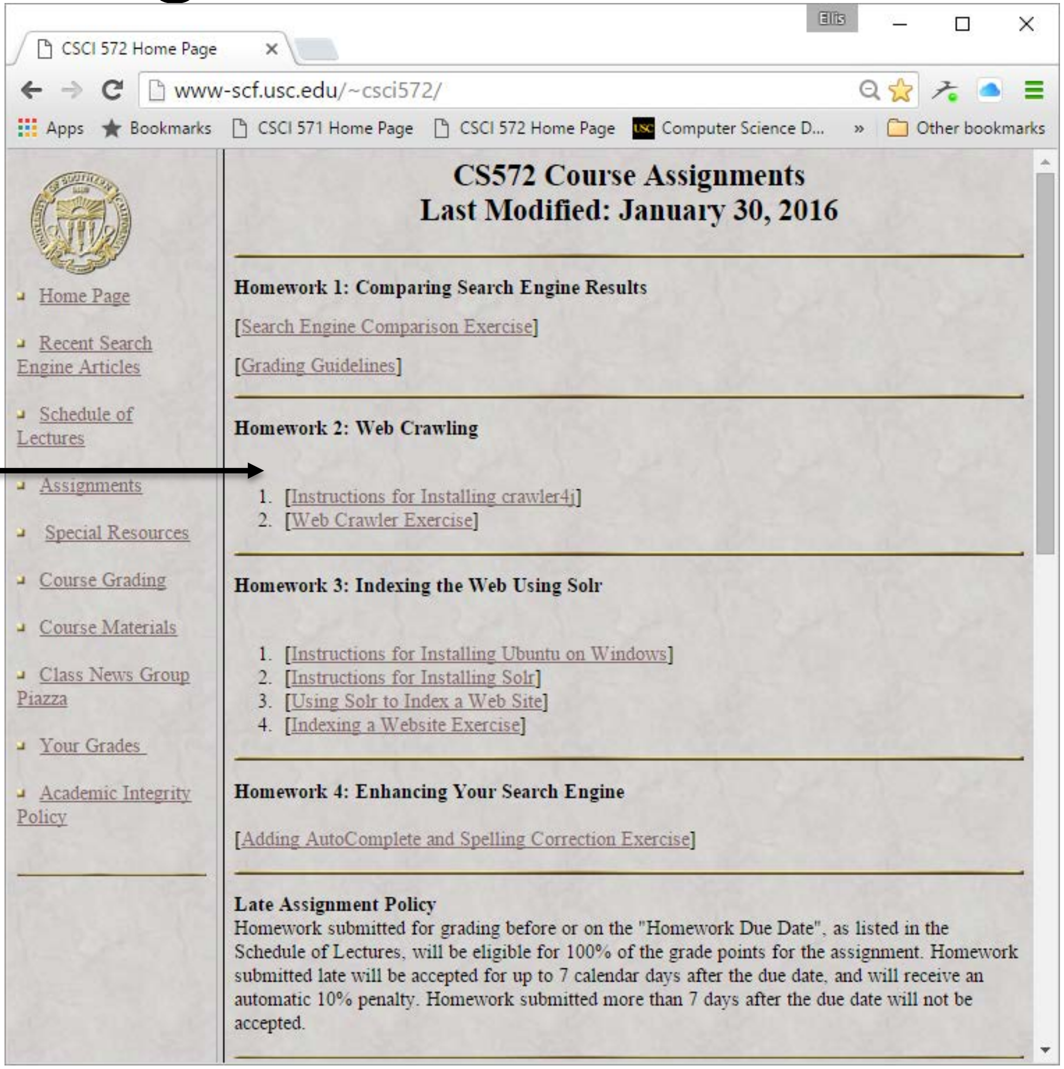
HW2 Assignment

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



CS572 Course Assignments
Last Modified: January 30, 2016

Homework 1: Comparing Search Engine Results

[Search Engine Comparison Exercise]

[Grading Guidelines]

Homework 2: Web Crawling

1. [Instructions for Installing crawler4j]
2. [Web Crawler Exercise]

Homework 3: Indexing the Web Using Solr

1. [Instructions for Installing Ubuntu on Windows]
2. [Instructions for Installing Solr]
3. [Using Solr to Index a Web Site]
4. [Indexing a Website Exercise]

Homework 4: Enhancing Your Search Engine

[Adding AutoComplete and Spelling Correction Exercise]

Late Assignment Policy
Homework submitted for grading before or on the "Homework Due Date", as listed in the Schedule of Lectures, will be eligible for 100% of the grade points for the assignment. Homework submitted late will be accepted for up to 7 calendar days after the due date, and will receive an automatic 10% penalty. Homework submitted more than 7 days after the due date will not be accepted.

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

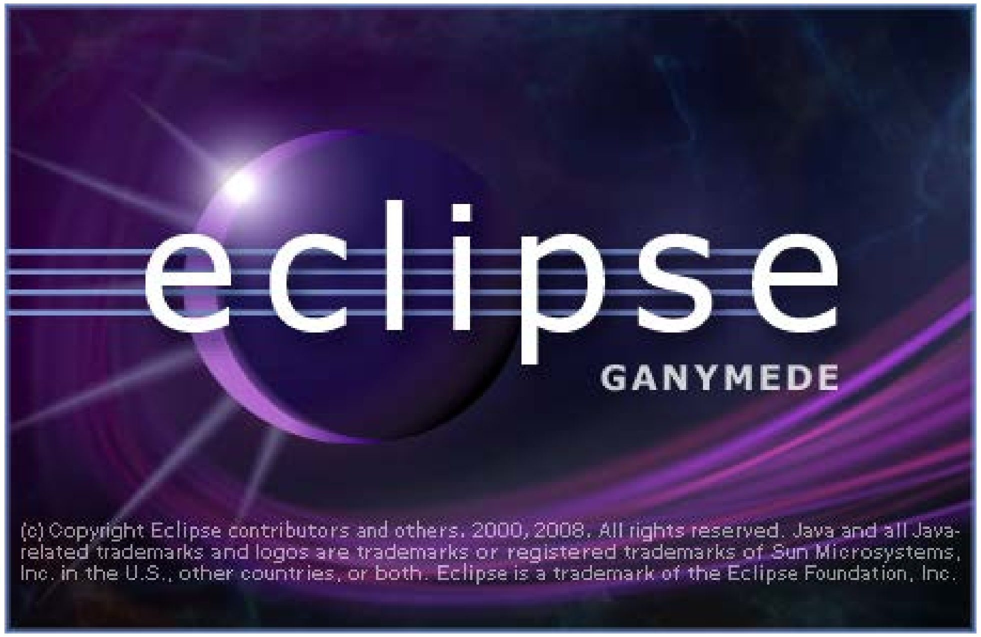
- Eclipse can be downloaded from…
  - http://www.eclipse.org/downloads/packages/
- Eclipse comes bundled as a zip file (Windows) or a tarball (all other operating systems)

Installing Eclipse

- Simply unwrap the zip file to some directory where you want to store the executables
- On windows
  - I typically unwrap the zip file to C:\eclipse\
  - I then typically create a shortcut on my desktop to the eclipse executable
    - C:\eclipse\eclipse.exe
- Under Linux
  - I typically unwrap to /eclipse/

Launching Eclipse

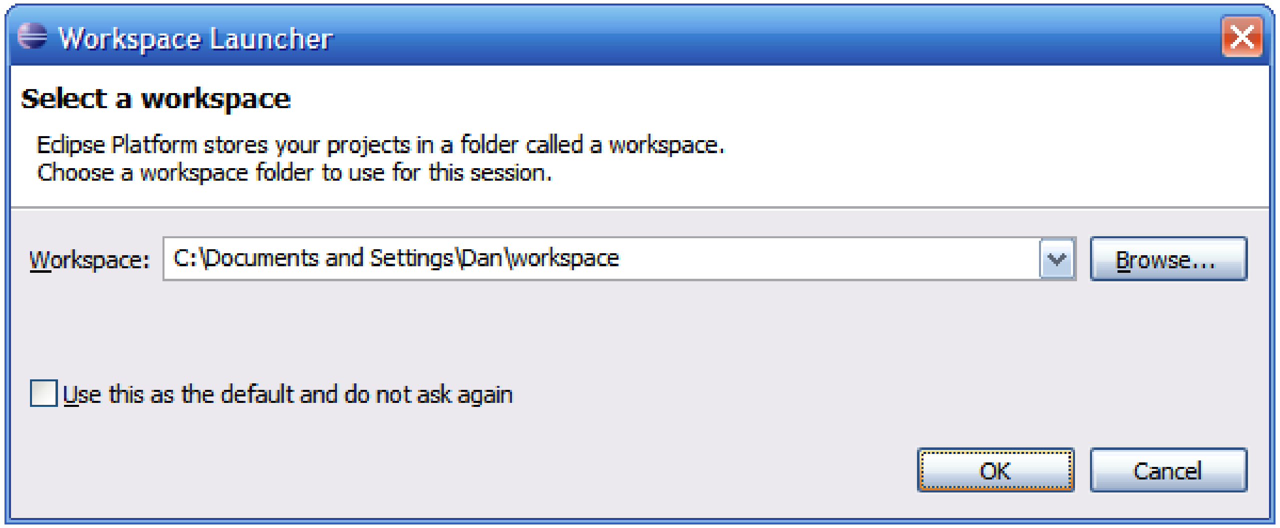
- Once you have the environment setup, go ahead and launch eclipse
- You should see the following splash screen…



(c) Copyright Eclipse contributors and others. 2000, 2008. All rights reserved. Java and all Java-related trademarks and logos are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S., other countries, or both. Eclipse is a trademark of the Eclipse Foundation, Inc.

