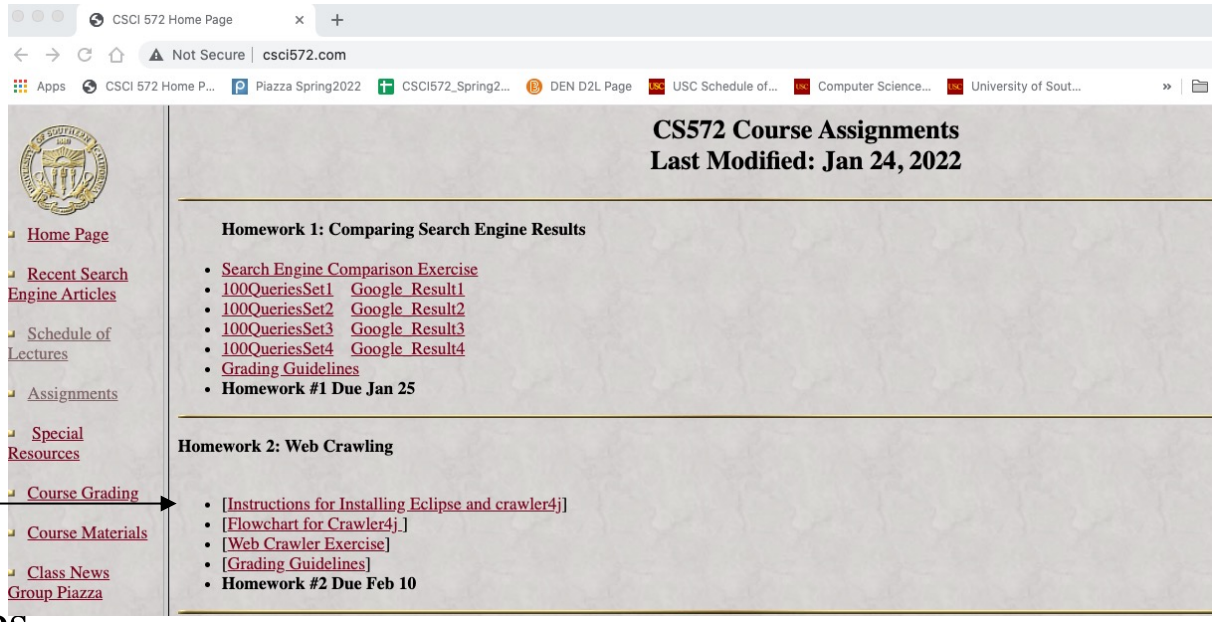
HW2 Assignment



- Involves
  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?

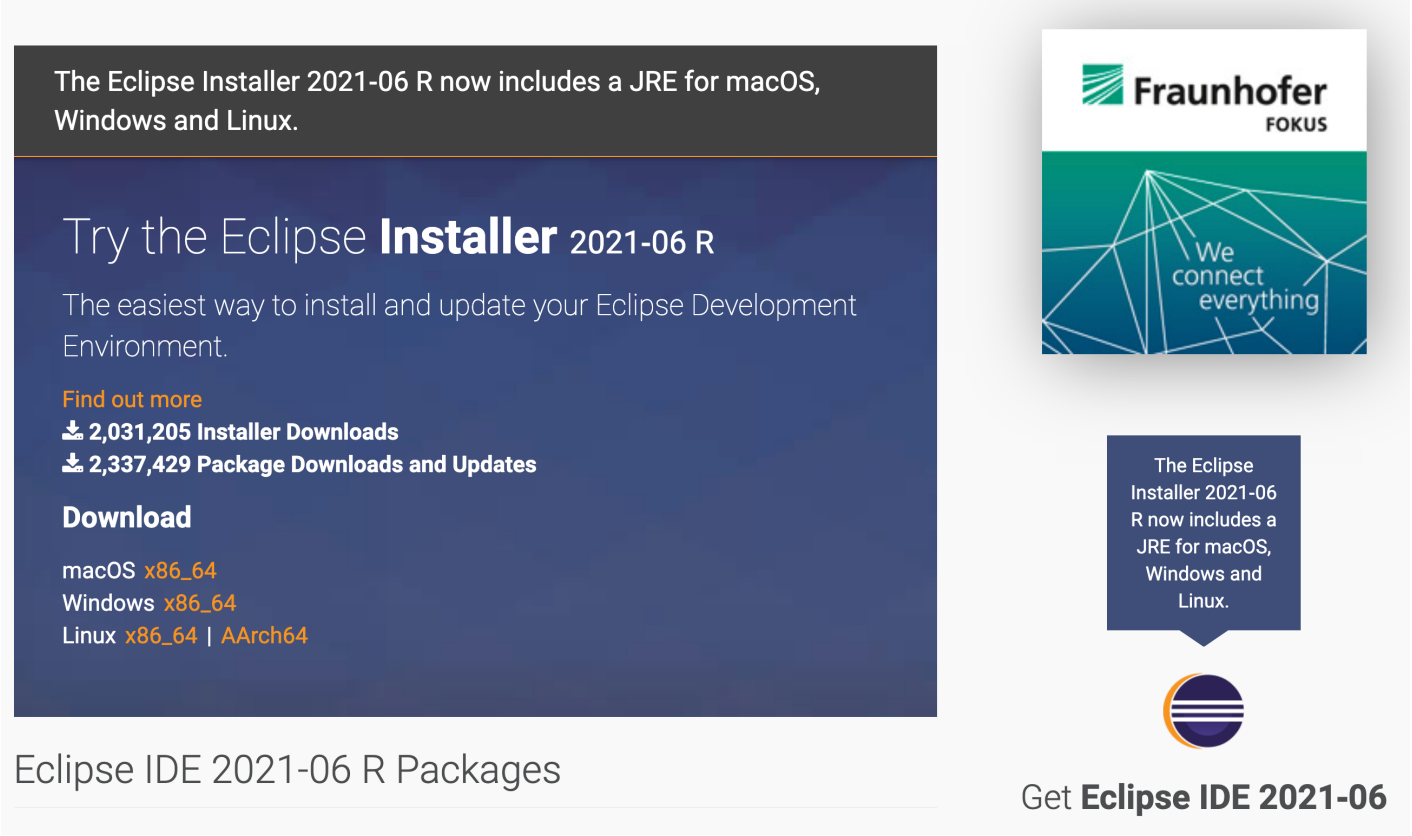
- Eclipse started as a proprietary IBM product (IBM Visual age for Smalltalk/Java)
  - Embracing the open source model IBM opened the product up
- Open Source
  - It is a general purpose open platform that facilitates and encourages the development of third party plug-ins
- Best known as an Integrated 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…
  https://www.eclipse.org/downloads/packages/
- Eclipse comes bundled as a zip file (Windows) or a tarball (all other operating systems)
- Eclipse version to Install - Eclipse IDE for Java Developers



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://csci572.com/2022Fall/hw2/Crawler4jinstallation.pdf

describes the installation for both 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

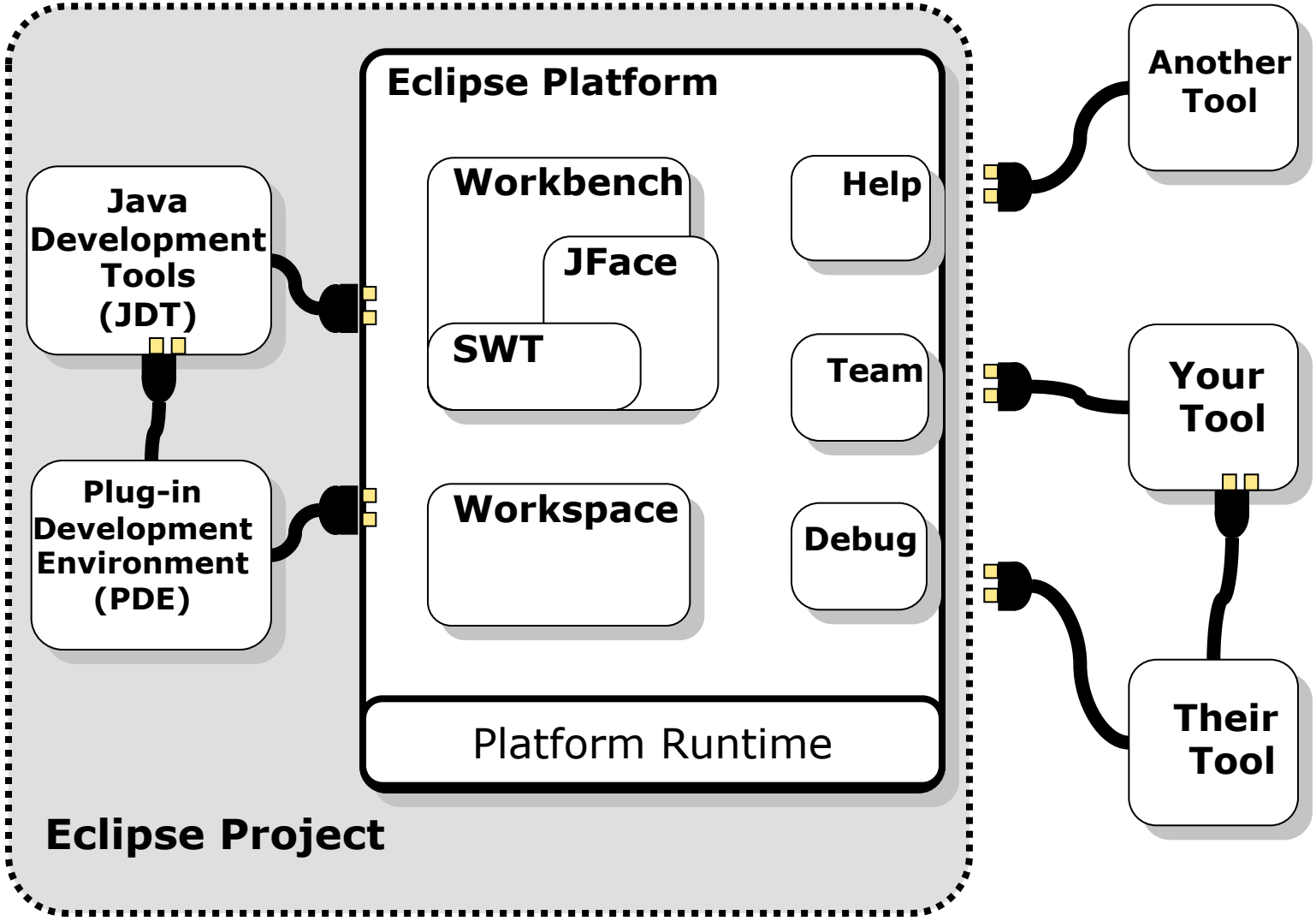
Eclipse Structure

» The following components constitute the rich client platform:

- Core platform - boot Eclipse, run plug-ins
- OSGi - a standard bundling framework
- the Standard Widget Toolkit (SWT) - a portable widget toolkit
- JFace - file buffers, text handling, text editors
- The Eclipse Workbench - views, editors, perspectives, wizards

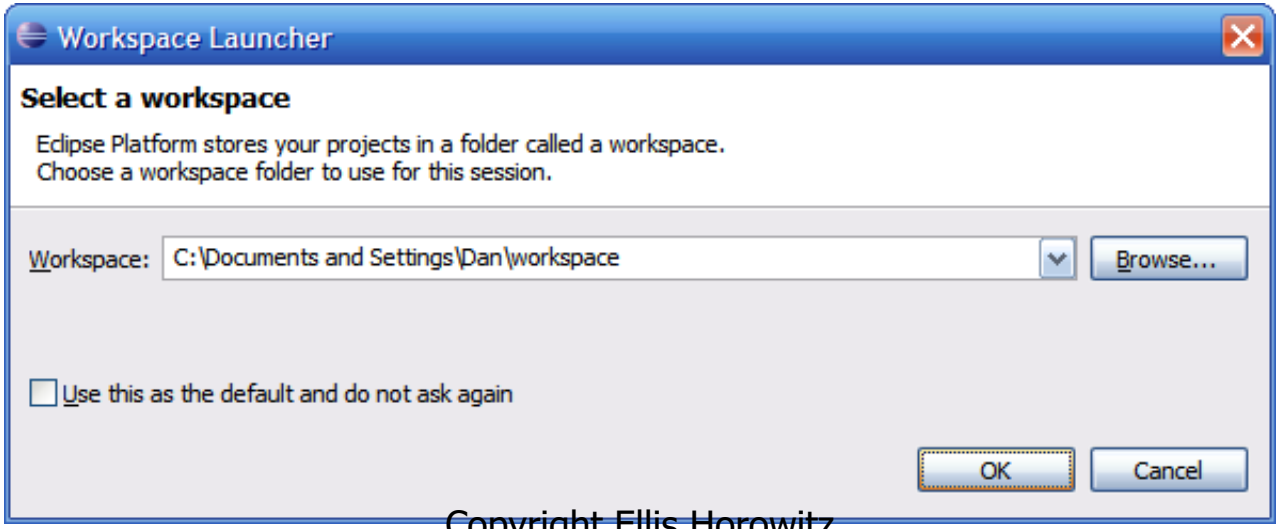
Eclipse Platform is the common base
Consists of several key components

