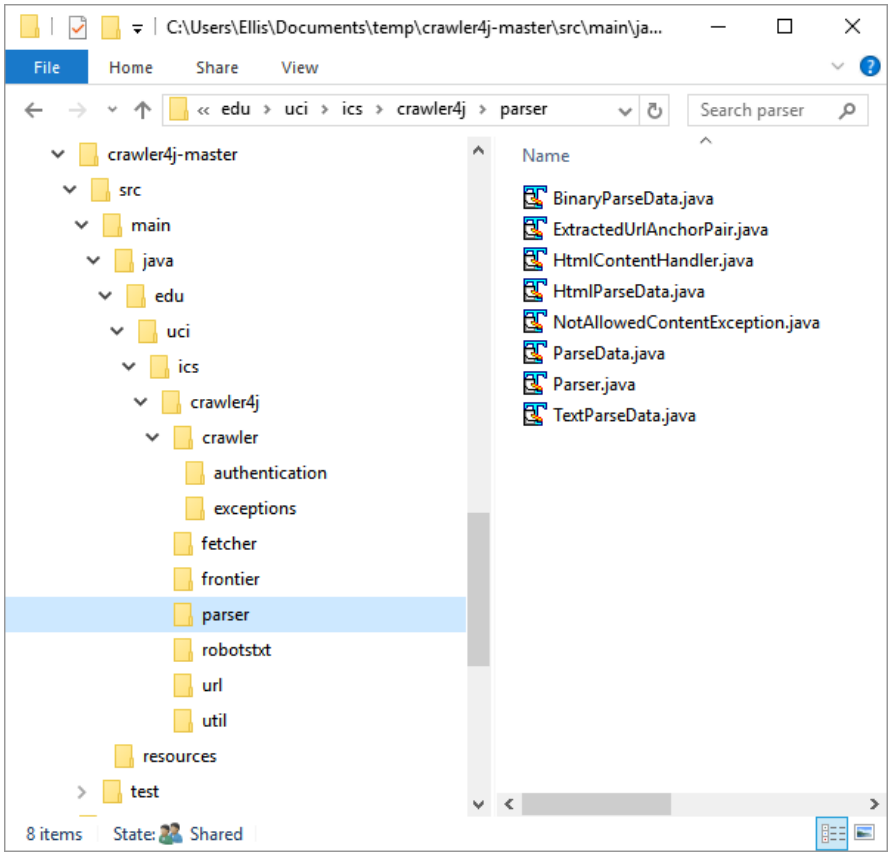


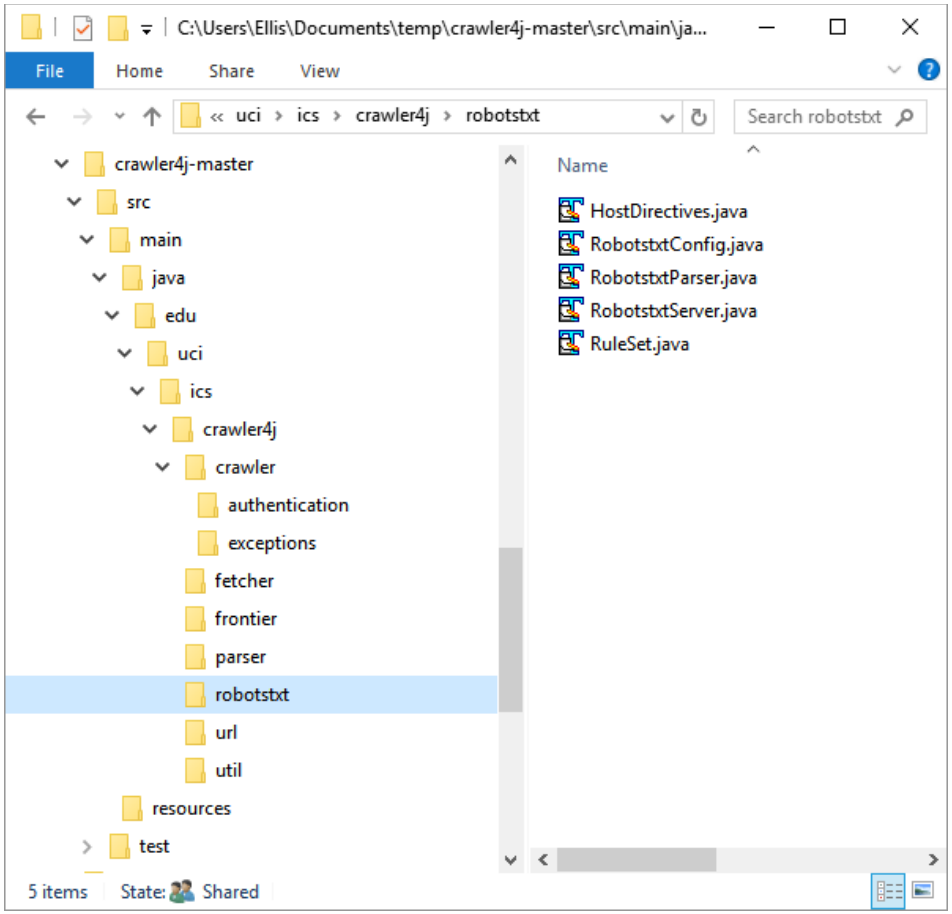


Fetcher Code handles:
- schemes: http, https
- politeness delay;
- redirects;
- max-size settings;
- expired connections

Frontier Code handles:
- statistics database;
- previously seen URLs
- queue of pending URLs