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 Launcher

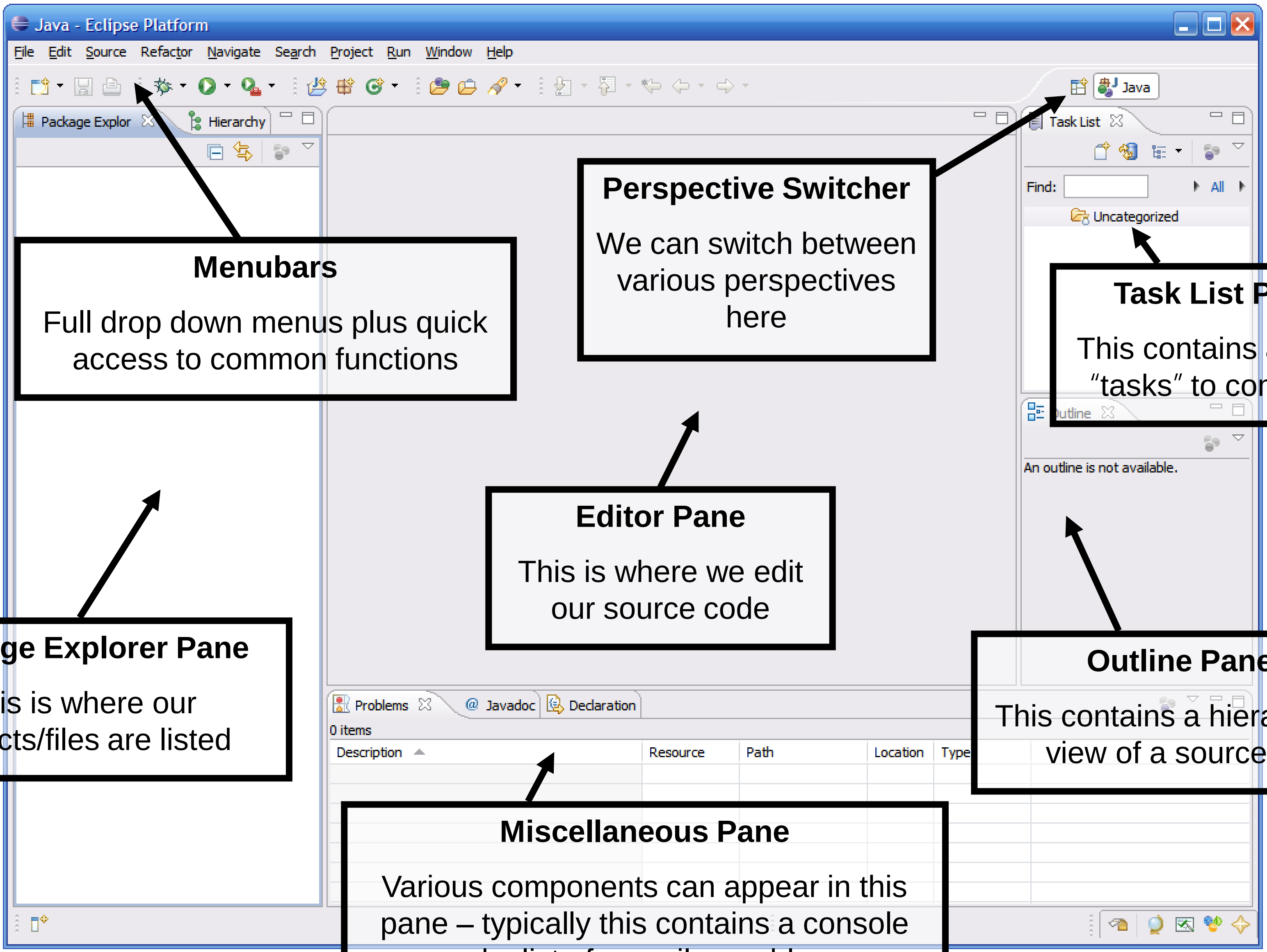
Select a workspace

Eclipse Platform stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

Workspace: C:\Documents and Settings\Dan\workspace

☐ Use this as the default and do not ask again

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

Editor Pane

This is where we edit our source code

Package Explorer Pane

This is where our projects/files are listed

Outline Pane

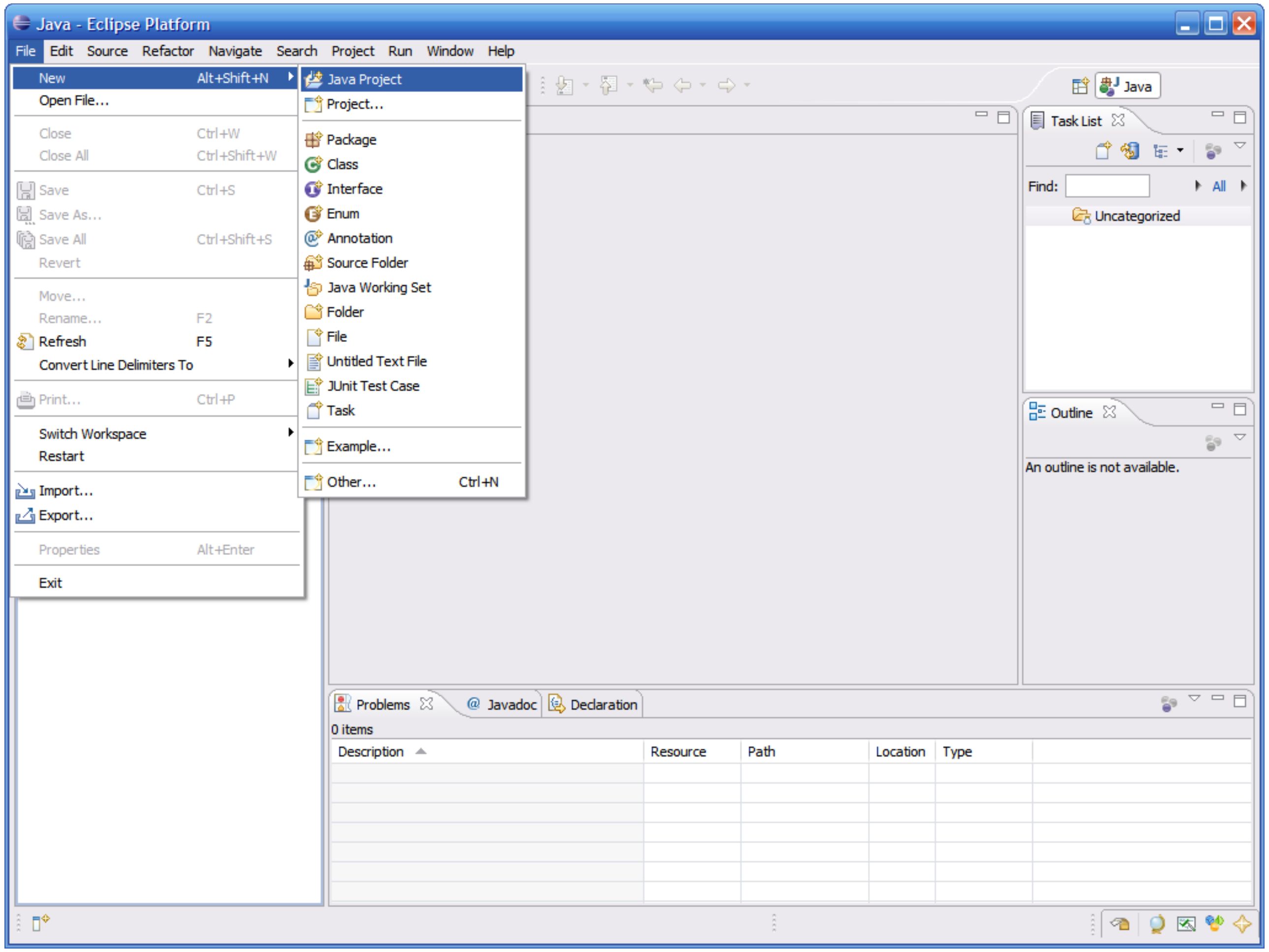
This contains a hierarchical view of a source file