Eclipse Overview



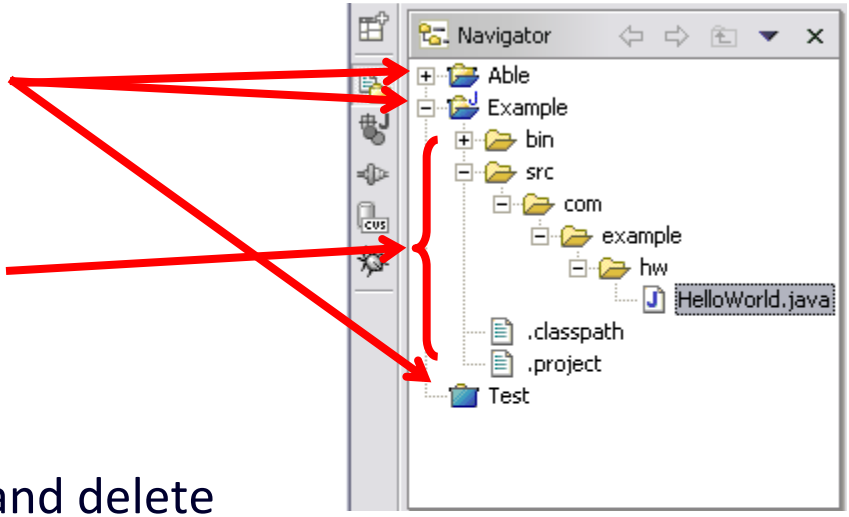
Selecting a Workspace

- 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



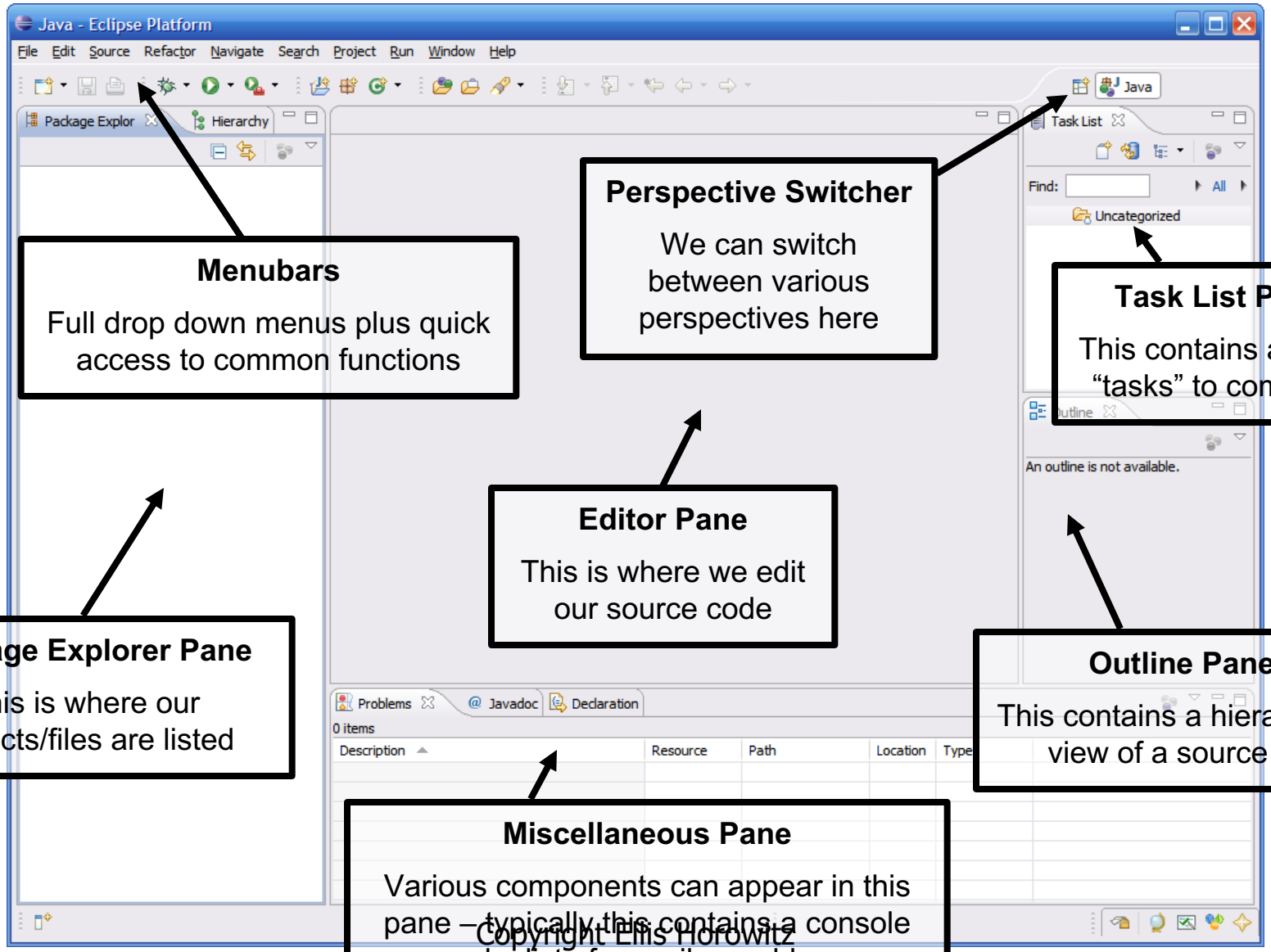
Workspace Component

- Tools operate on files in user's **workspace**
- Workspace holds 1 or more top-level **projects**
- Projects map to directories in file system
- Tree of **folders** and **files**
- {Files, Folders, Projects} termed **resources**



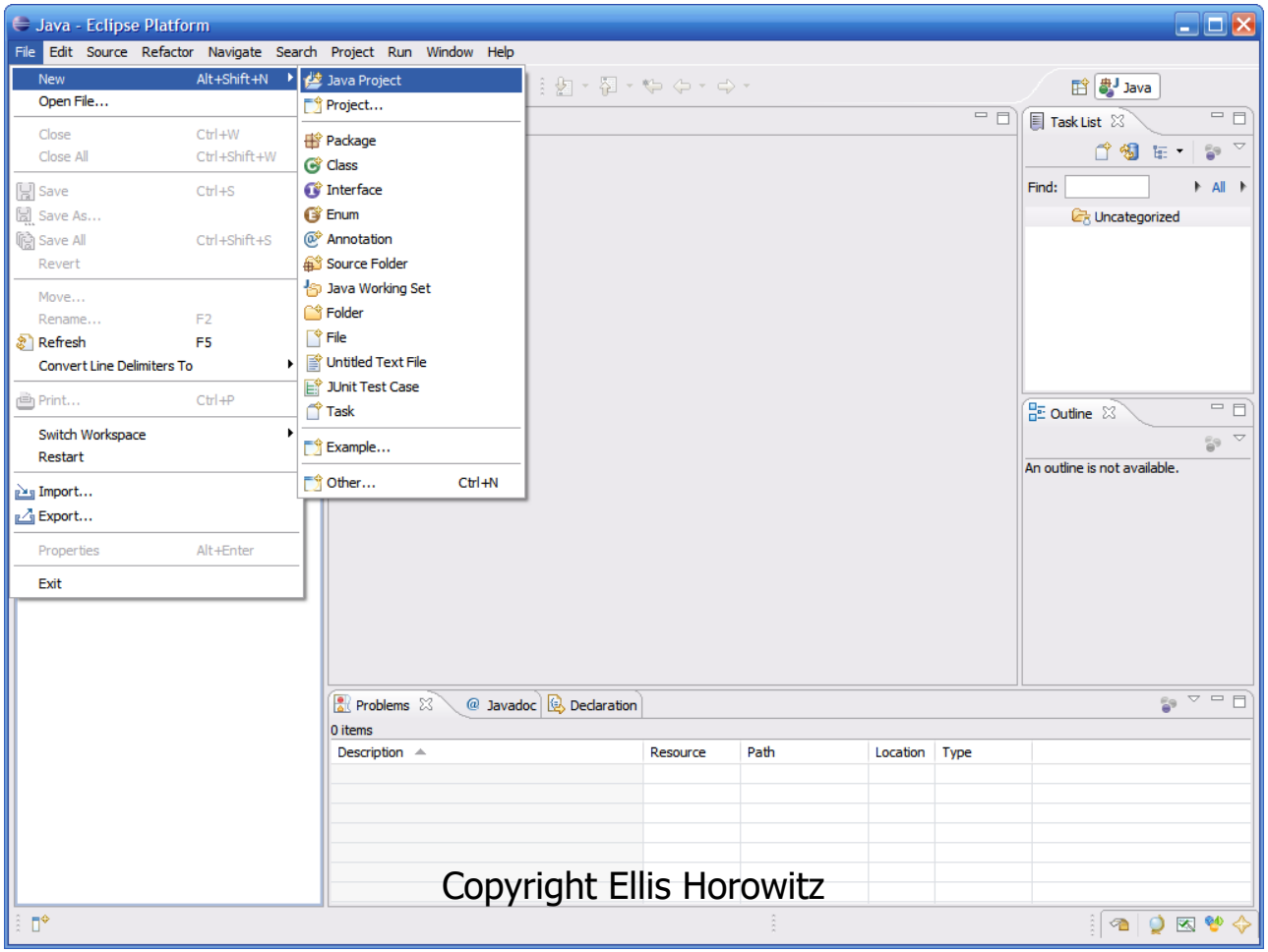
- Tools read, create, modify, and delete resources in workspace
- Plug-ins access via workspace and resource APIs

Eclipse IDE Components

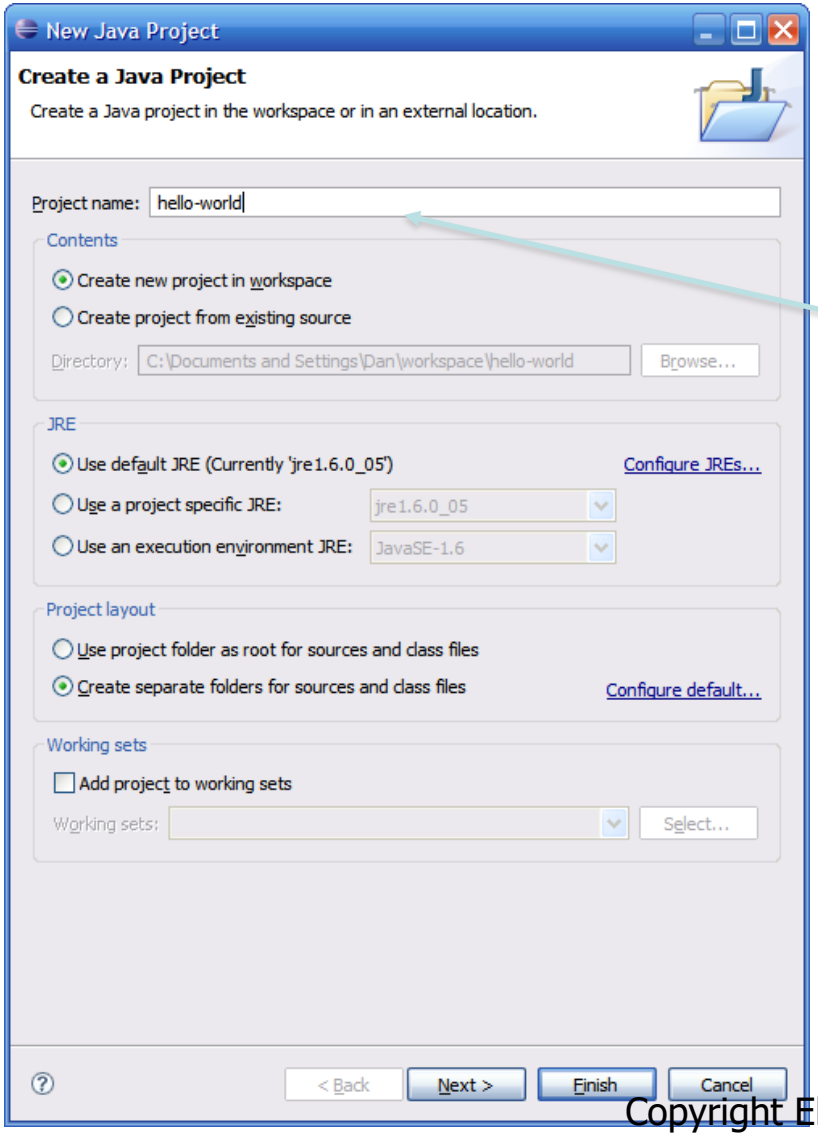


Creating a New Project

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



Creating a New Project (continued)