Crawler4j Routines are Named According to their Function

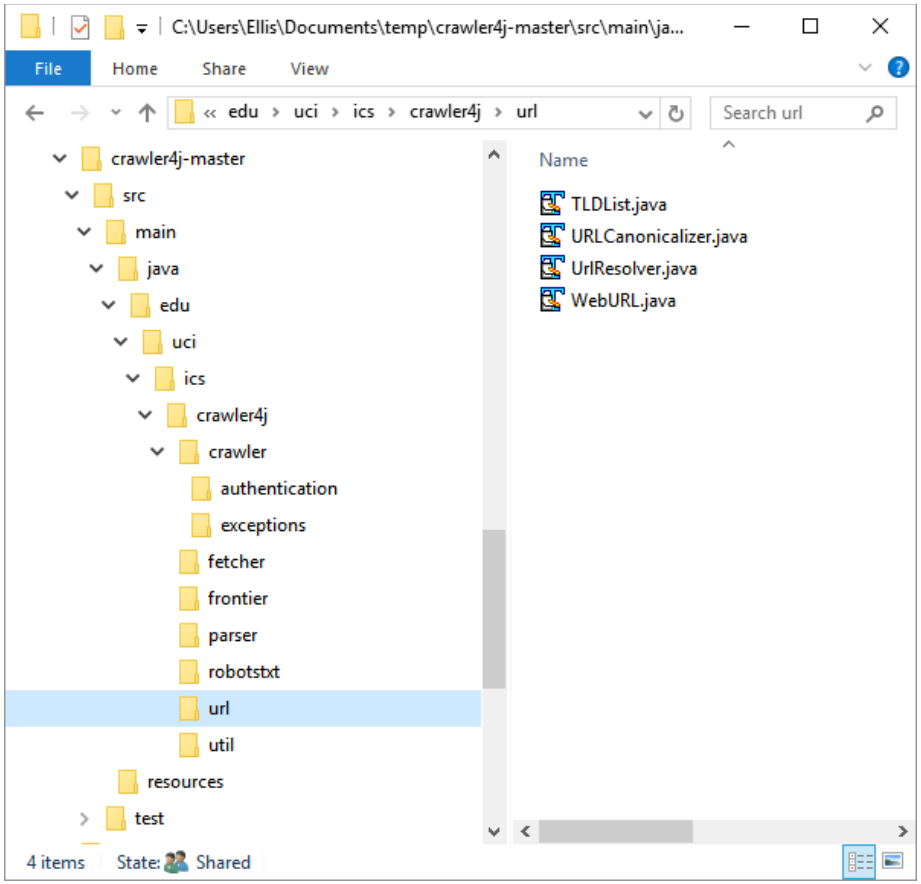


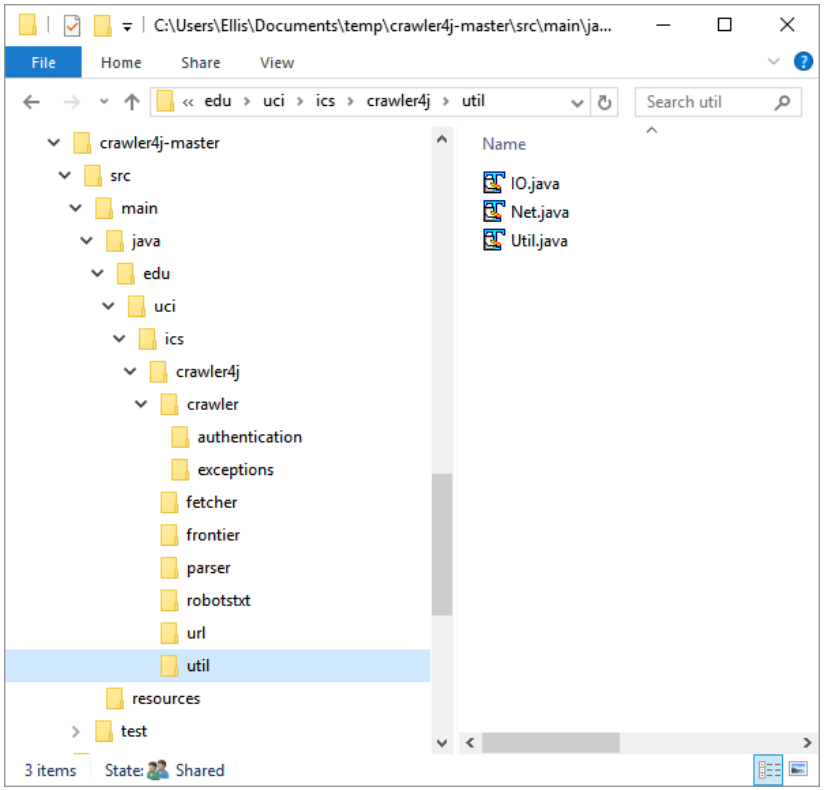


Parser Code handles:
- binary data
- html pages
- extracting links

Robots.txt Code handles:
- fetching and re-fetching robots.txt
- caching robots.txt files
- interpreting commands
- working with Page Fetcher