Miscellaneous Pane

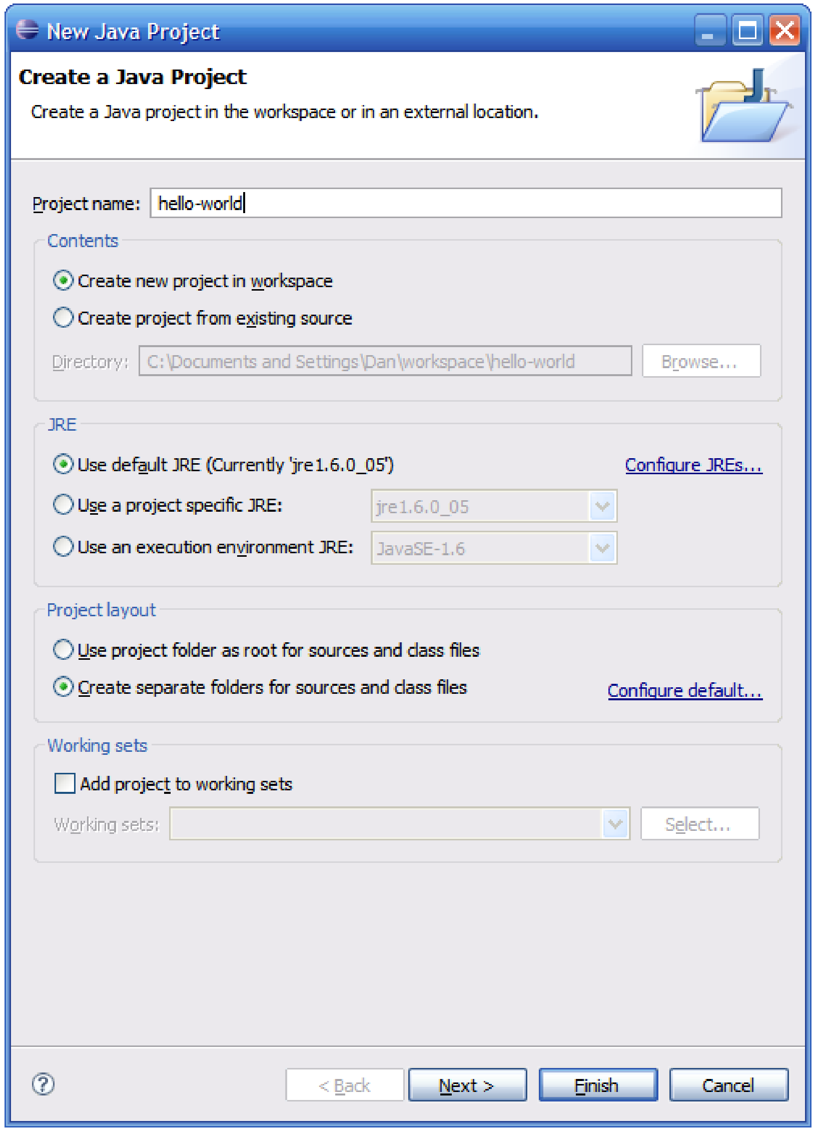
Various components can appear in this pane – typically this contains a console and a list of compiler problems

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)



New Java Project

Create a Java Project

Create a Java project in the workspace or in an external location.

Project name: hello-world

Contents

- [● Create new project in workspace]
- ○ Create project from existing source

Directory: C:\Documents and Settings\Dan\workspace\hello-world — Browse...

JRE

- [● Use default JRE (Currently 'jre1.6.0_05')] — Configure JREs...
- ○ Use a project specific JRE: jre1.6.0_05
- ○ Use an execution environment JRE: JavaSE-1.6

Project layout

- ○ Use project folder as root for sources and class files
- [● Create separate folders for sources and class files] — Configure default...

Working sets

- ☐ Add project to working sets

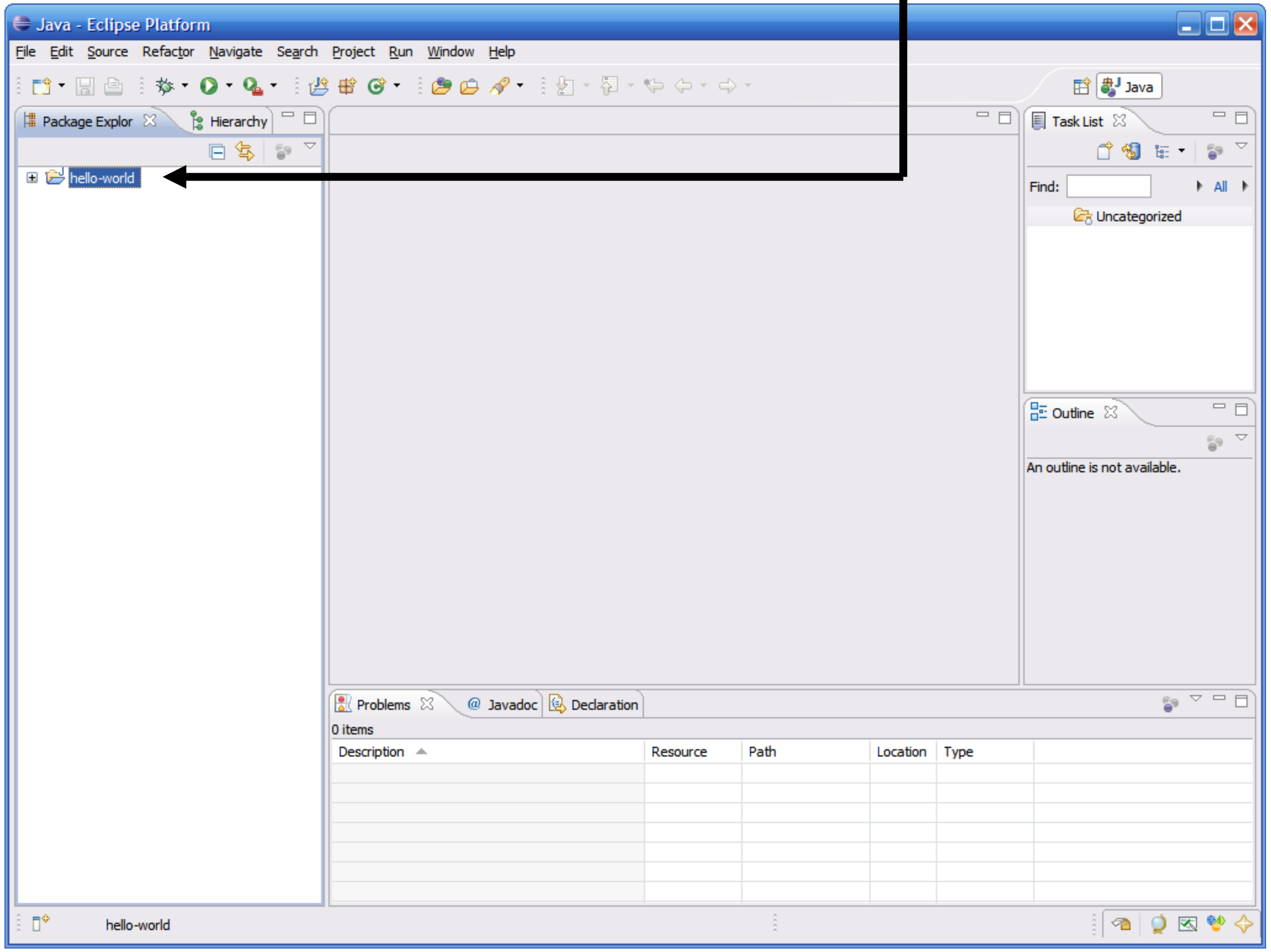
Working sets: — Select...

< Back | Next > | Finish | Cancel

- Enter a name for the project, then click Finish

Creating a New Project (continued)