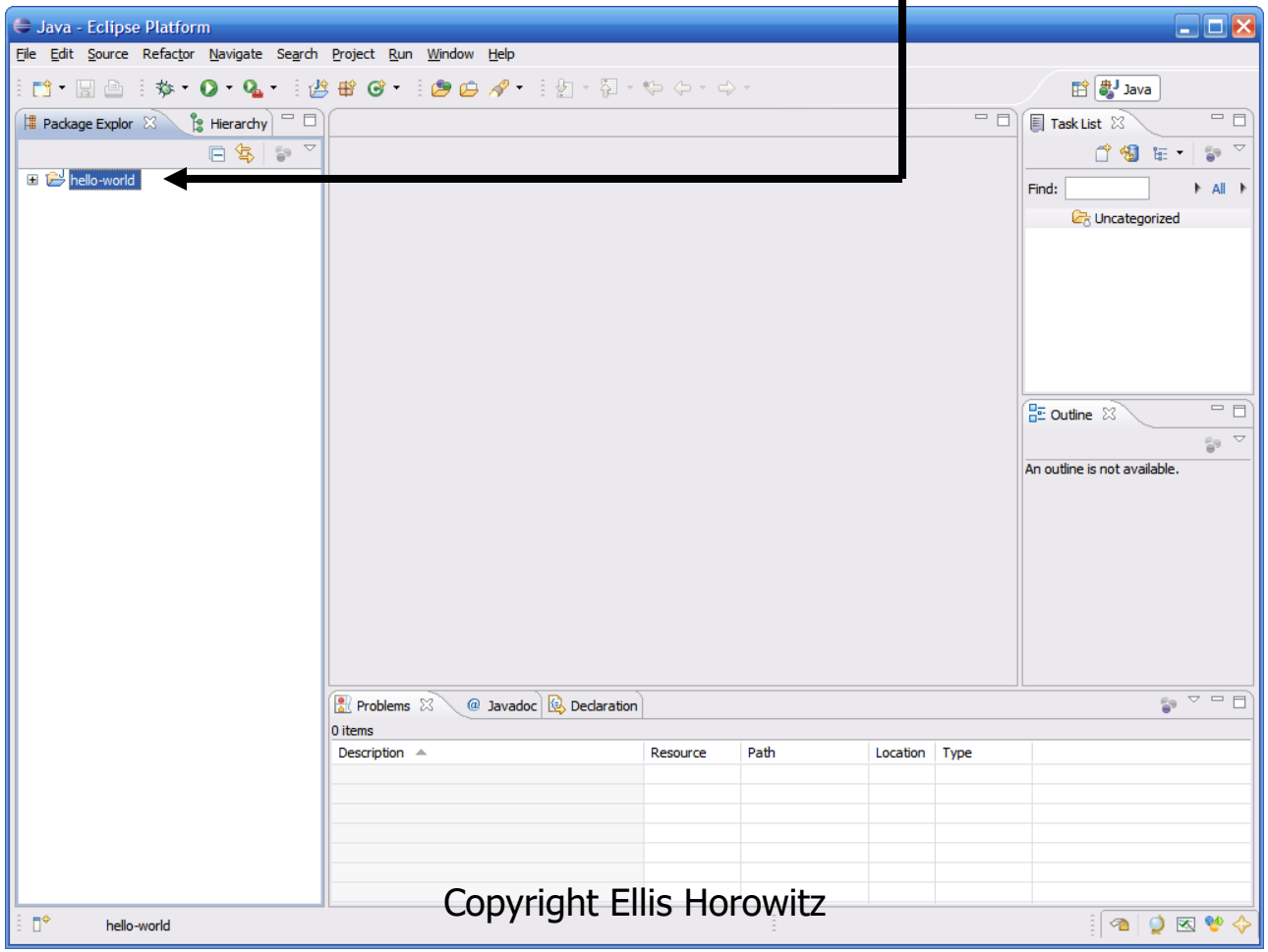


- Enter a name for the project, then click Finish

Hello-world Project

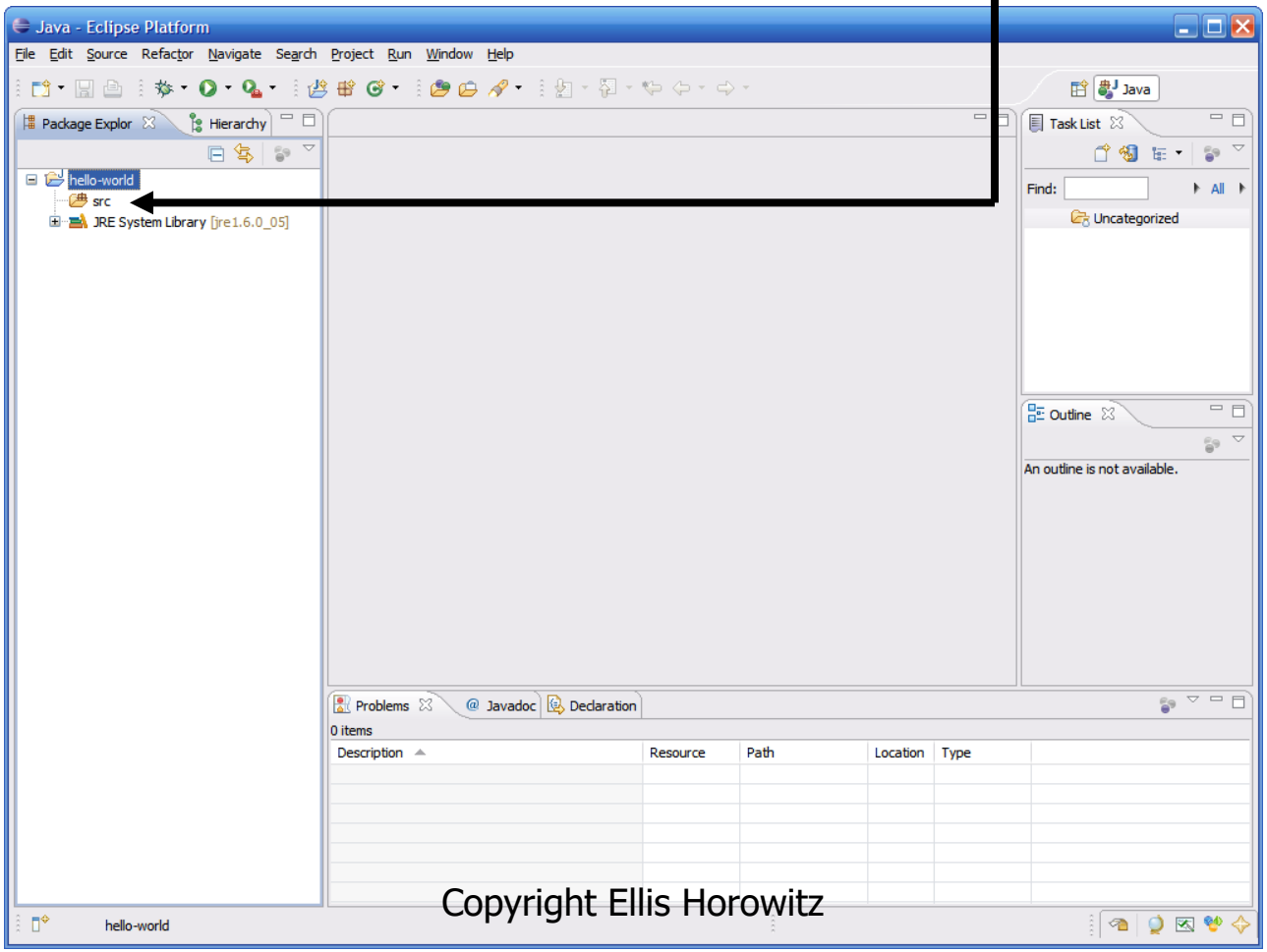
Creating a New Project (continued)