More crawler4j Source code

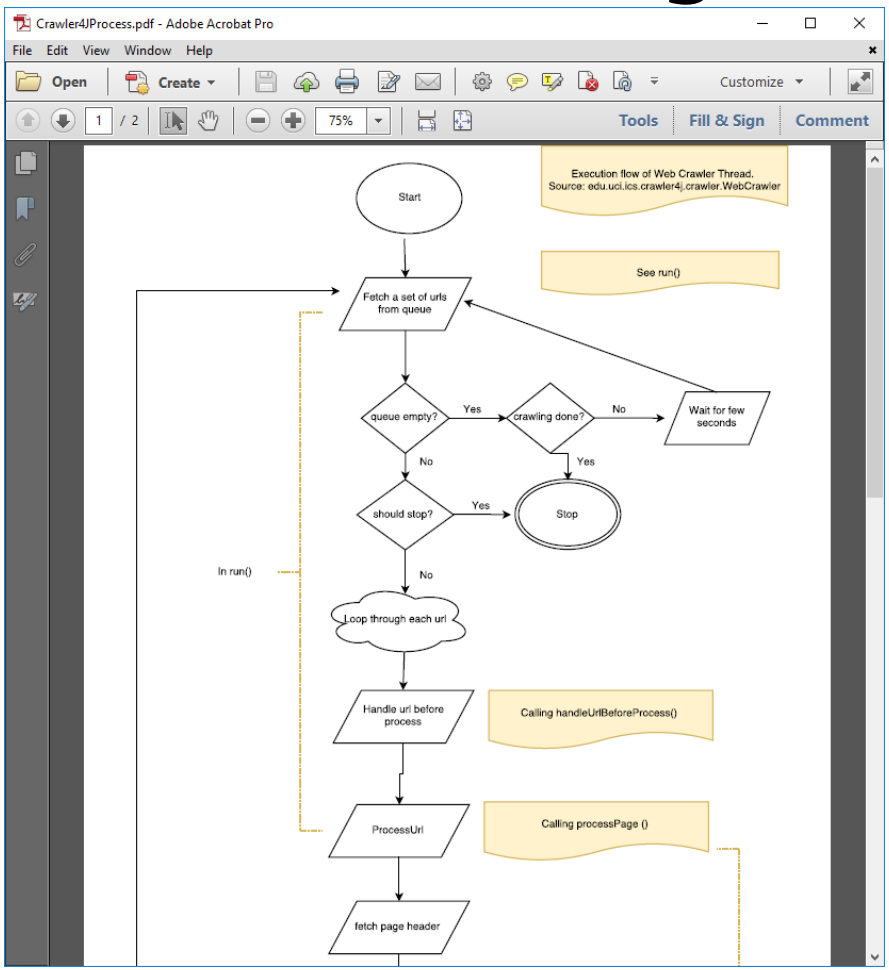


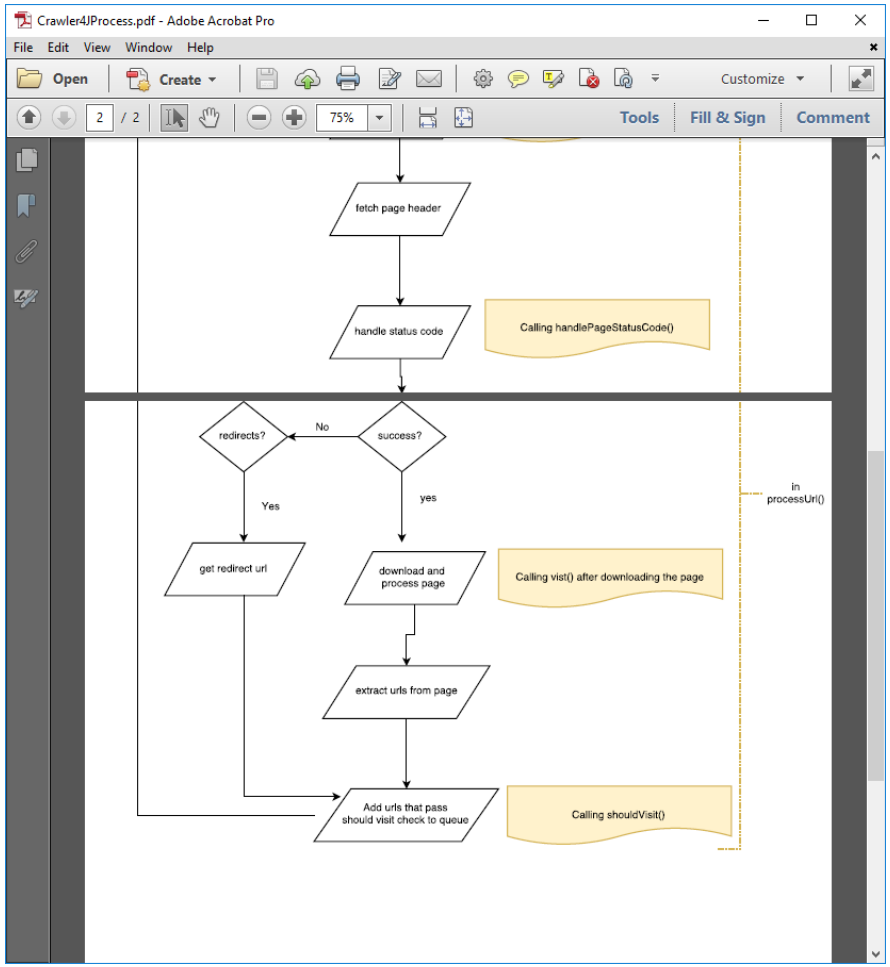


utility routines

URL resolver and canonicalizer handles:
- checking against list of TLDs
- normalizes URL, removes . or .., etc
- alters name/value pairs
- converts #nn values
- evaluates <base>

Logic Flowchart





http://www-scf.usc.edu/~csci572/2018Fall/hw2/Crawler4JProcess.pdf

Configuring the Crawler and Seeding it

```
public class Controller {
    public static void main(String[] args) throws Exception {
        String crawlStorageFolder = "/data/crawl";
        int numberOfCrawlers = 7;
        CrawlConfig config = new CrawlConfig();
        config.setCrawlStorageFolder(crawlStorageFolder);
        /*  Instantiate the controller for this crawl.*/
        PageFetcher pageFetcher = new PageFetcher(config);
        RobotstxtConfig robotstxtConfig = new RobotstxtConfig();
        RobotstxtServer robotstxtServer = new RobotstxtServer(robotstxtConfig, pageFetcher);
        CrawlController controller = new CrawlController(config, pageFetcher, robotstxtServer);
        /*  For each crawl, you need to add some seed urls. These are the first
         * URLs that are fetched and then the crawler starts following links
         * which are found in these pages  */
        controller.addSeed("http://www.latimes.com/”);
        /*  Start the crawl. This is a blocking operation, meaning that your code
         * will reach the line after this only when crawling is finished.  */
        controller.start(MyCrawler.class, numberOfCrawlers);
    }
}
```

folder to store downloads;