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



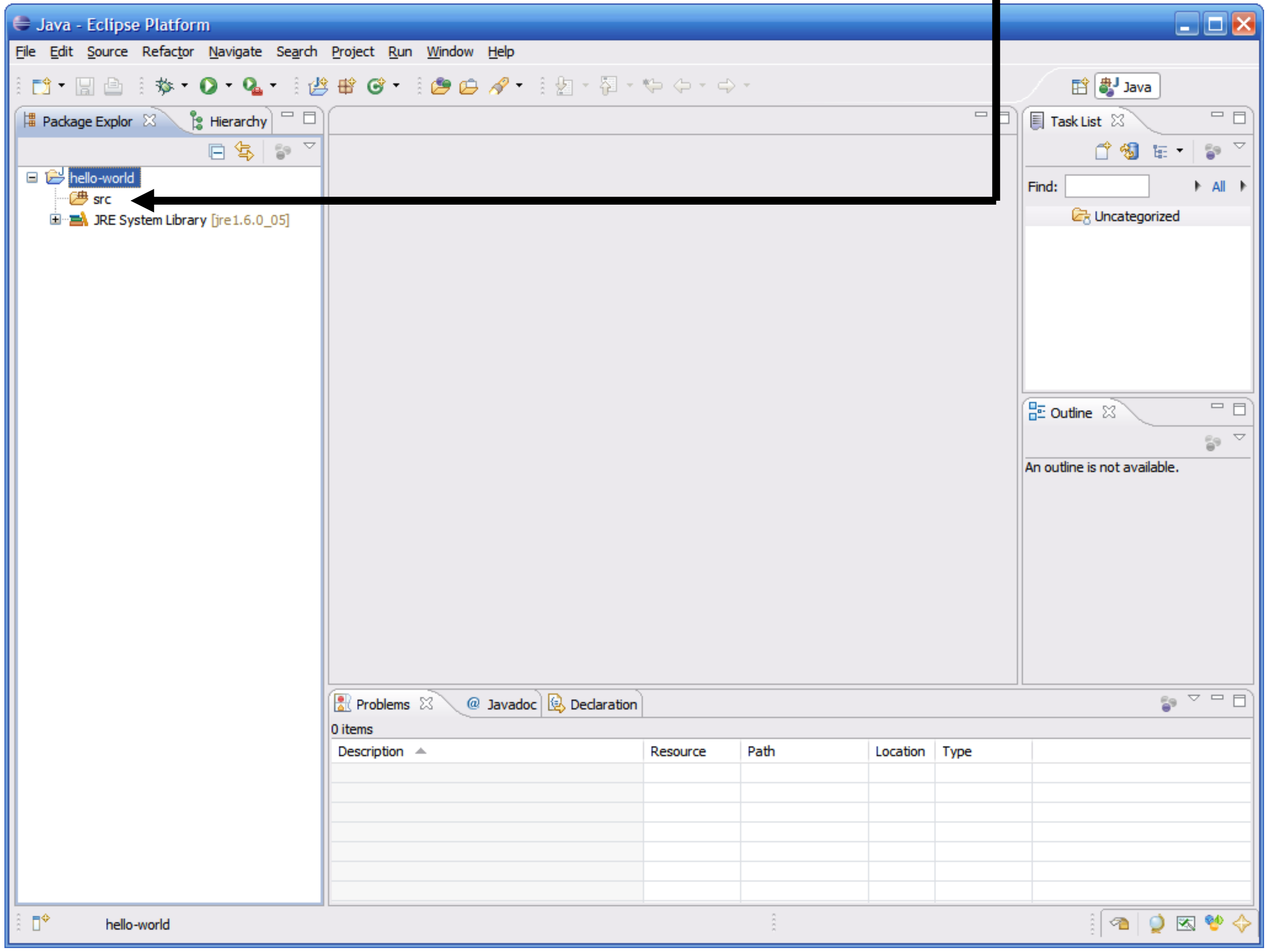
An outline is not available.

0 items

Description	Resource	Path	Location	Type

The src folder

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

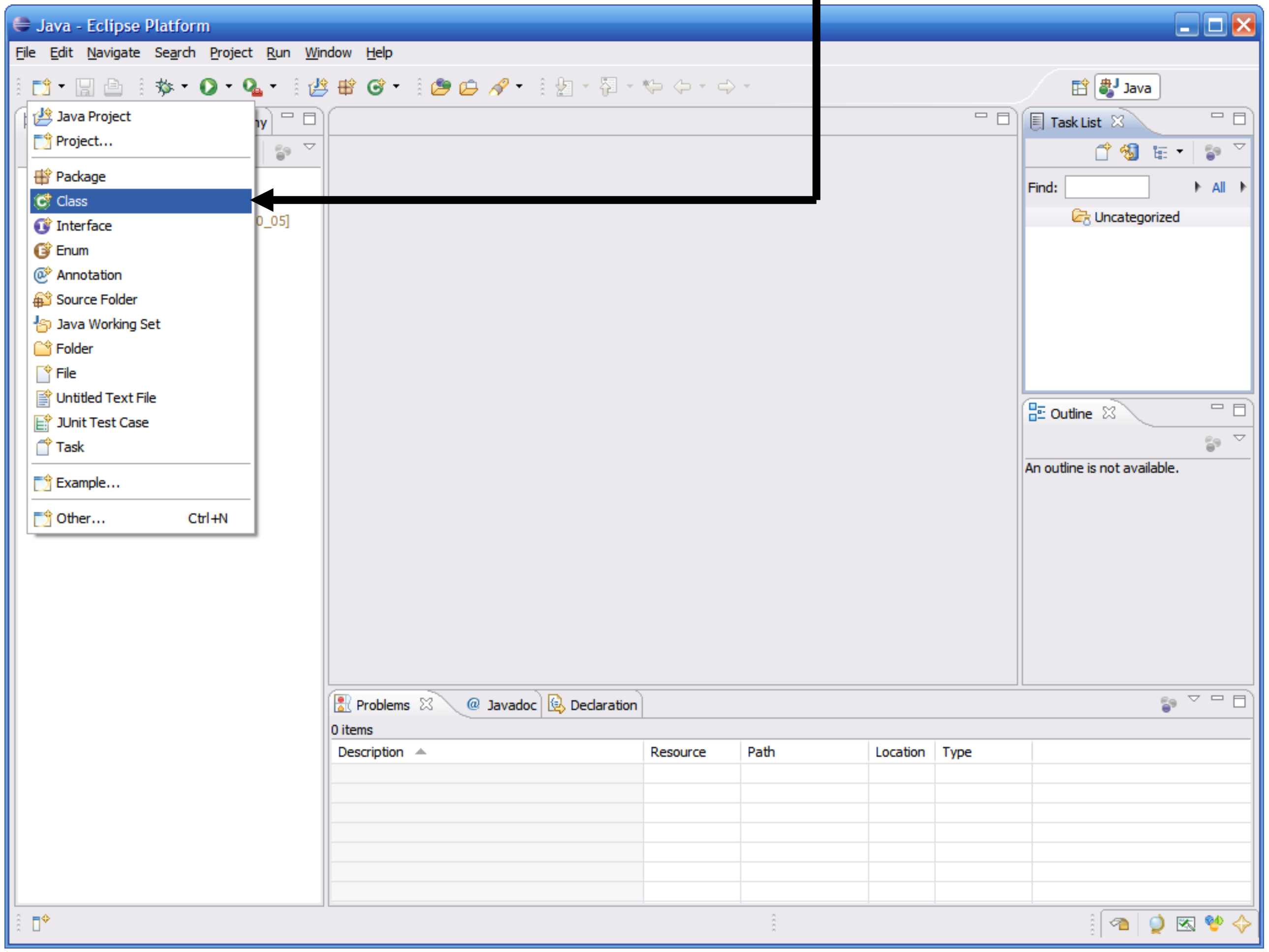


An outline is not available.

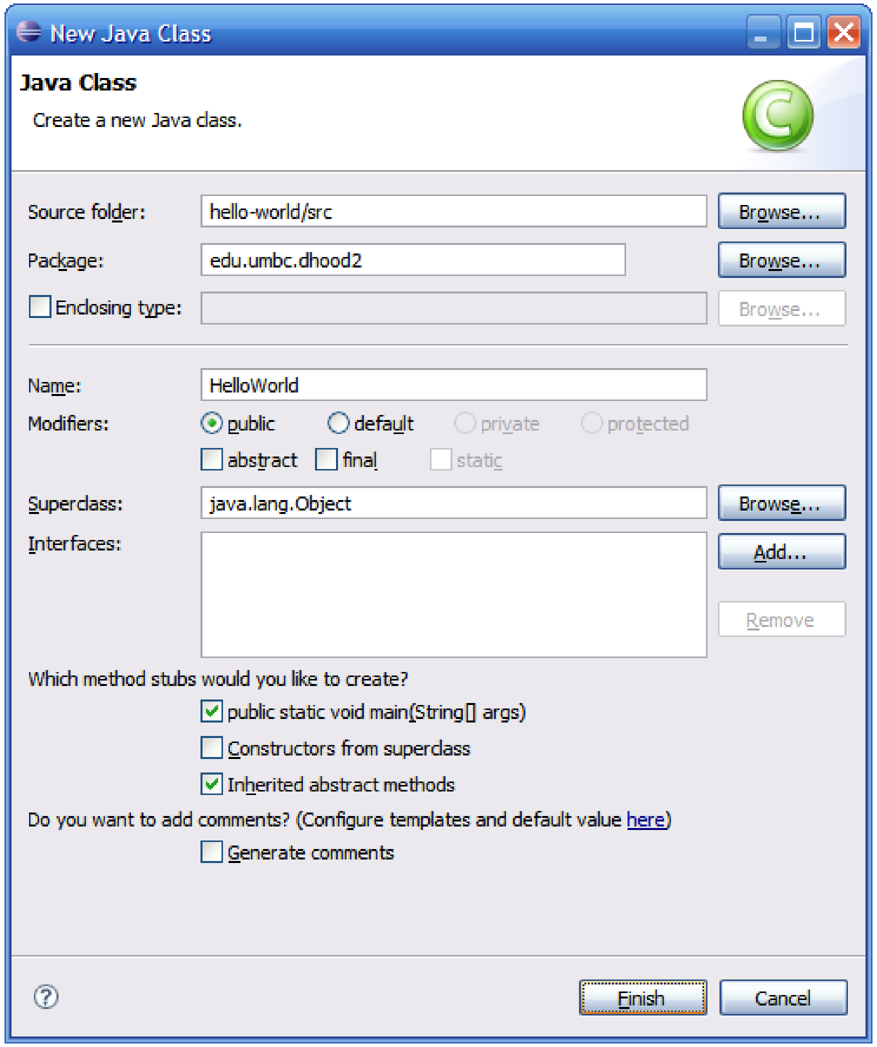
0 items

Creating a Class

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



Creating a Class (continued)



New Java Class

Java Class

Create a new Java class.