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



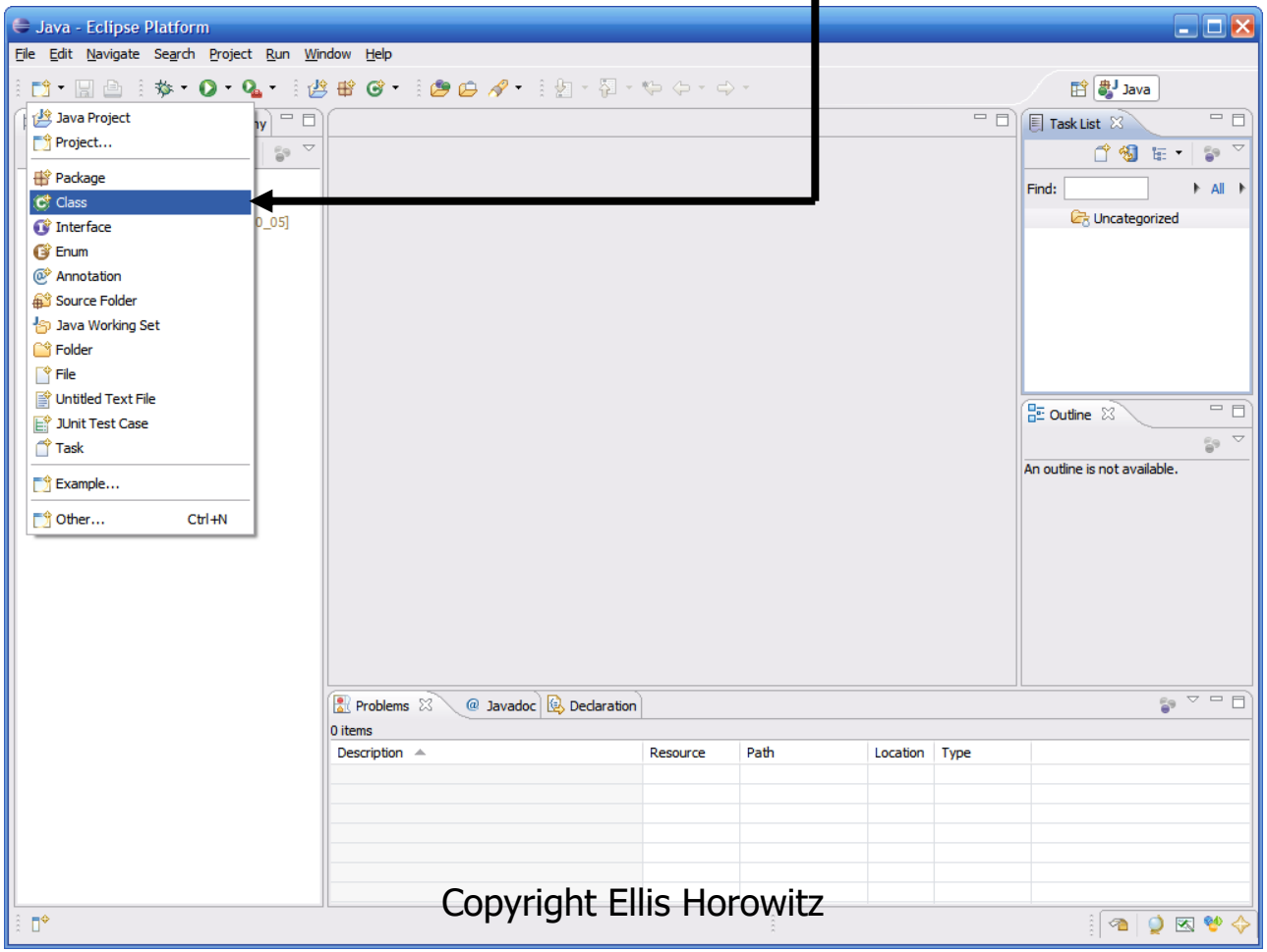
The src folder

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

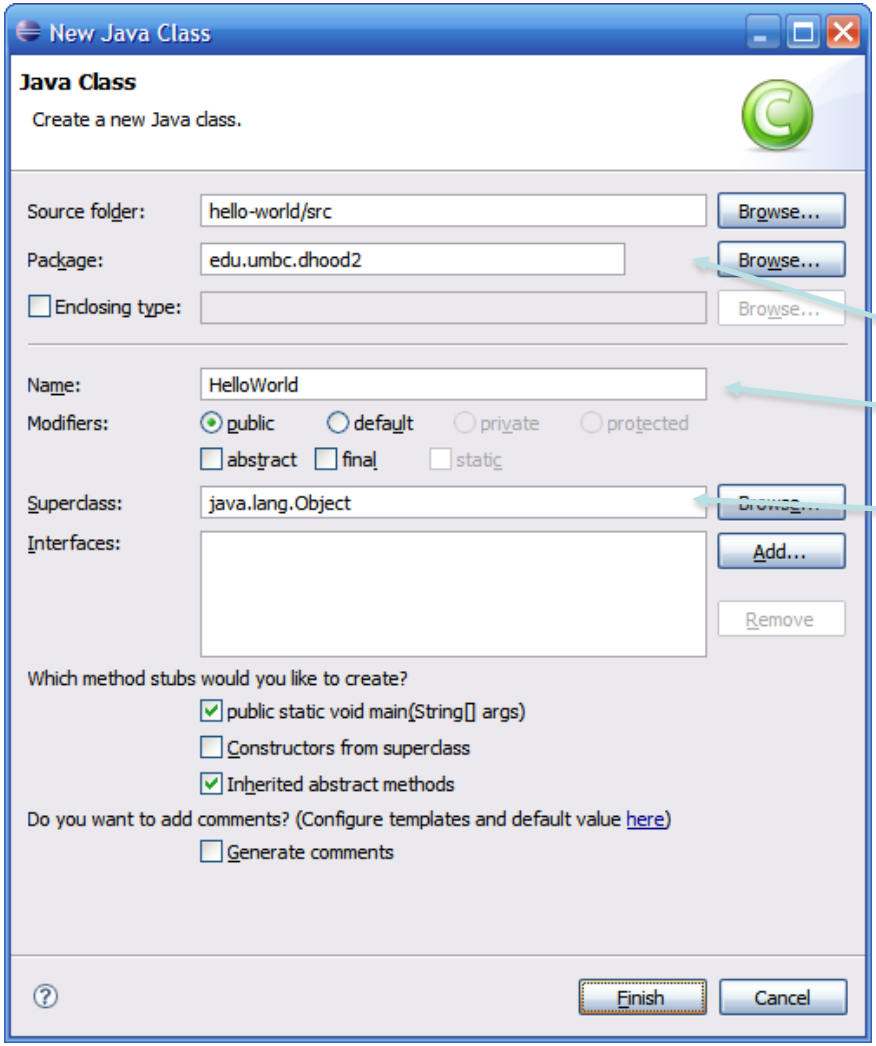


Creating a Class

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



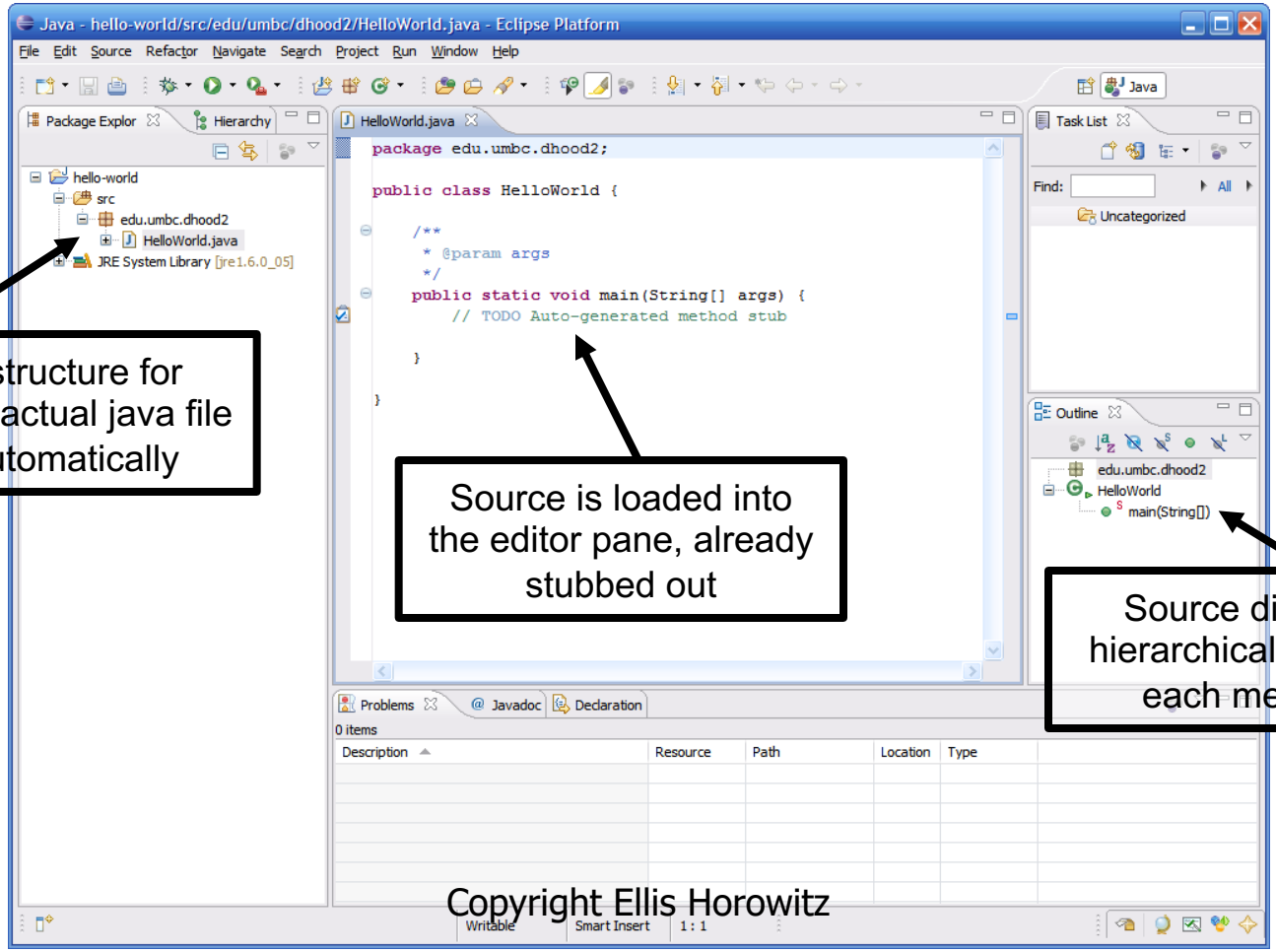
Creating a Class (continued)