#crawlers

set up pagefetcher and robots.txt handler

crawling www.latimes.com

Crawling Pages Other Than HTML

- To make sure you are not just crawling HTML, and missing pdf and doc files, you need to set BinaryContent to true
- Examine the routine BasicCrawlController and see line 81
- Turn config.setIncludeBinaryContentInCrawling(false);

https://github.com/yasserg/crawler4j/blob/master/crawler4j-examples/crawler4j-examples-base/src/test/java/edu/uci/ics/crawler4j/examples/basic/BasicCrawlController.java

Defining Which Pages to Crawl

```
public class MyCrawler extends WebCrawler {
    private final static Pattern FILTERS =
Pattern.compile(".*(\\.(css|js|gif|jpg"  + "|png|mp3|mp3|zip|gz))$");
   /** This method receives two parameters. The first parameter is the page
     * in which we have discovered this new url and the second parameter is
     * the new url. You should implement this function to specify whether
     * the given url should be crawled or not (based on your crawling logic).
     * In this example, we are instructing the crawler to ignore urls that
     * have css, js, git, ... extensions and to only accept urls that start
     * with "http://www.latimes.com/". In this case, we didn't need the
     * referring Page parameter to make the decision.    */
     @Override
     public boolean shouldVisit(Page referringPage, WebURL url) {
         String href = url.getURL().toLowerCase();
         return !FILTERS.matcher(href).matches()
                 && href.startsWith("http://www.latimes.com/");
     }
```

see next slide

Matching URLs

- ".*(\\.(css|js|gif|jpg" + "|png|mp3|mp4|zip|gz))$"

- A regular expression, specified as a string, must first be compiled into an instance of this class.
- a Matcher object that can match arbitrary character sequences against the regular expression
- See https://docs.oracle.com/javase/7/docs/api/java/util/regex/Pattern.html

- In the above there are two strings concatenated by plus; consider the simpler form:
- ".*(\\.(css|js|zip|gz))$"

- . matches any character
- * matches zero or more of preceding character
- \. matches a literal dot
- $ anchors the pattern at the end of the string

Parsing the Downloaded Page

```
/**   This function is called when a page is fetched and ready
      * to be processed by your program.  */
    @Override
    public void visit(Page page) {
        String url = page.getWebURL().getURL();
        System.out.println("URL: " + url);
        if (page.getParseData() instanceof HtmlParseData) {
            HtmlParseData htmlParseData = (HtmlParseData) page.getParseData();
            String text = htmlParseData.getText();
            String html = htmlParseData.getHtml();
            Set<WebURL> links = htmlParseData.getOutgoingUrls();
            System.out.println("Text length: " + text.length());
            System.out.println("Html length: " + html.length());
            System.out.println("Number of outgoing links: " + links.size());
        }
    }
```

The Actual Exercise

- *the URLs it attempts to fetch*, **fetch.csv.** The number of rows should be no more than 20,000 as that is our pre-set limit.
- *the files it successfully downloads*, **visit.csv;** clearly the number of rows will be less than the number of rows in fetch.csv
- *all of the URLs that were discovered* and processed in some way; **urls.csv**. This file could be much larger than 20,000 rows as it will have numerous repeated URLs

Things to Save

- Fetch statistics:
  - # fetches attempted:
    The total number of URLs that the crawler attempted to fetch. This is usually equal to the MAXPAGES setting if the crawler reached that limit; less if the website is smaller than that.
  - # fetches succeeded:
    The number of URLs that were successfully downloaded in their entirety, i.e. returning a HTTP status code of 2XX.
  - # fetches failed or aborted:
    The number of fetches that failed for whatever reason, including, but not limited to: HTTP redirections (3XX), client errors (4XX), server errors (5XX) and other network-related errors.
-

Outgoing URLs

- Outgoing URLs: statistics about URLs extracted from visited HTML pages
  - Total URLs extracted:
    The grand total number of URLs extracted from all visited pages
  - # unique URLs extracted:
    The number of *unique* URLs encountered by the crawler
  - # unique URLs within the news web site:
    The number of *unique* URLs encountered that are associated with the news website,
    i.e. the URL begins with the given root URL of the news website.
  - # unique URLs outside the news website:
    The number of *unique* URLs encountered that were *not* from the website.