Source folder: hello-world/src

Package: edu.umbc.dhood2

[ ] Enclosing type:

Name: HelloWorld

Modifiers: [✓ public] default private protected

[ ] abstract [ ] final [ ] static

Superclass: java.lang.Object

Interfaces:

Which method stubs would you like to create?

[✓] public static void main(String[] args)

[ ] Constructors from superclass

[✓] Inherited abstract methods

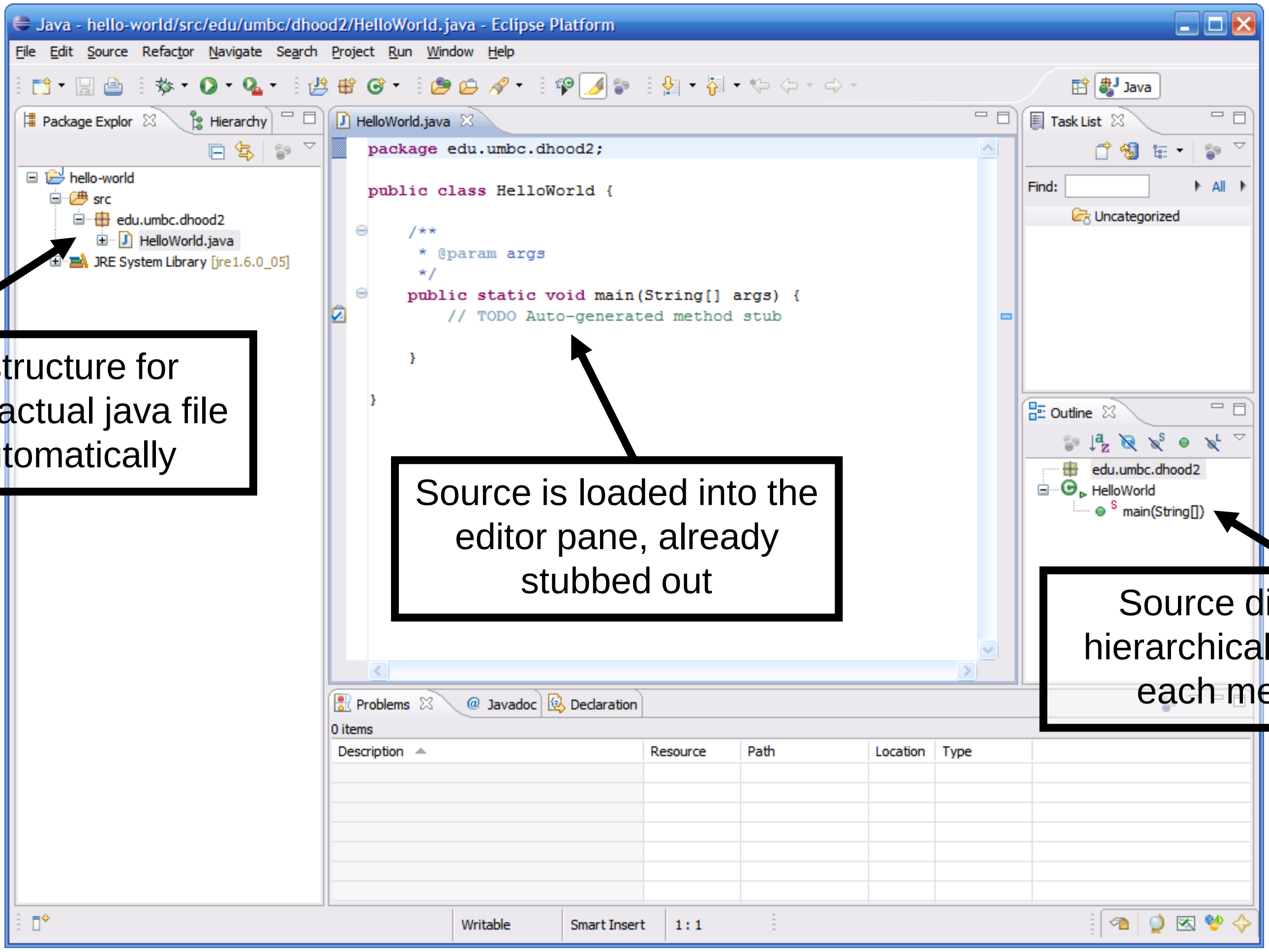
Do you want to add comments? (Configure templates and default value here)

[ ] Generate comments

- 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…



```
package edu.umbc.dhood2;

public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {
        // TODO Auto-generated method stub

    }

}
```

Directory structure for package and actual java file created automatically

Source is loaded into the editor pane, already stubbed out

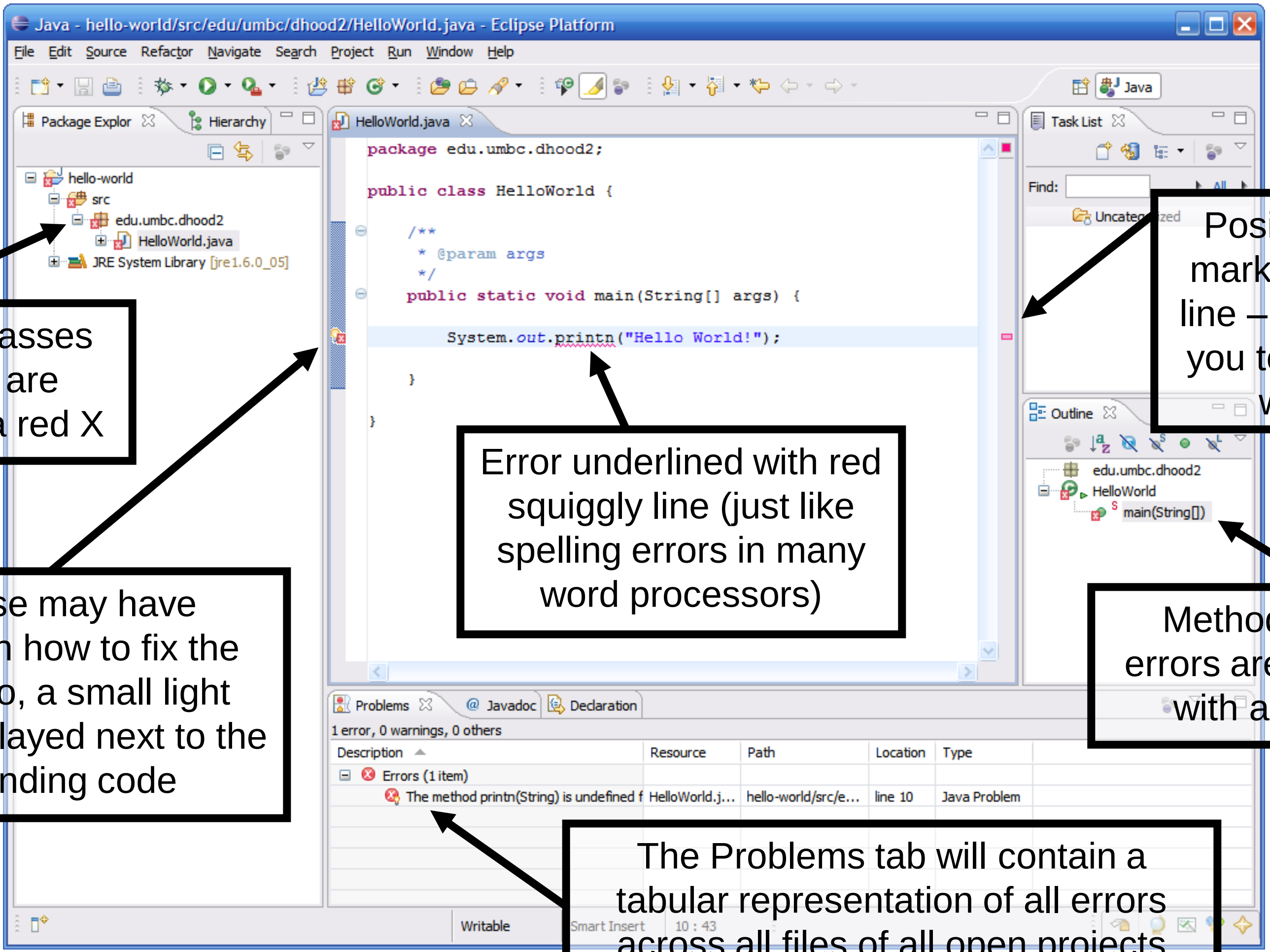
Source displayed in a hierarchical fashion listing each method name

Compiling Source Code

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

Example Compilation Error

- This code contains a typo in the println statement…



```
package edu.umbc.dhood2;

public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {

        System.out.printn("Hello World!");

    }

}
```

1 error, 0 warnings, 0 others

Description	Resource	Path	Location	Type
Errors (1 item)				
The method printn(String) is undefined f	HelloWorld.j…	hello-world/src/e…	line 10	Java Problem

Packages/Classes with errors are marked with a red X

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)