- 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…

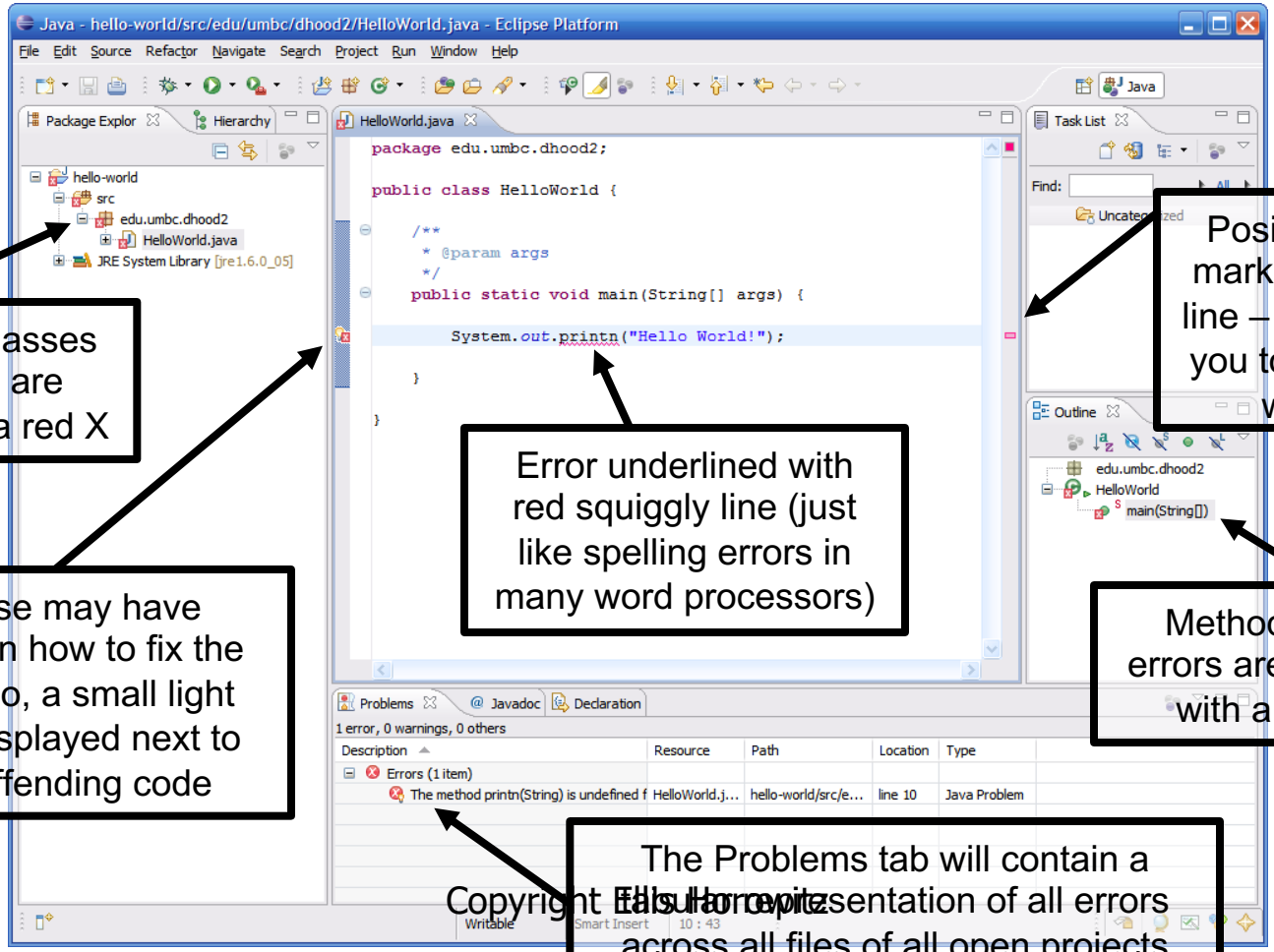


Compiling Source Code

- 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

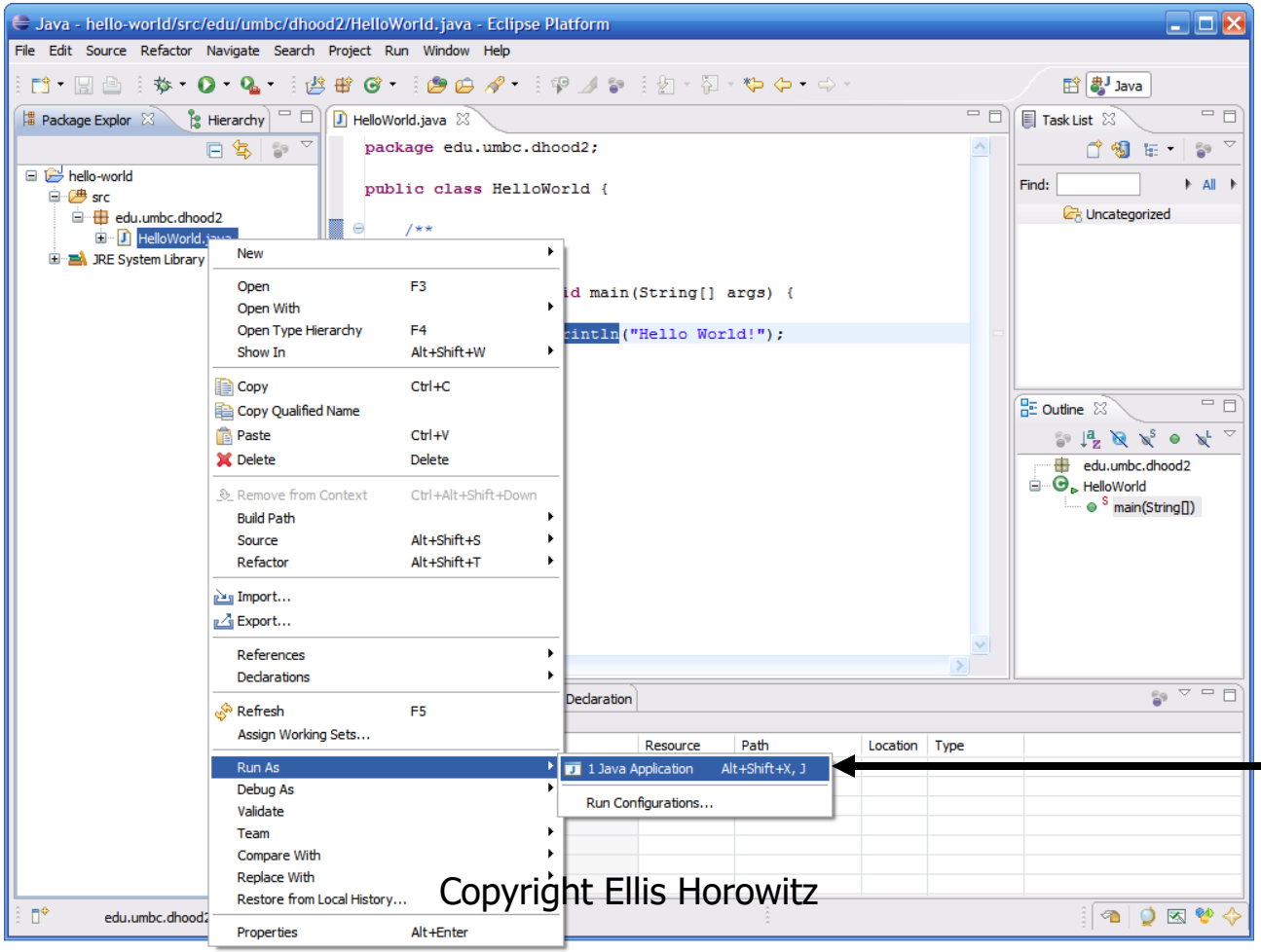
Example Compilation Error

- This code contains a typo in the println statement…



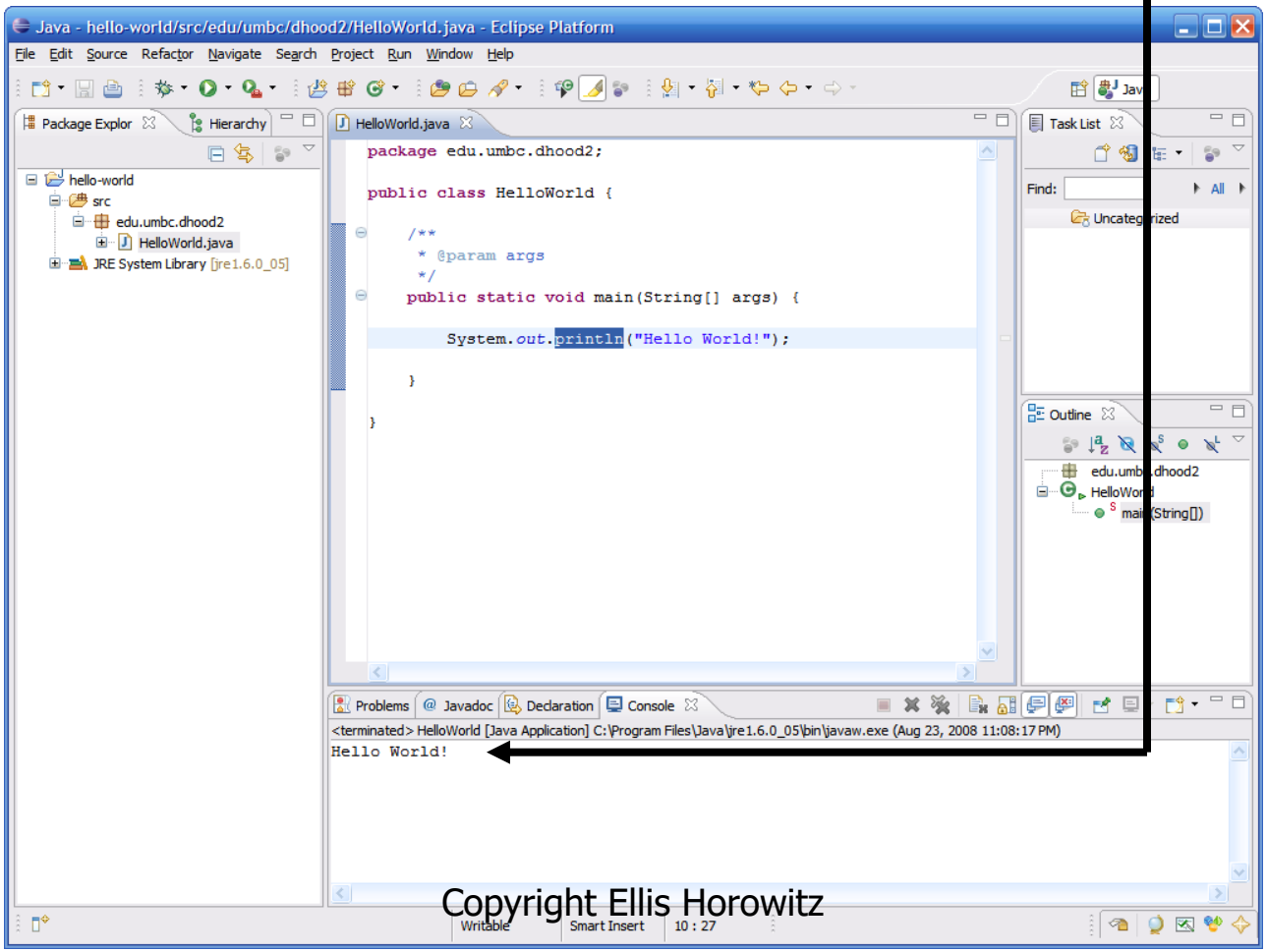
Running Code

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



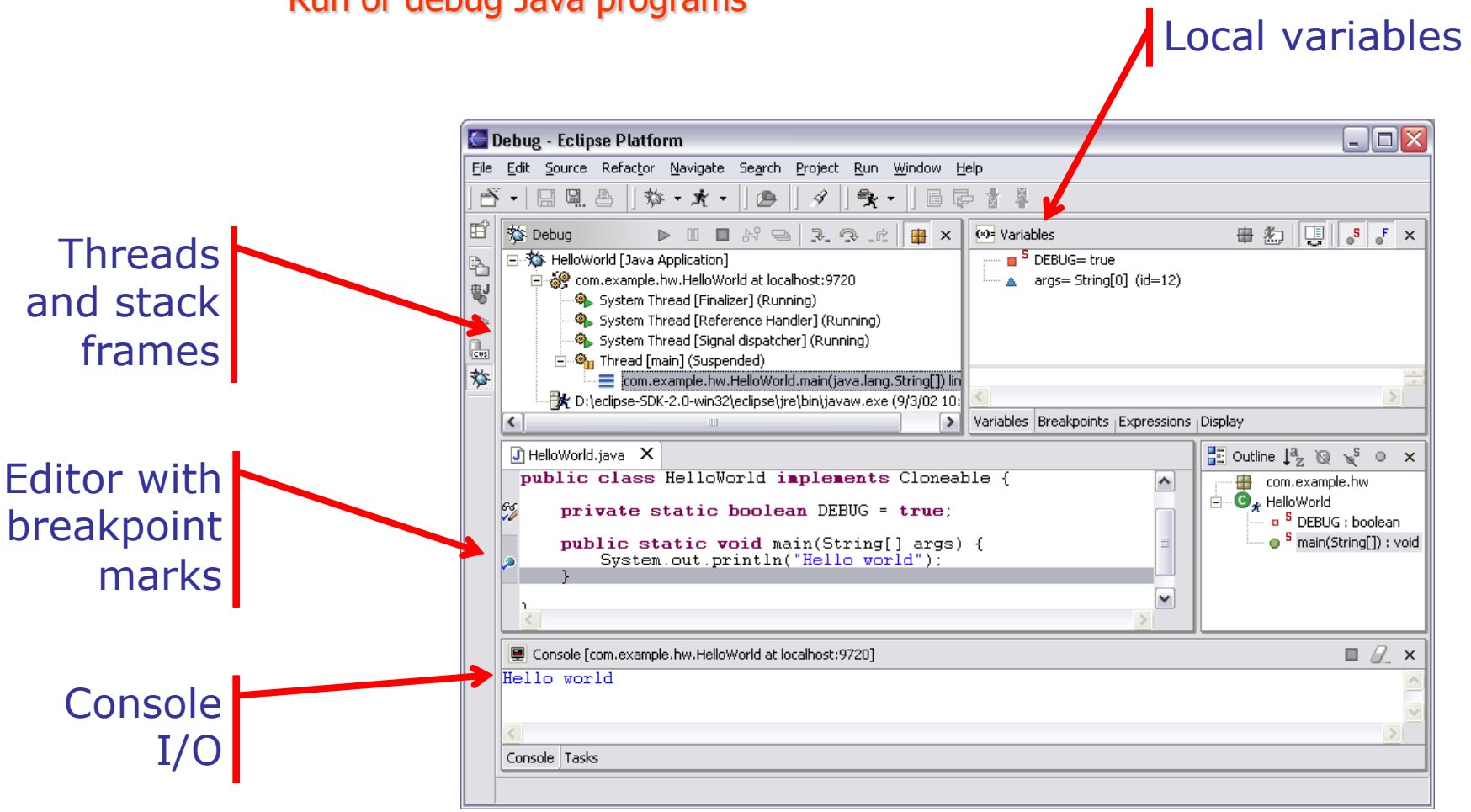
Running Code (continued)

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



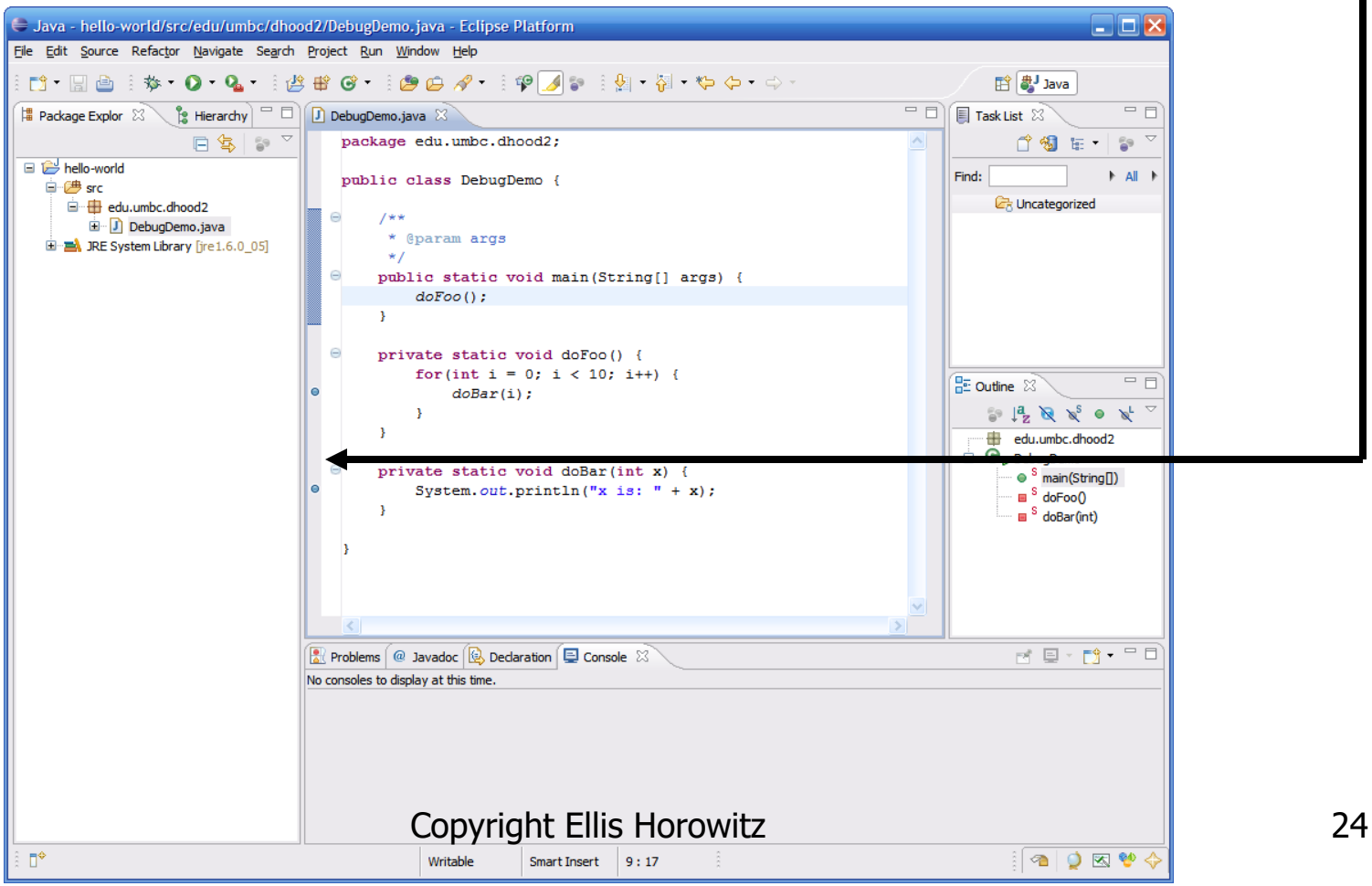
Eclipse Java Debugger

Run or debug Java programs



Debugging Code

- Eclipse comes with a pretty good built-in debugger
- You can set breakpoints 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