Sample Crawl Report for LA Times Using 20,000 as the Download Limit

News site crawled: www.latimes.com/

Fetch Statistics
==========
#fetches attempted: 19951
#fetches succeeded: 19526
#fetches aborted/failed: 425

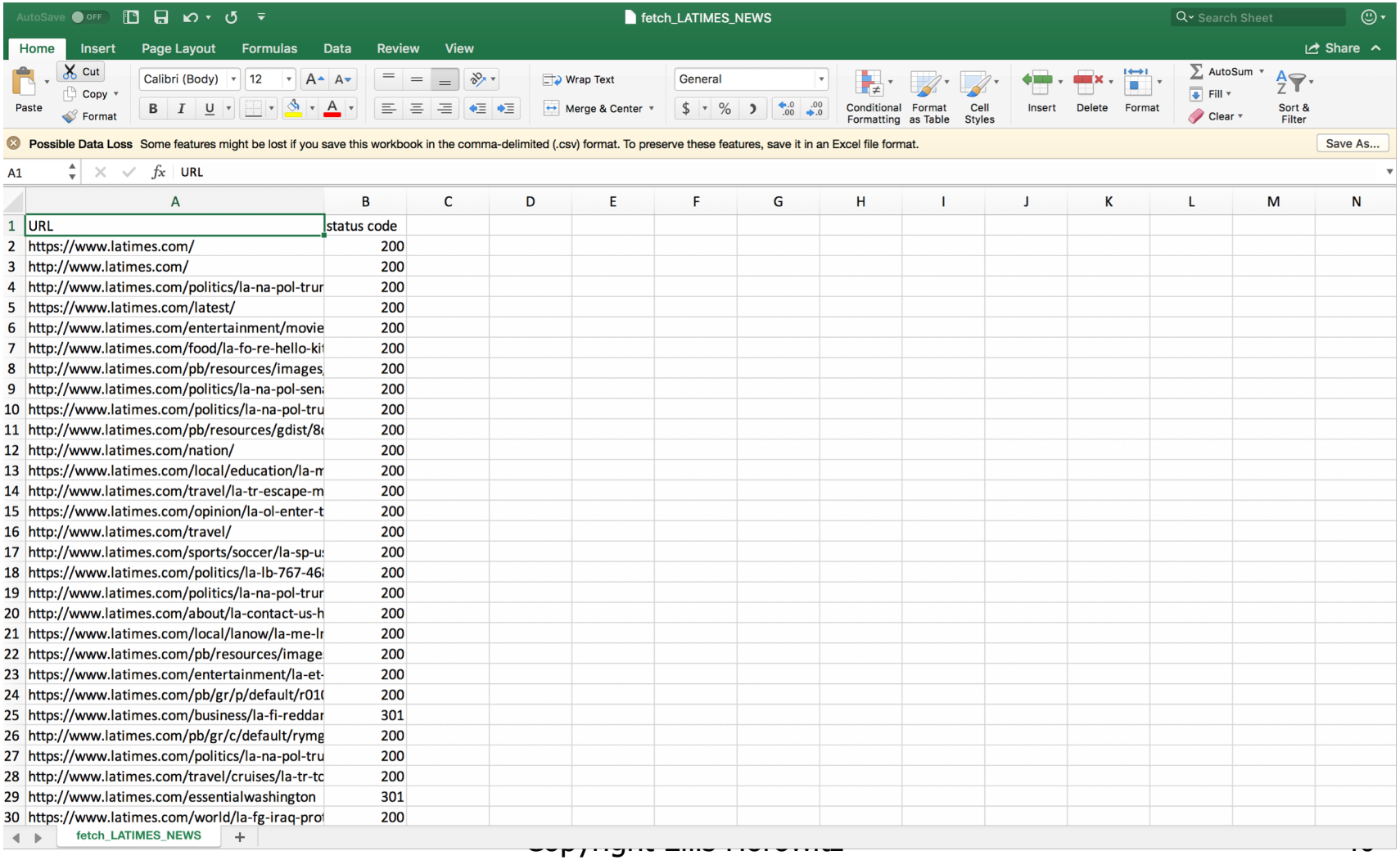
Outgoing URLs
==========
Total URLs extracted: 2510566
#unique URLs extracted: 68568
#unique URLs within News Site: 4880
#unique URLs outside News Site: 63688

Status Codes
==========
200 OK: 19483
301 Moved Permanently: 389
302 Found: 6
404 Not Found: 30

File Sizes:
==========
< 1KB: 106
1KB-10KB: 135
10KB-100KB: 2869
100KB - 1MB: 16399
>=1MB: 10