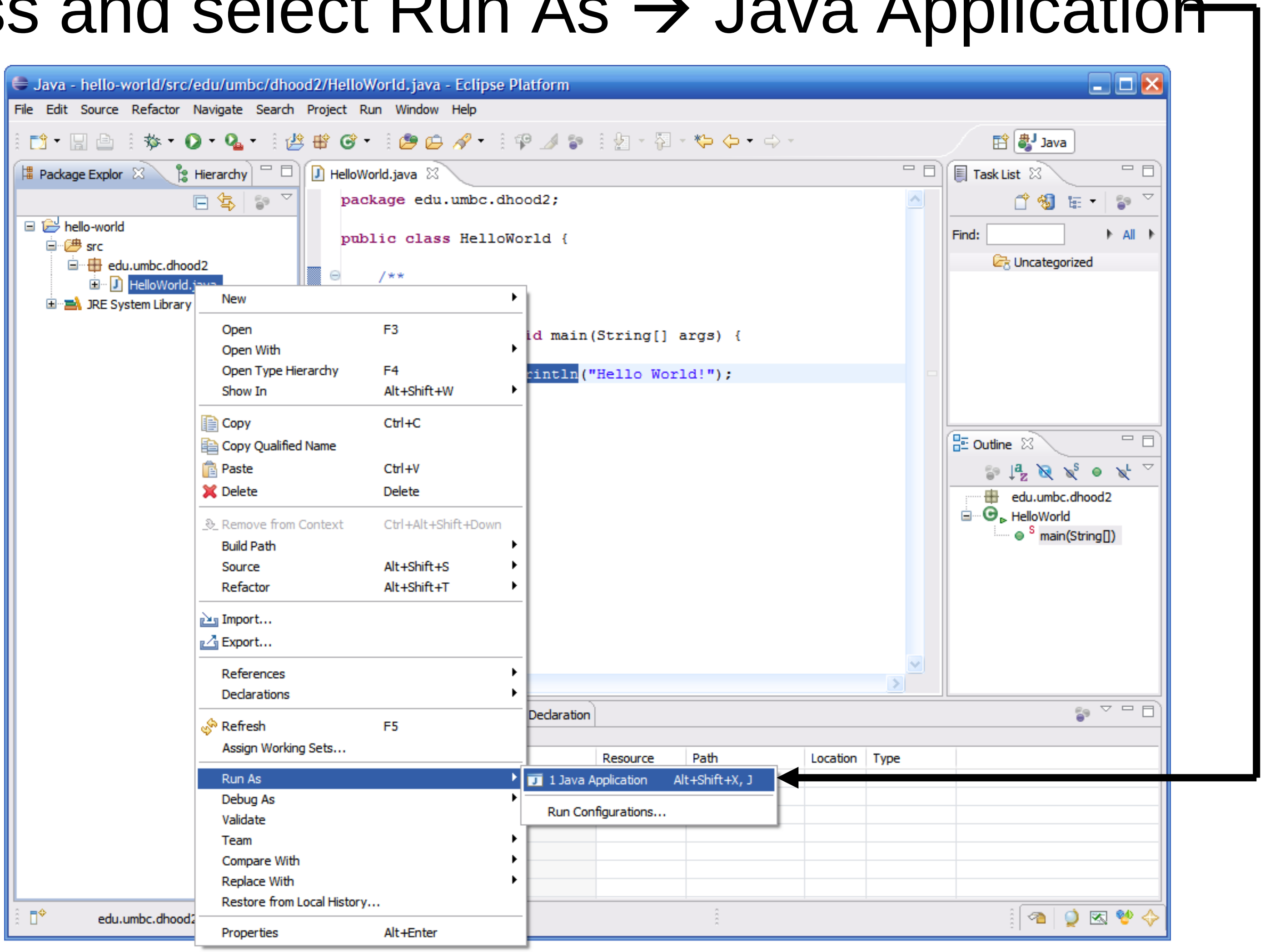
Position in file is marked with a red line – 1 click allows you to jump to line with error

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

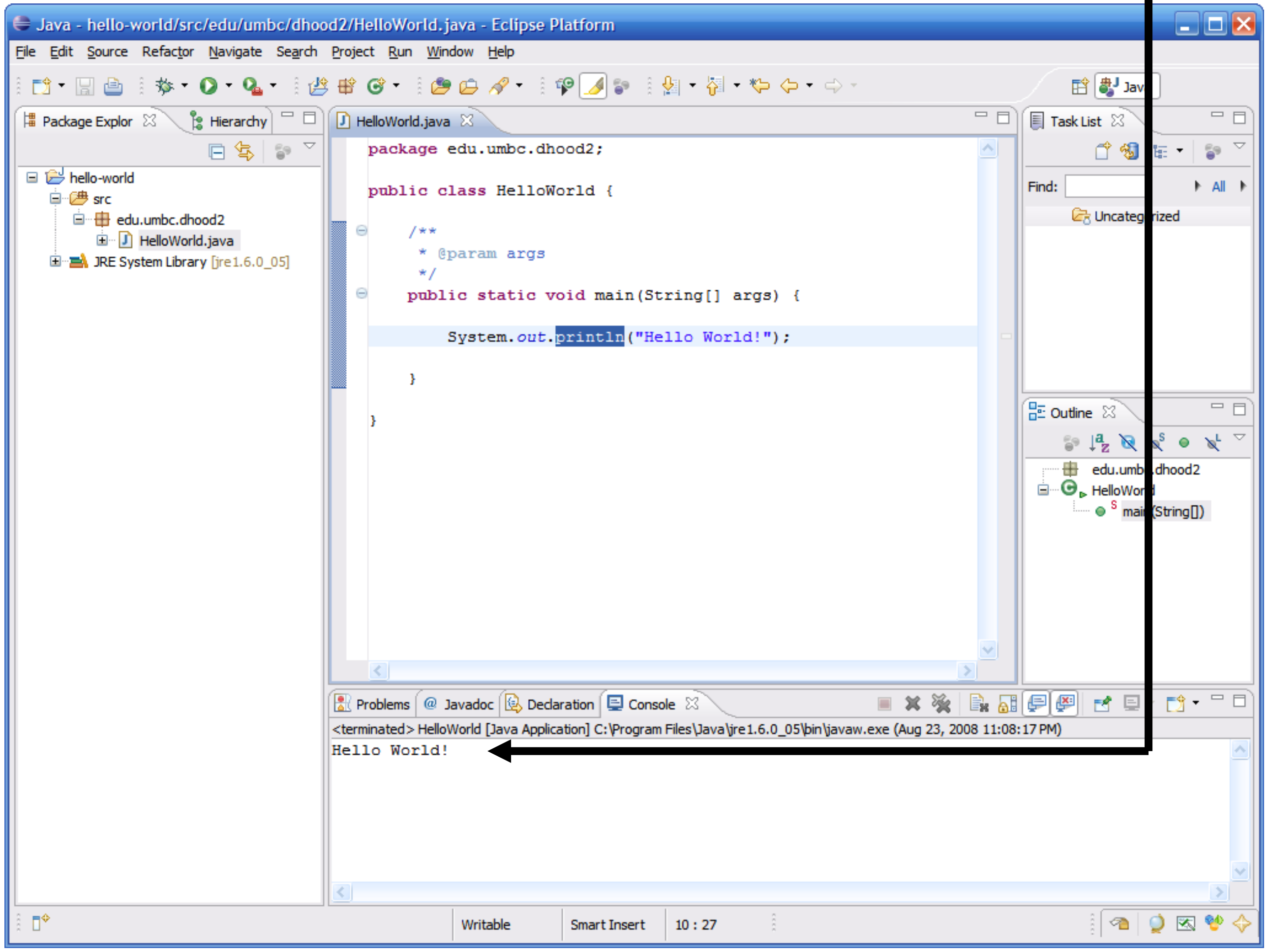
- An easy way to run code is to right click on the class and select Run As → Java Application



```
package edu.umbc.dhood2;

public class HelloWorld {

    /**

id main(String[] args) {

intln("Hello World!");
```

Running Code (continued)