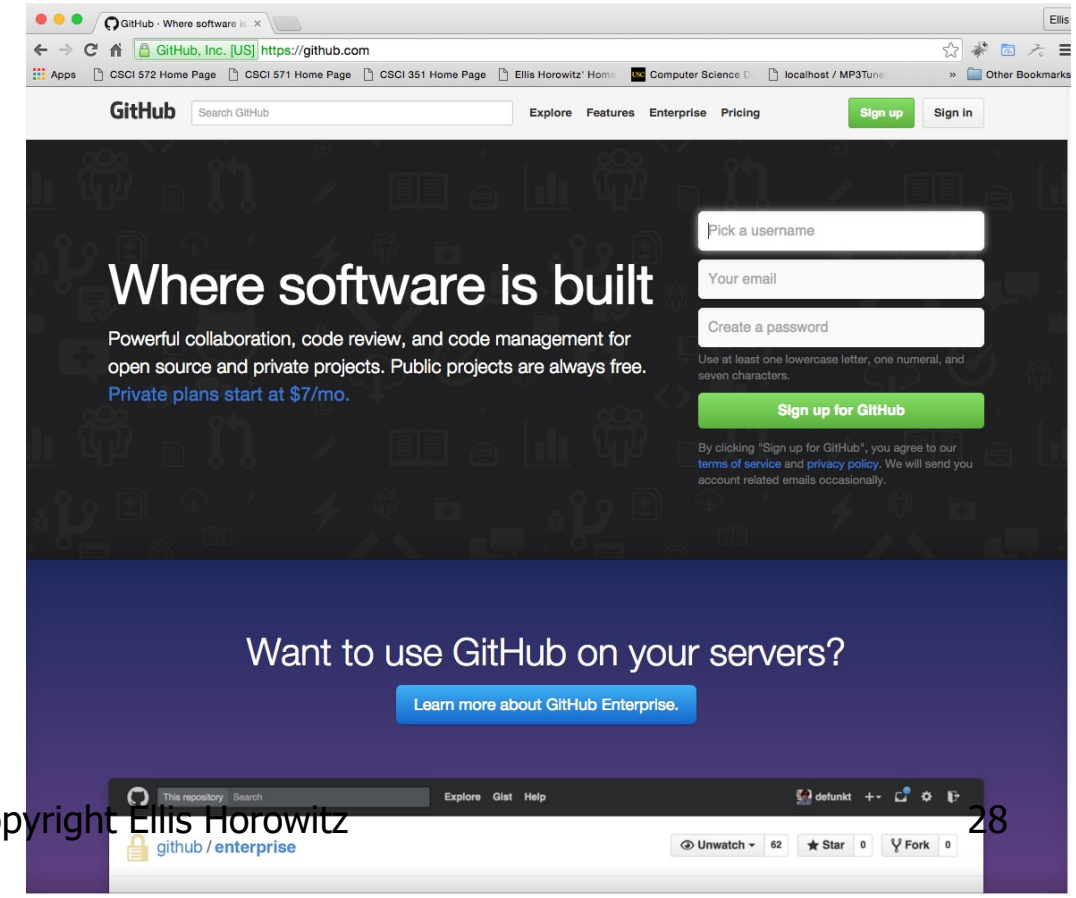
How To Get a Web Page in Java

```
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

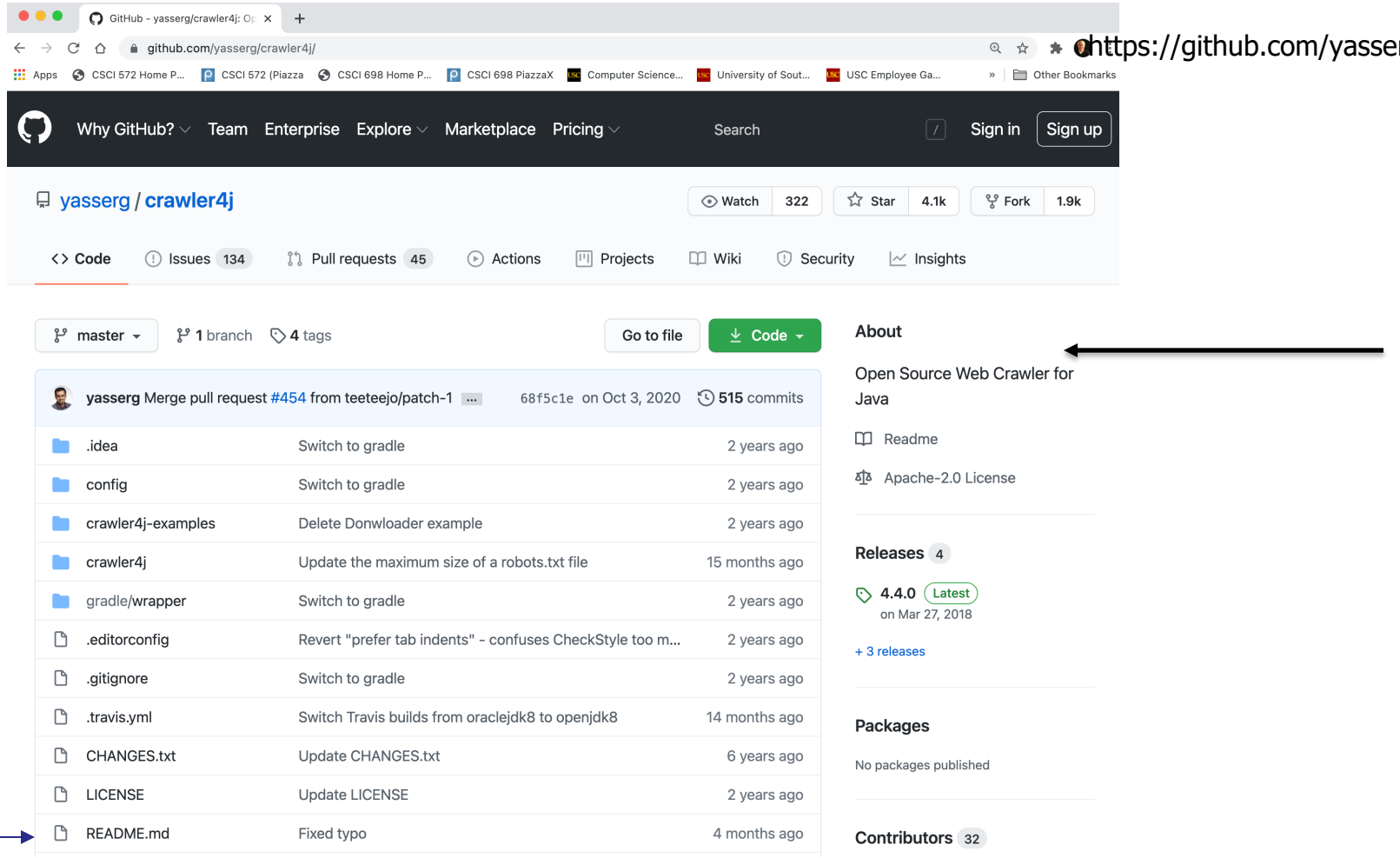
Instructions for Installing Crawler4j

- 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 January 2020, GitHub reports having over 56 million users and over 100 million repositories
  - Microsoft purchased GitHub



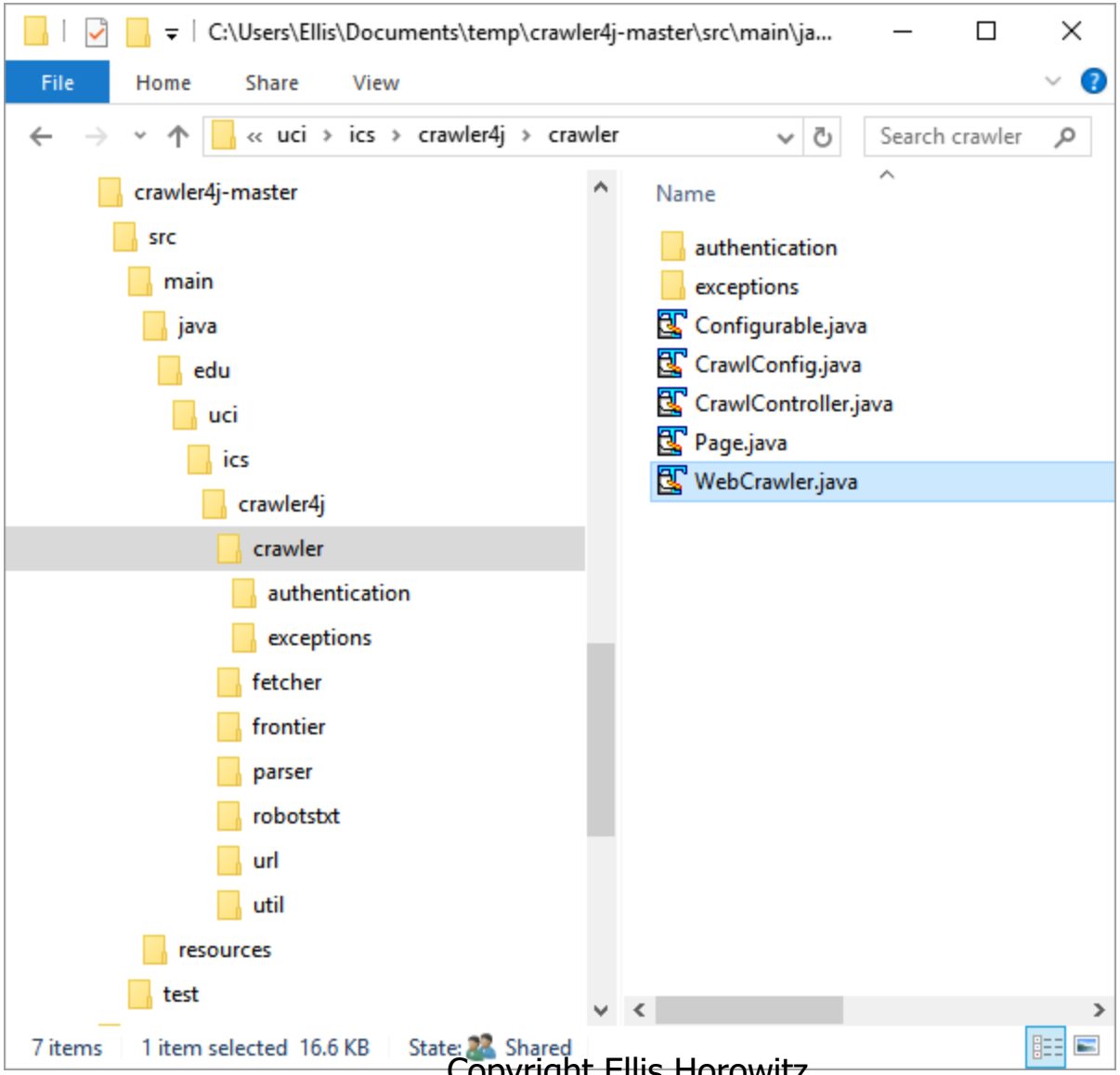
Downloading Crawler4j from GitHub

https://github.com/yasserg/crawler4j/



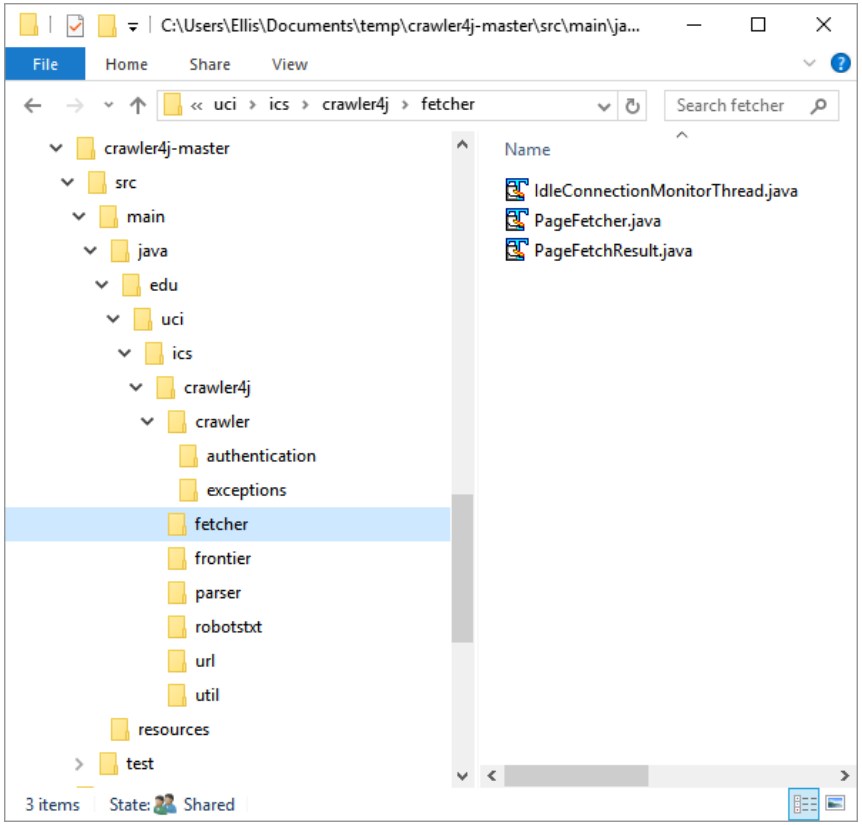
See especially the README file page at
https://github.com/yasserg/crawler4j/blob/master/README.md

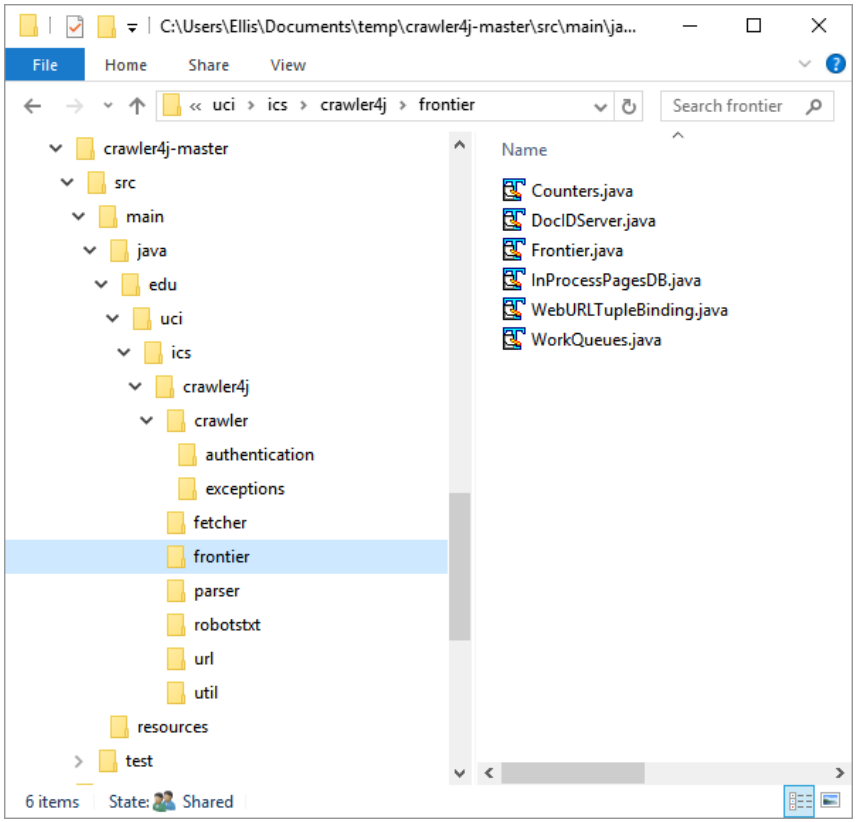
Crawler4j Source Code



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

Crawler4j Source code is Logically Organized into folders

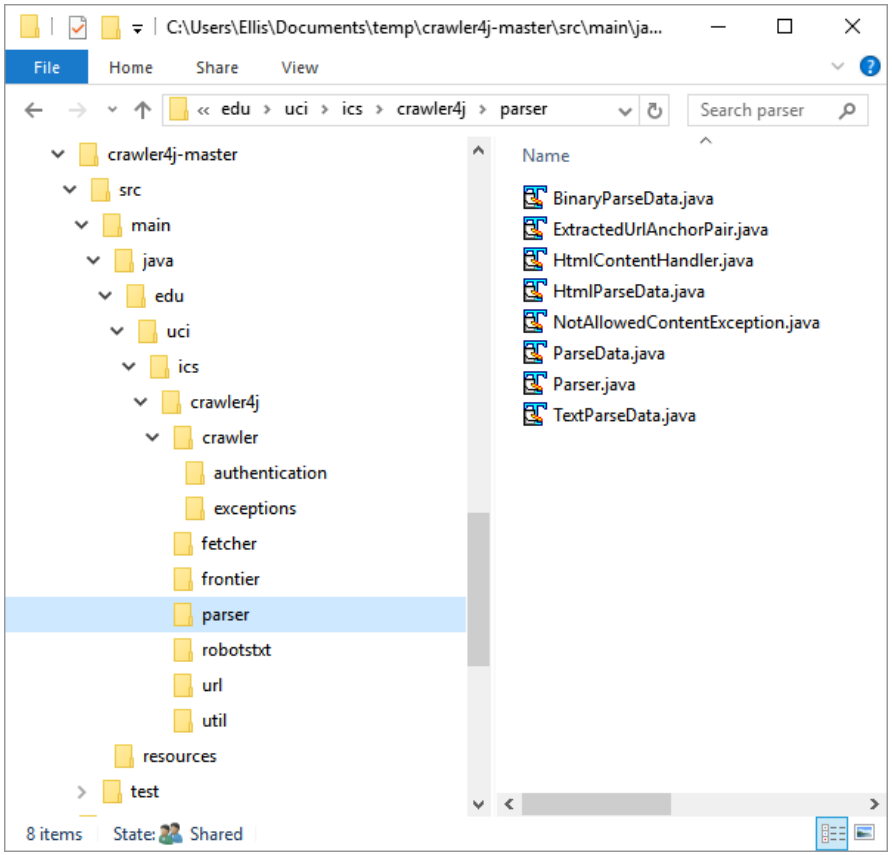


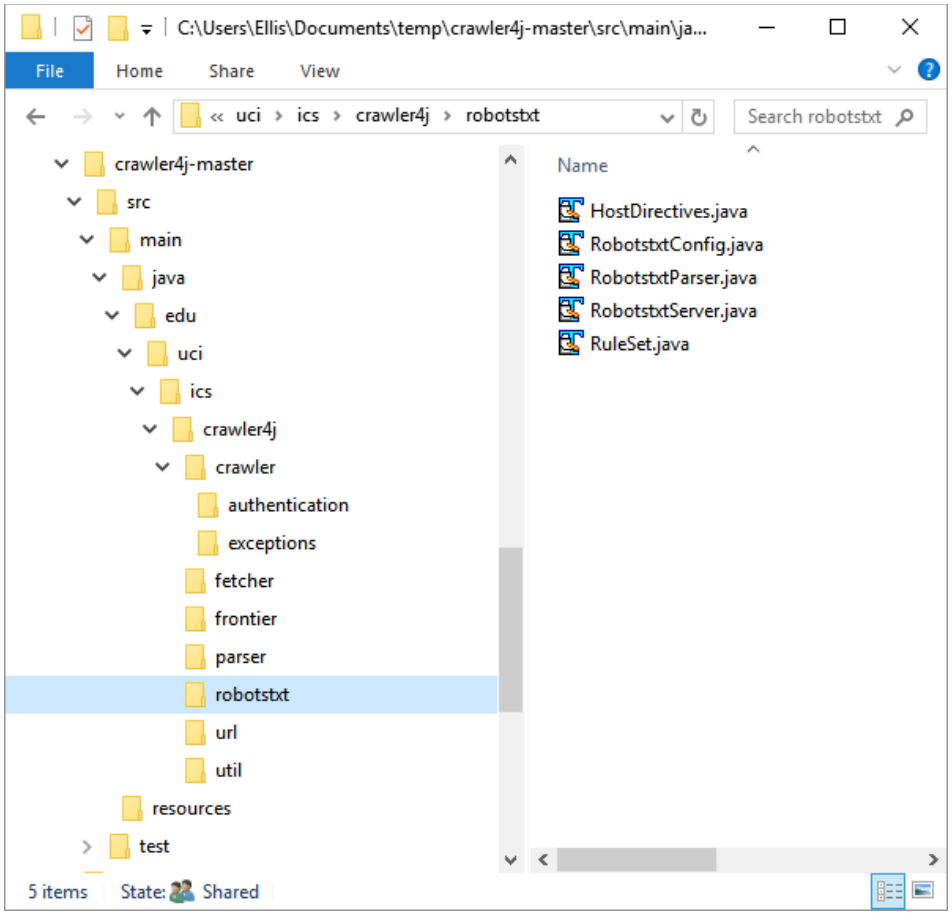


Fetcher Code handles:
- schemes: http, https
- politeness delay;
- redirects;
- max-size settings;
- expired connections

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

Crawler4j Routines are Named According to their Function

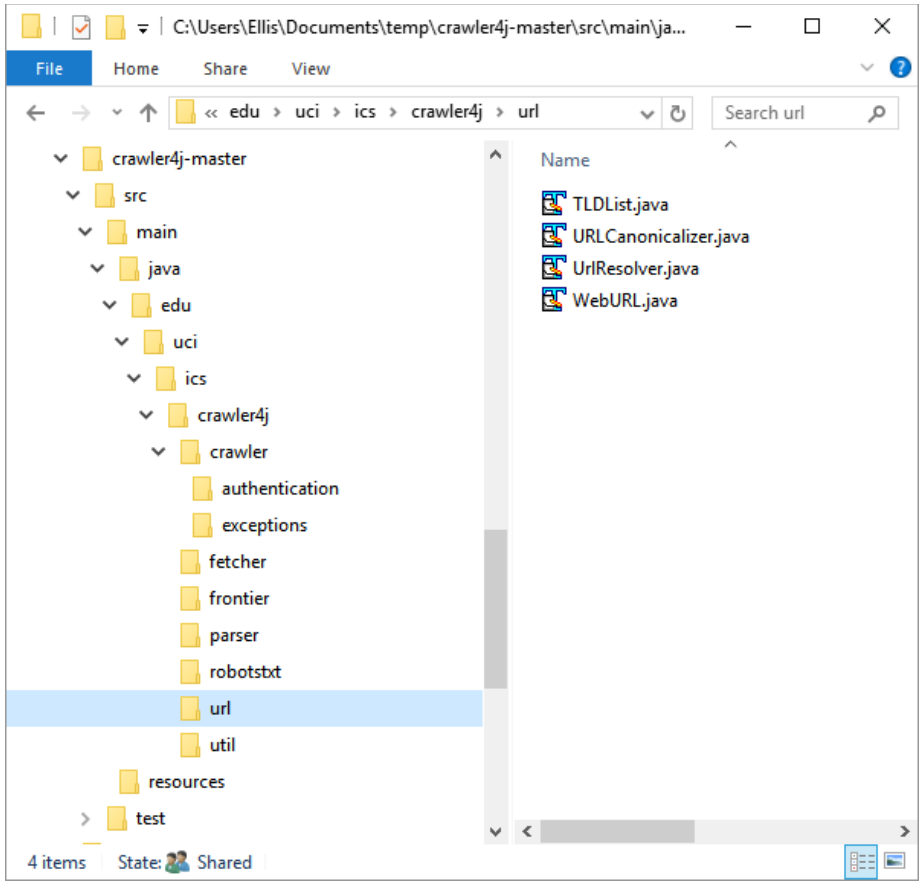


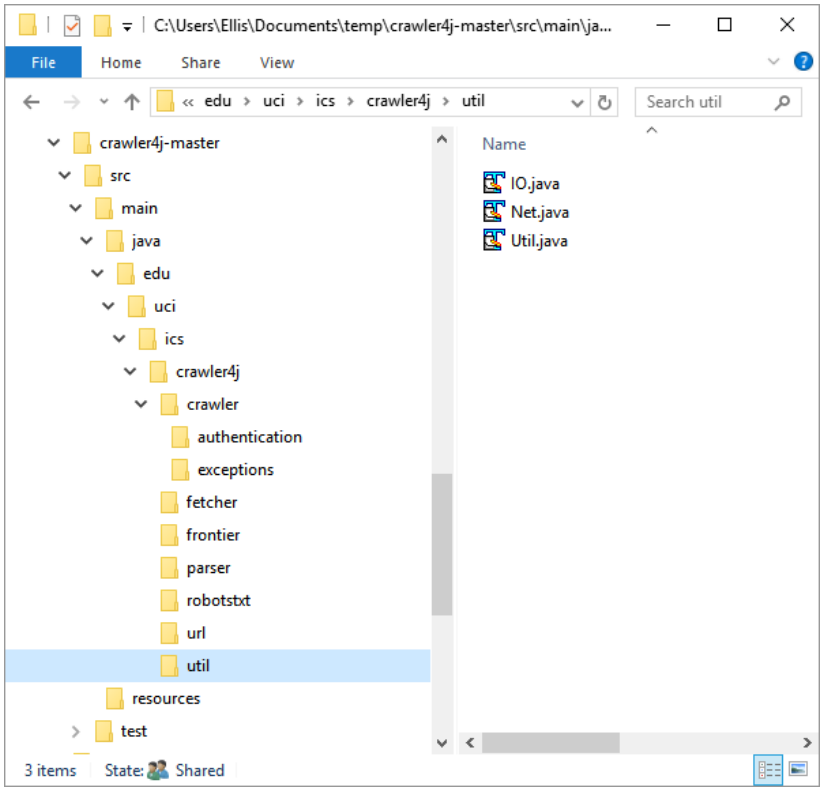


Parser Code handles:
- binary data
- html pages
- extracting links

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

More crawler4j Source code

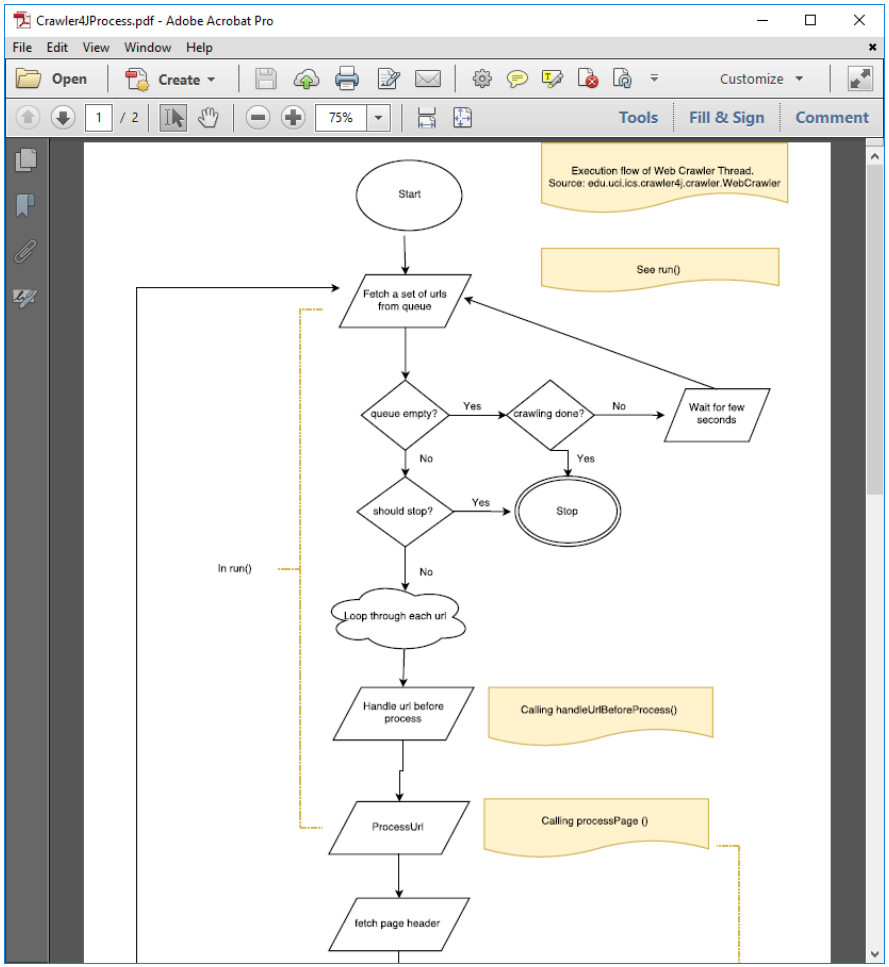


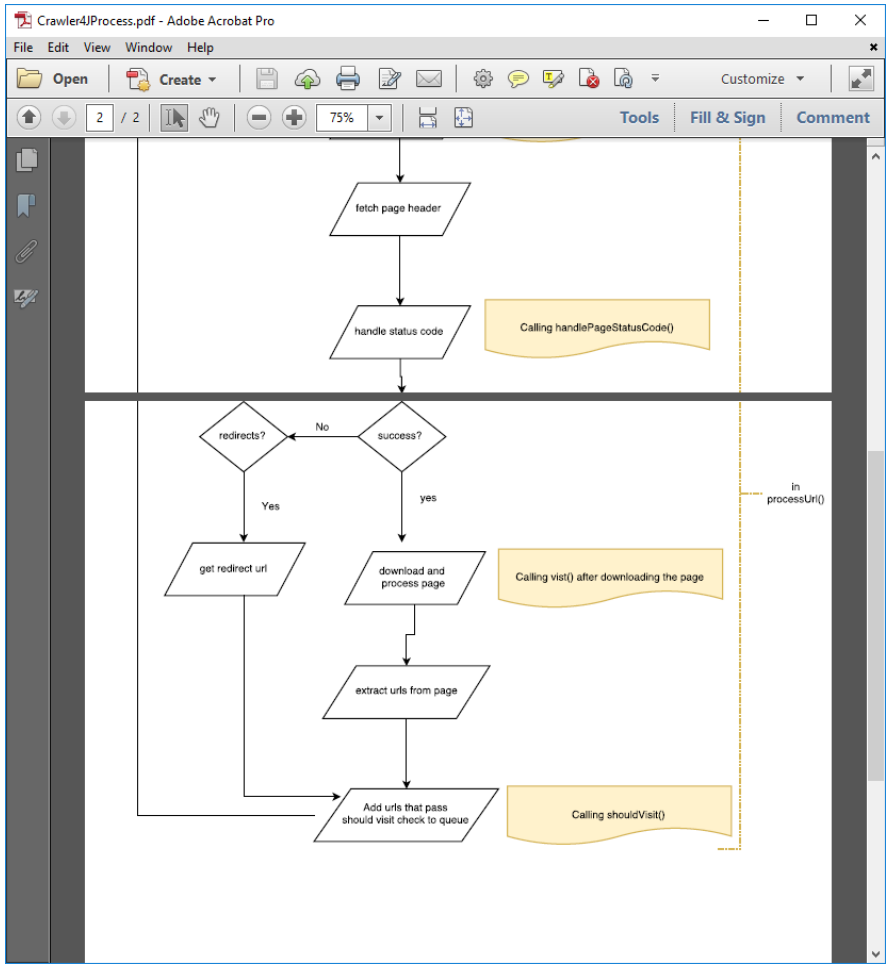


utility routines

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://csci572.com/2022Fall/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("https://www.nytimes.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.nytimes.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 31
- 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.nytimes.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.
-

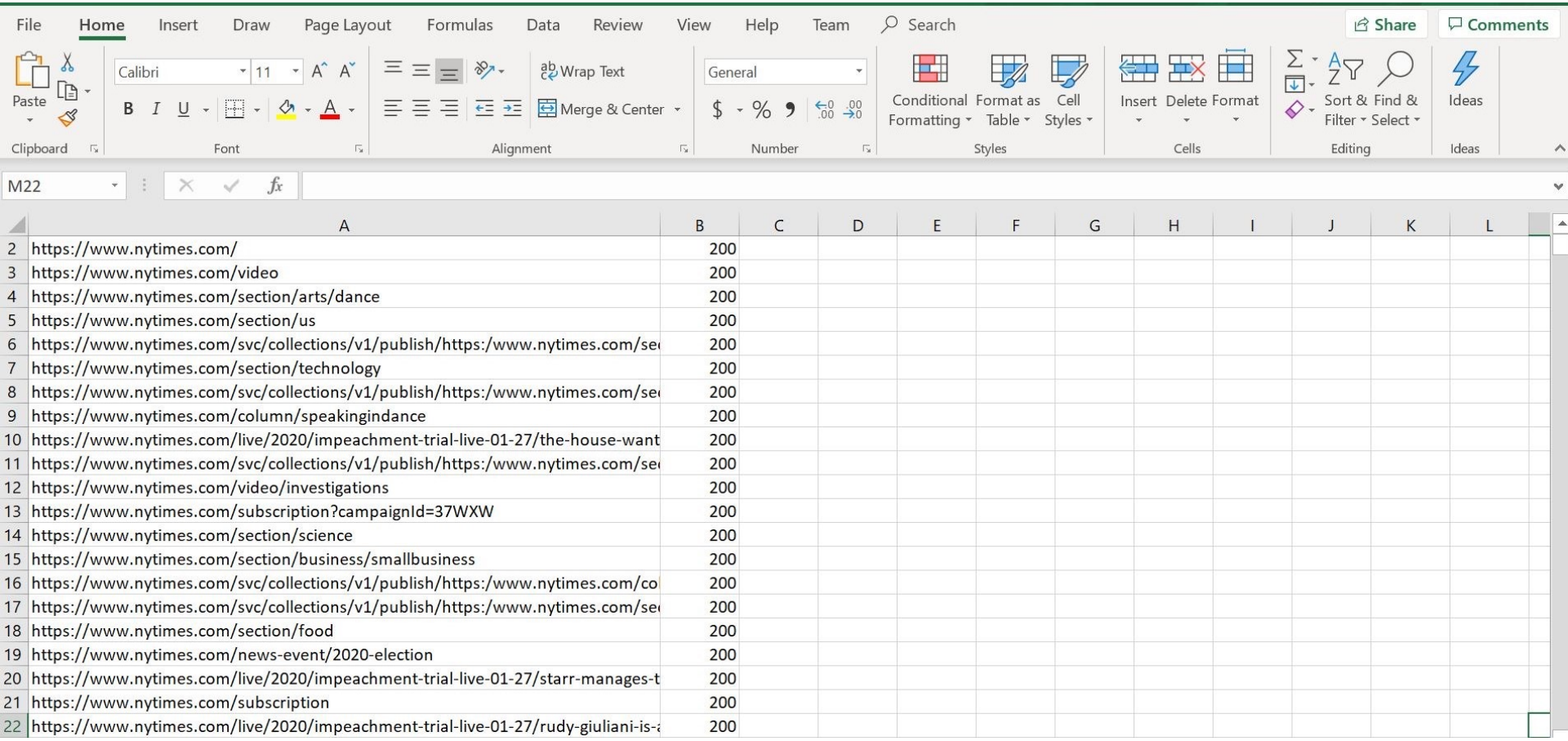