Content Types:
==========
text/html : 16837
image/png : 193
image/jpeg : 2453
text/css : 8
image/gif : 28

Sample Fetch File for LA Times

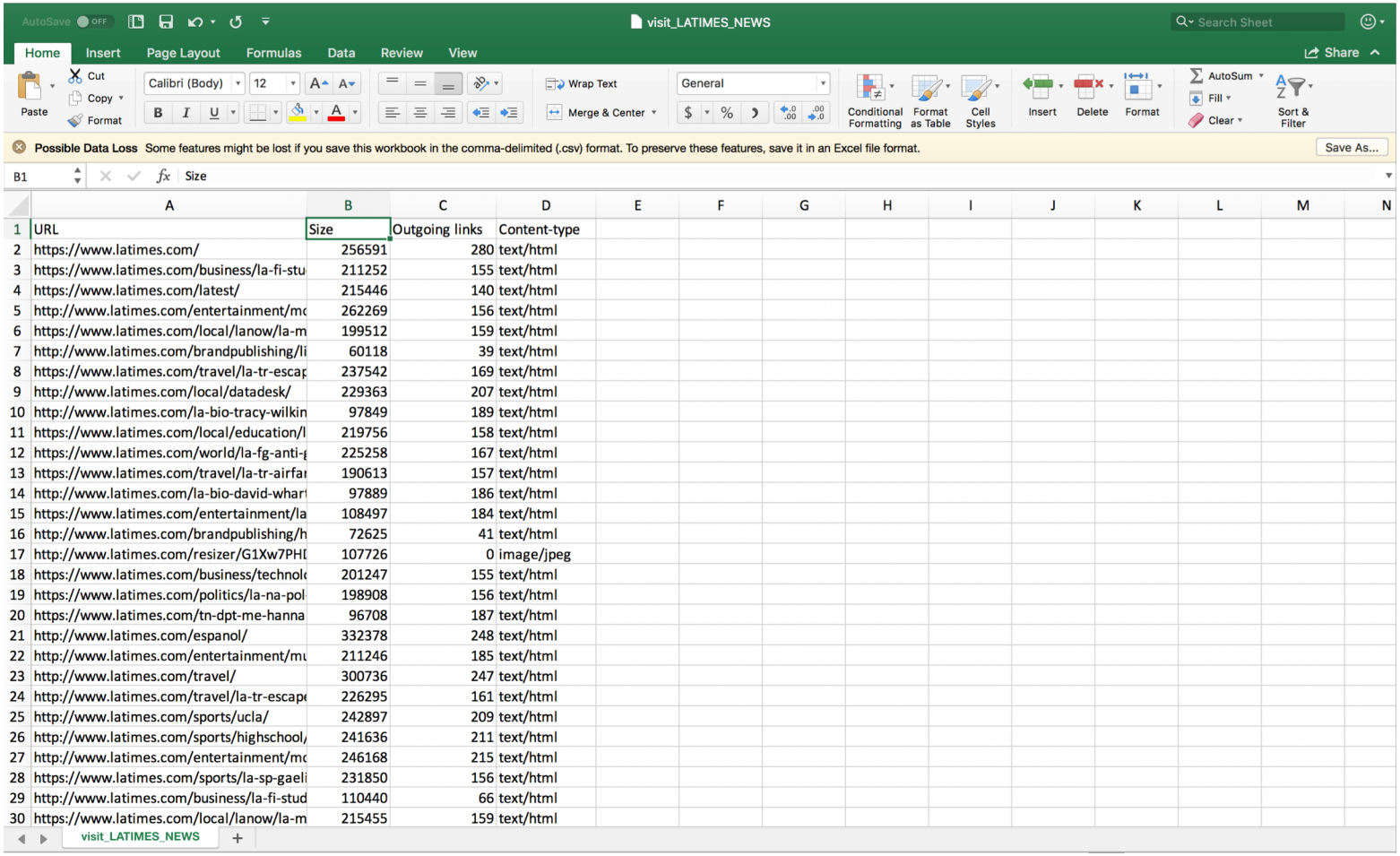


fetch_LATIMES_NEWS

Possible Data Loss Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B
1	URL	status code
2	https://www.latimes.com/	200
3	http://www.latimes.com/	200
4	http://www.latimes.com/politics/la-na-pol-trur	200
5	https://www.latimes.com/latest/	200
6	http://www.latimes.com/entertainment/movie	200
7	http://www.latimes.com/food/la-fo-re-hello-ki	200
8	http://www.latimes.com/pb/resources/images	200
9	http://www.latimes.com/politics/la-na-pol-sen	200
10	https://www.latimes.com/politics/la-na-pol-tru	200
11	http://www.latimes.com/pb/resources/gdist/8	200
12	http://www.latimes.com/nation/	200
13	https://www.latimes.com/local/education/la-m	200
14	http://www.latimes.com/travel/la-tr-escape-m	200
15	https://www.latimes.com/opinion/la-ol-enter-t	200
16	http://www.latimes.com/travel/	200
17	http://www.latimes.com/sports/soccer/la-sp-u	200
18	https://www.latimes.com/politics/la-lb-767-46	200
19	http://www.latimes.com/politics/la-na-pol-trur	200
20	http://www.latimes.com/about/la-contact-us-h	200
21	https://www.latimes.com/local/lanow/la-me-l	200
22	https://www.latimes.com/pb/resources/image	200
23	https://www.latimes.com/entertainment/la-et-	200
24	https://www.latimes.com/pb/gr/p/default/r01	200
25	https://www.latimes.com/business/la-fi-reddar	301
26	http://www.latimes.com/pb/gr/c/default/rymg	200
27	https://www.latimes.com/politics/la-na-pol-tru	200
28	http://www.latimes.com/travel/cruises/la-tr-tc	200
29	http://www.latimes.com/essentialwashington	301
30	https://www.latimes.com/world/la-fg-iraq-pro	200

Sample Visit File for LA Times



visit_LATIMES_NEWS

Possible Data Loss Some features might be lost if you save this workbook in the comma-delimited (.csv) format. To preserve these features, save it in an Excel file format.

	A	B	C	D
1	URL	Size	Outgoing links	Content-type
2	https://www.latimes.com/	256591	280	text/html
3	https://www.latimes.com/business/la-fi-stu	211252	155	text/html
4	https://www.latimes.com/latest/	215446	140	text/html