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

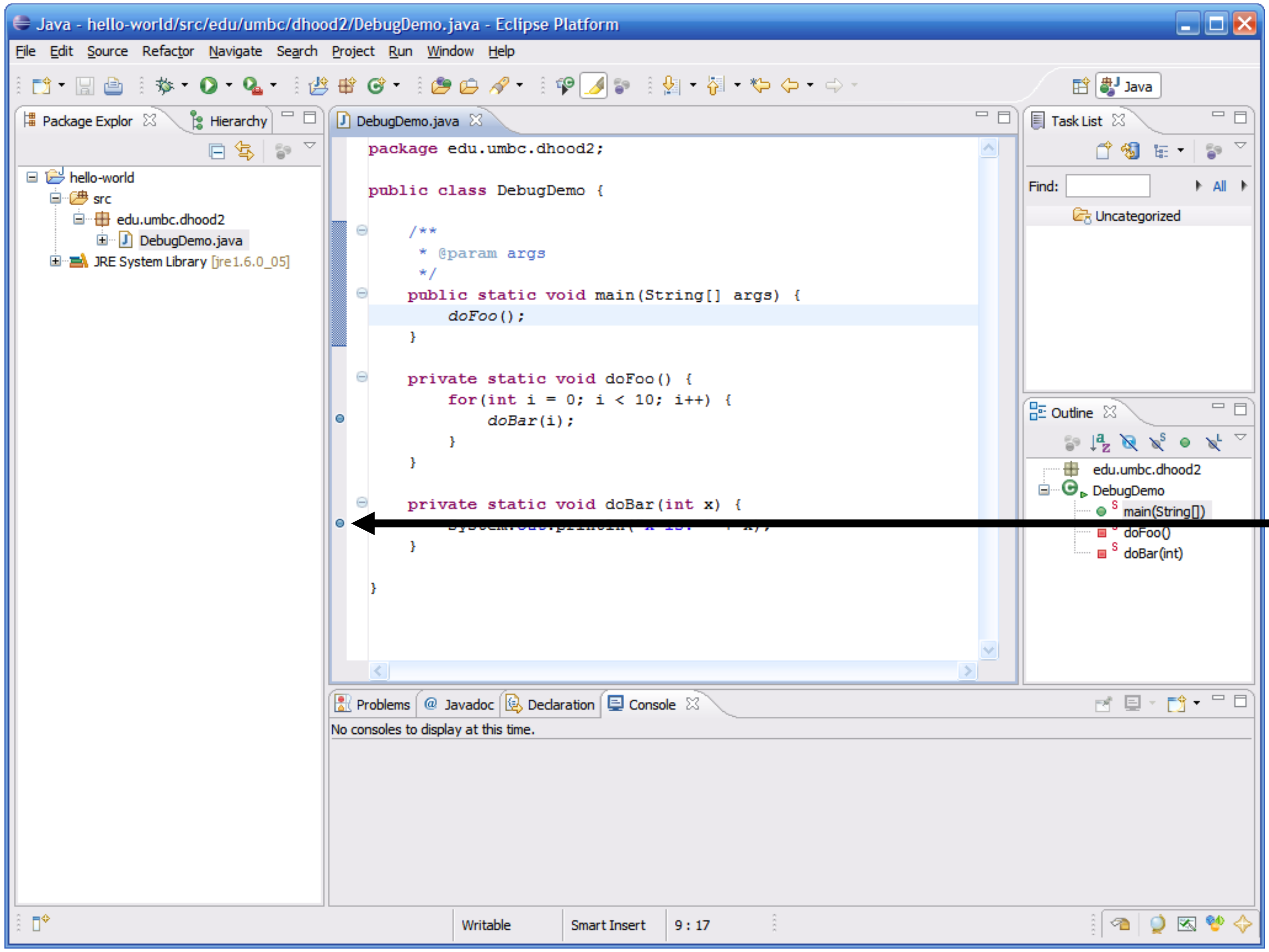


```java
package edu.umbc.dhood2;

public class HelloWorld {

    /**
     * @param args
     */
    public static void main(String[] args) {

        System.out.println("Hello World!");

    }

}
```

```
<terminated> HelloWorld [Java Application] C:\Program Files\Java\jre1.6.0_05\bin\javaw.exe (Aug 23, 2008 11:08:17 PM)
Hello World!
```

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



Java - hello-world/src/edu/umbc/dhood2/DebugDemo.java - Eclipse Platform

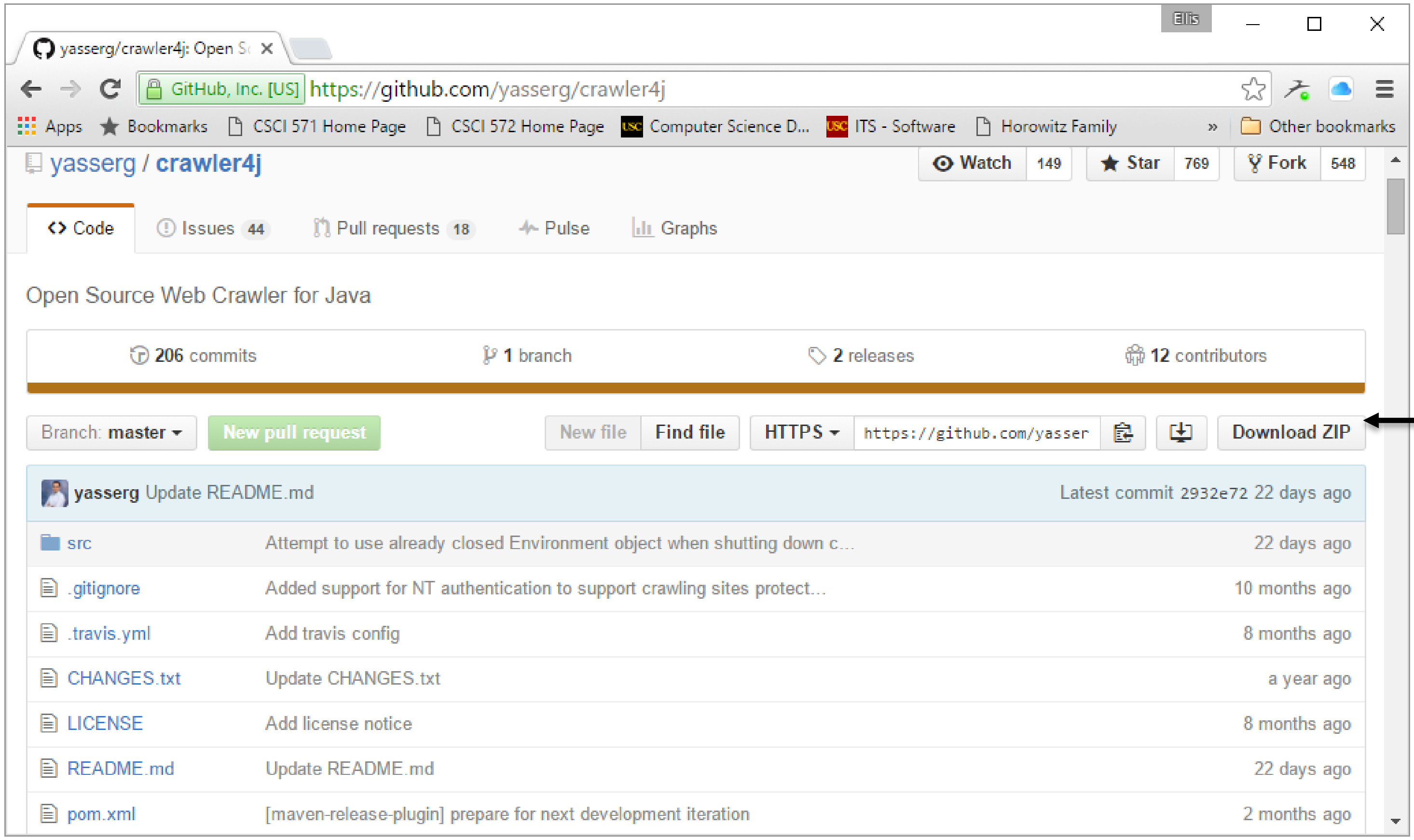
```
package edu.umbc.dhood2;

public class DebugDemo {

    /**
     * @param args
     */
    public static void main(String[] args) {
        doFoo();
    }

    private static void doFoo() {
        for(int i = 0; i < 10; i++) {
            doBar(i);
        }
    }

    private static void doBar(int x) {
        System.out.println("x is: " + x);
    }

}
```

No consoles to display at this time.

Instructions for Installing Eclipse and Crawler4j

- the document describes
  - installation of Eclipse on either Windows or MacOS
    - you will need to set Windows environment variables pointing to your JDK
- 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 2015, GitHub reports having over 11 million users and over 29.4 million repositories