Outgoing URLs

- 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 NY Times Using 20,000 as the Download Limit

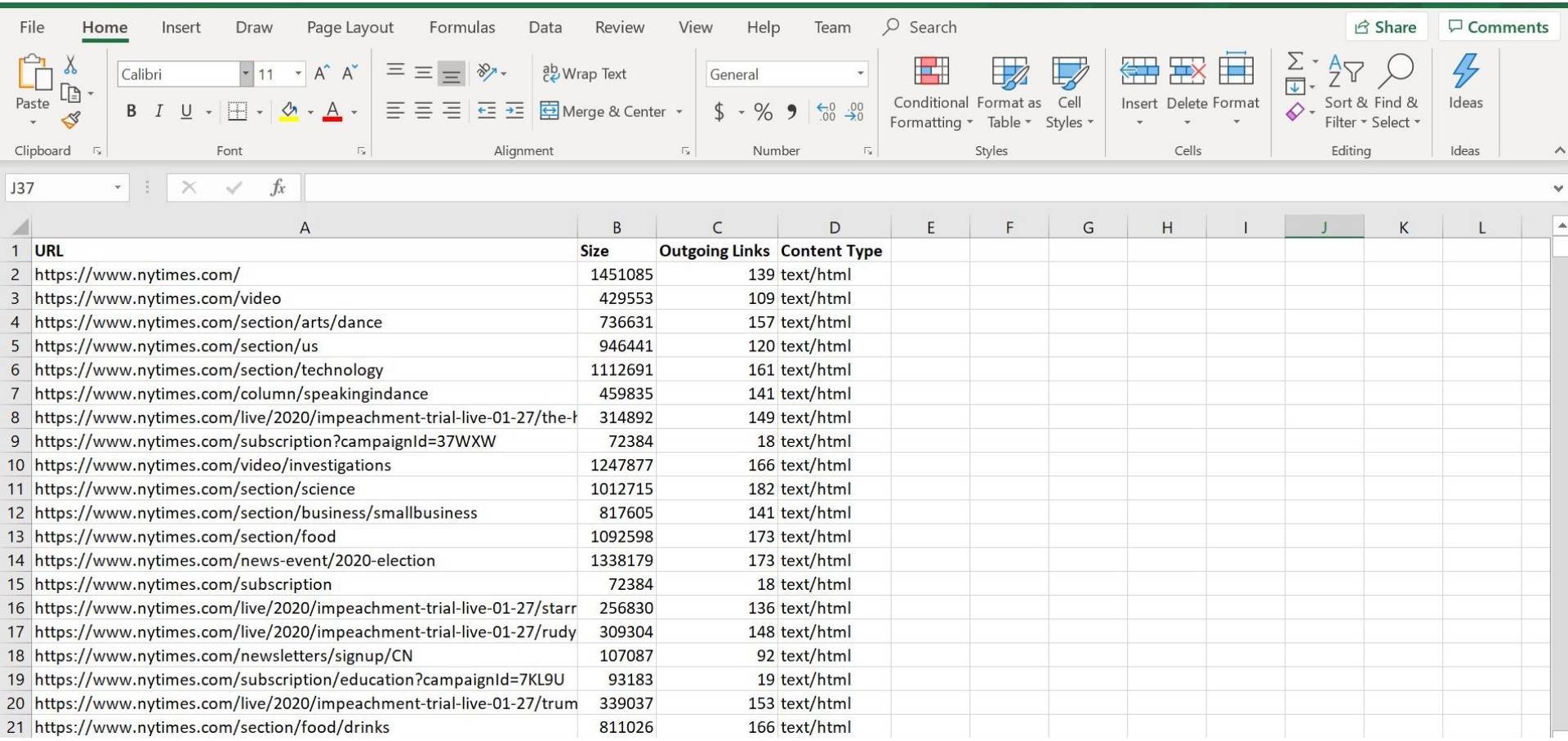
```
News site crawled: https://www.nytimes.com/
Fetch Statistics
==================
# fetches attempted:19750
# fetches succeeded:19067
# fetches aborted or failed:683
Outgoing URLs
==================
Total URLs extracted:964461
# unique URLs extracted:73450
# unique URLs within nytimes:31115
# unique URLs outside nytimes:42335
Status Codes
==================
200 OK:19067
301 Moved Permanently:140
302 Moved Temporarily:21
503 Service Unavailable:366
401 Unauthorized Error:1
403 Forbidden:8
404 Not Found:143
500 Internal Server Error:1
400 Bad Request:3
File Sizes
==================
<1KB:14
1KB~<10KB:1052
10KB~<100KB:694
100KB~<1MB:8602
>=1MB:1
Content Type
==================
text/html:18349
image/jpeg=5
image/png=3
```

Sample Fetch File for NY Times



	A	B
2	https://www.nytimes.com/	200
3	https://www.nytimes.com/video	200
4	https://www.nytimes.com/section/arts/dance	200
5	https://www.nytimes.com/section/us	200
6	https://www.nytimes.com/svc/collections/v1/publish/https:/www.nytimes.com/se	200
7	https://www.nytimes.com/section/technology	200
8	https://www.nytimes.com/svc/collections/v1/publish/https:/www.nytimes.com/se	200
9	https://www.nytimes.com/column/speakingindance	200
10	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/the-house-want	200
11	https://www.nytimes.com/svc/collections/v1/publish/https:/www.nytimes.com/se	200
12	https://www.nytimes.com/video/investigations	200
13	https://www.nytimes.com/subscription?campaignId=37WXW	200
14	https://www.nytimes.com/section/science	200
15	https://www.nytimes.com/section/business/smallbusiness	200
16	https://www.nytimes.com/svc/collections/v1/publish/https:/www.nytimes.com/co	200
17	https://www.nytimes.com/svc/collections/v1/publish/https:/www.nytimes.com/se	200
18	https://www.nytimes.com/section/food	200