5	http://www.latimes.com/entertainment/mc	262269	156	text/html
6	https://www.latimes.com/local/lanow/la-m	199512	159	text/html
7	http://www.latimes.com/brandpublishing/li	60118	39	text/html
8	https://www.latimes.com/travel/la-tr-escap	237542	169	text/html
9	http://www.latimes.com/local/datadesk/	229363	207	text/html
10	http://www.latimes.com/la-bio-tracy-wilkin	97849	189	text/html
11	https://www.latimes.com/local/education/l	219756	158	text/html
12	https://www.latimes.com/world/la-fg-anti-g	225258	167	text/html
13	https://www.latimes.com/travel/la-tr-airfar	190613	157	text/html
14	http://www.latimes.com/la-bio-david-whart	97889	186	text/html
15	https://www.latimes.com/entertainment/la	108497	184	text/html
16	http://www.latimes.com/brandpublishing/h	72625	41	text/html
17	http://www.latimes.com/resizer/G1Xw7PH0	107726	0	image/jpeg
18	https://www.latimes.com/business/technol	201247	155	text/html
19	https://www.latimes.com/politics/la-na-pol	198908	156	text/html
20	https://www.latimes.com/tn-dpt-me-hanna	96708	187	text/html
21	http://www.latimes.com/espanol/	332378	248	text/html
22	http://www.latimes.com/entertainment/mu	211246	185	text/html
23	http://www.latimes.com/travel/	300736	247	text/html
24	http://www.latimes.com/travel/la-tr-escape	226295	161	text/html
25	http://www.latimes.com/sports/ucla/	242897	209	text/html
26	http://www.latimes.com/sports/highschool/	241636	211	text/html
27	http://www.latimes.com/entertainment/mc	246168	215	text/html
28	https://www.latimes.com/sports/la-sp-gaeli	231850	156	text/html
29	http://www.latimes.com/business/la-fi-stud	110440	66	text/html
30	https://www.latimes.com/local/lanow/la-m	215455	159	text/html

What to Submit

- Compress all of the above into a single zip archive and name it:

  crawl.zip
- Use only standard zip format. Do **NOT** use other formats such as zipx, rar, ace, etc. For example the zip file might contain the following three files:

1. CrawlReport_LATimes.txt,
2. fetch_LATimes.csv
3. visit_LATimes.csv

- To submit your file electronically to the csci572 account enter the following command from your UNIX prompt:
- $ submit -user csci572 -tag hw2 crawl.zip