19	https://www.nytimes.com/news-event/2020-election	200
20	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/starr-manages-t	200
21	https://www.nytimes.com/subscription	200
22	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/rudy-giuliani-is-	200

Sample Visit File for NY Times



	A	B	C	D
1	**URL**	**Size**	**Outgoing Links**	**Content Type**
2	https://www.nytimes.com/	1451085	139	text/html
3	https://www.nytimes.com/video	429553	109	text/html
4	https://www.nytimes.com/section/arts/dance	736631	157	text/html
5	https://www.nytimes.com/section/us	946441	120	text/html
6	https://www.nytimes.com/section/technology	1112691	161	text/html
7	https://www.nytimes.com/column/speakingindance	459835	141	text/html
8	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/the-h	314892	149	text/html
9	https://www.nytimes.com/subscription?campaignId=37WXW	72384	18	text/html
10	https://www.nytimes.com/video/investigations	1247877	166	text/html
11	https://www.nytimes.com/section/science	1012715	182	text/html
12	https://www.nytimes.com/section/business/smallbusiness	817605	141	text/html
13	https://www.nytimes.com/section/food	1092598	173	text/html
14	https://www.nytimes.com/news-event/2020-election	1338179	173	text/html
15	https://www.nytimes.com/subscription	72384	18	text/html
16	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/starr	256830	136	text/html
17	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/rudy	309304	148	text/html
18	https://www.nytimes.com/newsletters/signup/CN	107087	92	text/html
19	https://www.nytimes.com/subscription/education?campaignId=7KL9U	93183	19	text/html
20	https://www.nytimes.com/live/2020/impeachment-trial-live-01-27/trum	339037	153	text/html
21	https://www.nytimes.com/section/food/drinks	811026	166	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_nytimes.txt,
2. fetch_nytimes.csv
3. visit_nytimes.csv

- Place your crawl.zip file in your csci572/hw2 folder