Downloading Crawler4j from GitHub


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

yasserg / crawler4j — Watch 149 · Star 769 · Fork 548

Code · Issues 44 · Pull requests 18 · Pulse · Graphs

Open Source Web Crawler for Java

206 commits · 1 branch · 2 releases · 12 contributors

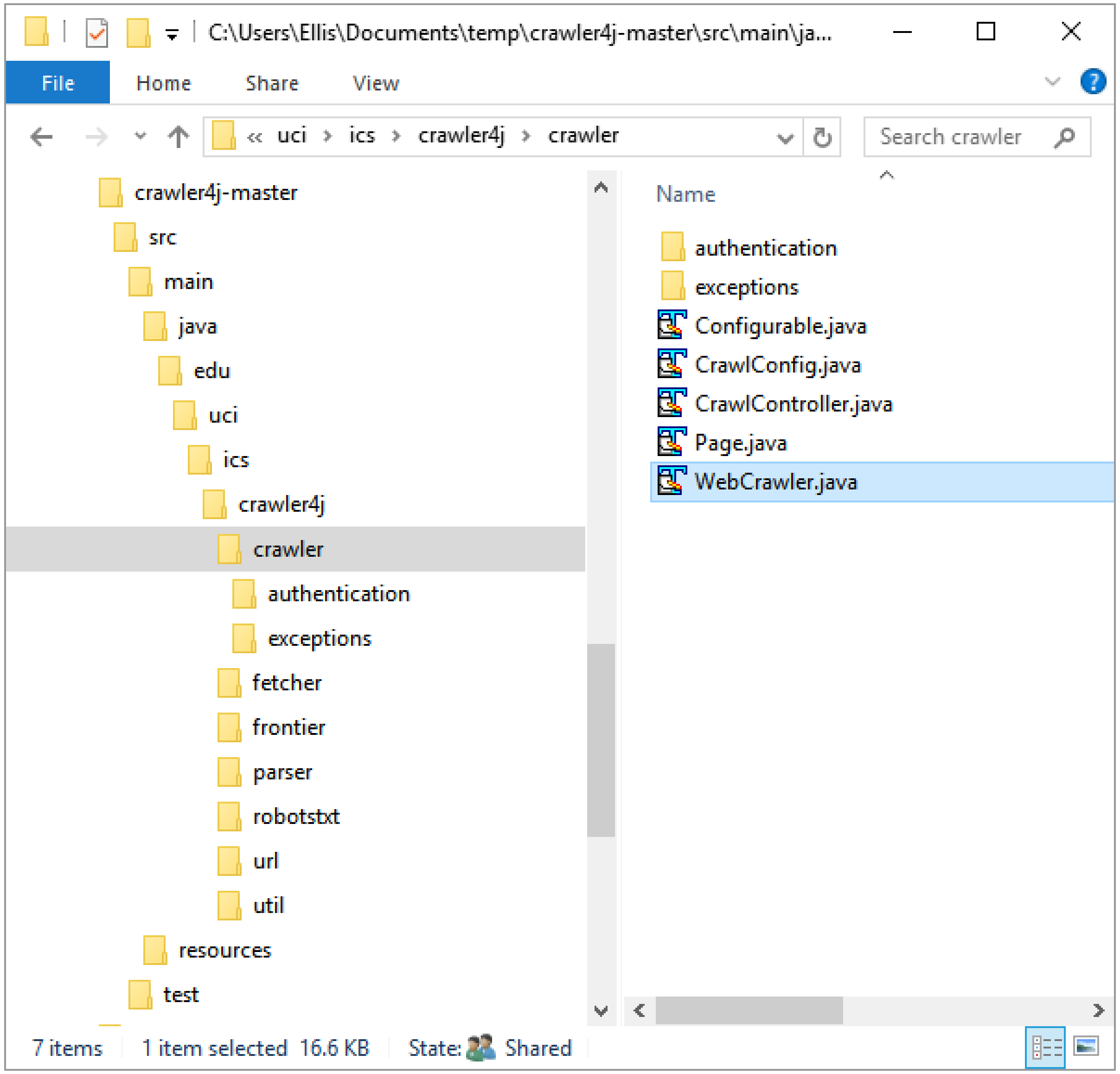
Branch: master · New pull request · New file · Find file · HTTPS · https://github.com/yasser · Download ZIP

yasserg Update README.md — Latest commit 2932e72 22 days ago

src	Attempt to use already closed Environment object when shutting down c…	22 days ago
.gitignore	Added support for NT authentication to support crawling sites protect…	10 months ago
.travis.yml	Add travis config	8 months ago
CHANGES.txt	Update CHANGES.txt	a year ago
LICENSE	Add license notice	8 months ago
README.md	Update README.md	22 days ago
pom.xml	[maven-release-plugin] prepare for next development iteration	2 months ago

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

Crawler4j Source code



C:\Users\Ellis\Documents\temp\crawler4j-master\src\main\ja...

« uci › ics › crawler4j › crawler

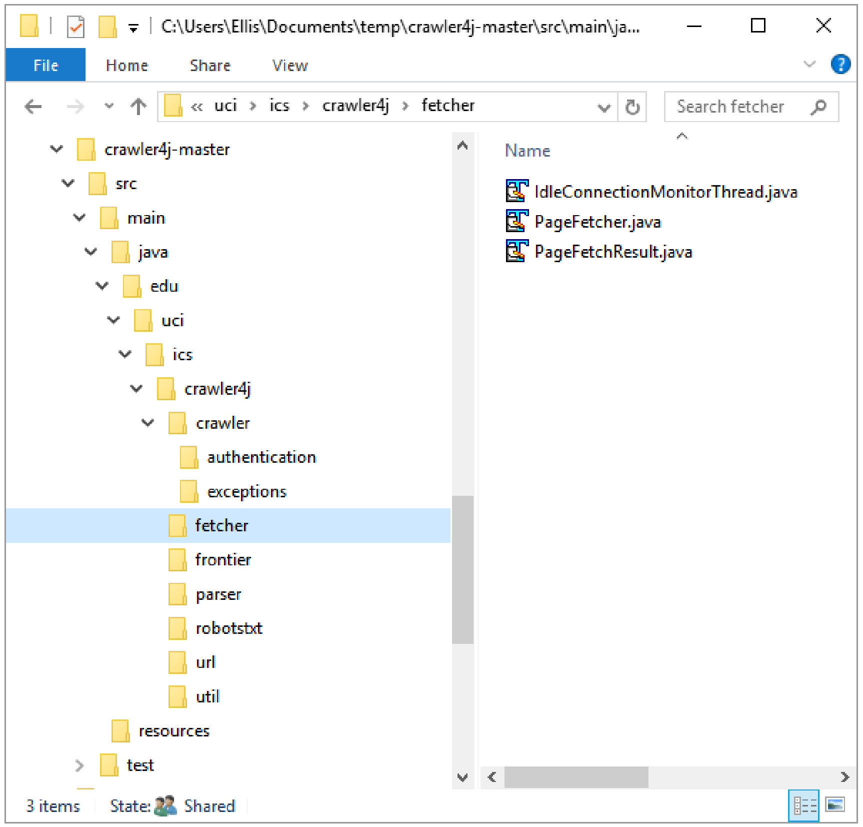
- crawler4j-master
  - src
    - main
      - java
        - edu
          - uci
            - ics
              - crawler4j
                - crawler
                  - authentication
                  - exceptions
                - fetcher
                - frontier
                - parser
                - robotstxt
                - url
                - util
      - resources
    - test

Name

- authentication
- exceptions
- Configurable.java
- CrawlConfig.java
- CrawlController.java
- Page.java
- WebCrawler.java

7 items   1 item selected 16.6 KB   State: Shared

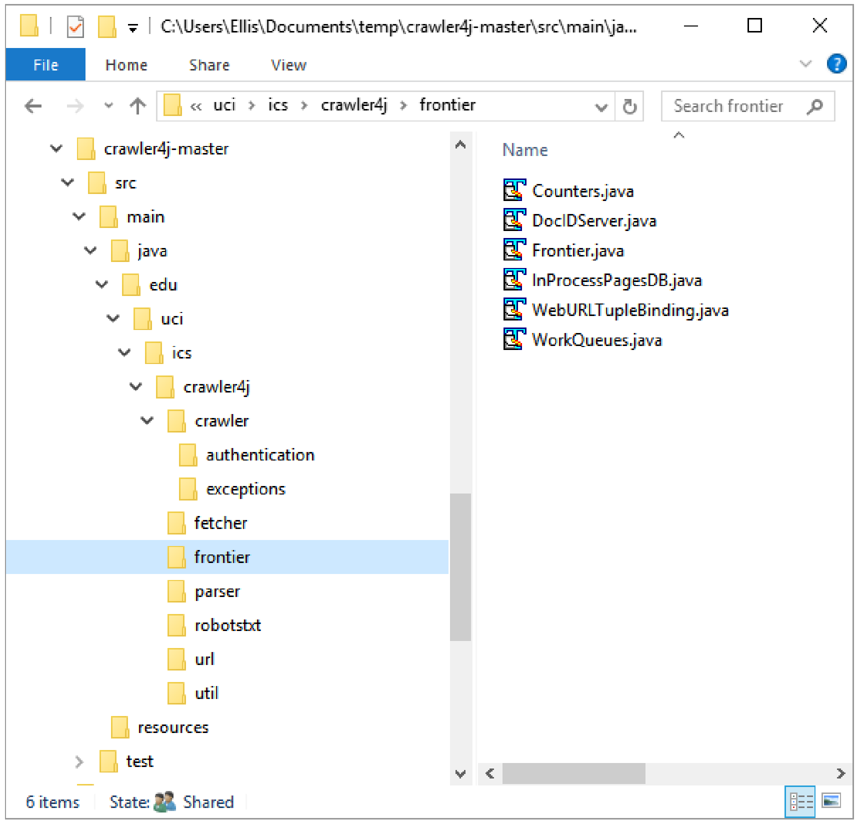
Crawler folder, a good place to start; look especially at WebCrawler.java


C:\Users\Ellis\Documents\temp\crawler4j-master\src\main\ja...
File
Home
Share
View
« uci › ics › crawler4j › fetcher
Search fetcher

- crawler4j-master
  - src
    - main
      - java
        - edu
          - uci
            - ics
              - crawler4j
                - crawler
                  - authentication
                  - exceptions
                - fetcher
                - frontier
                - parser
                - robotstxt
                - url
                - util
      - resources
    - test

Name
IdleConnectionMonitorThread.java
PageFetcher.java
PageFetchResult.java

3 items State: Shared


C:\Users\Ellis\Documents\temp\crawler4j-master\src\main\ja...
File
Home
Share
View
« uci › ics › crawler4j › frontier
Search frontier

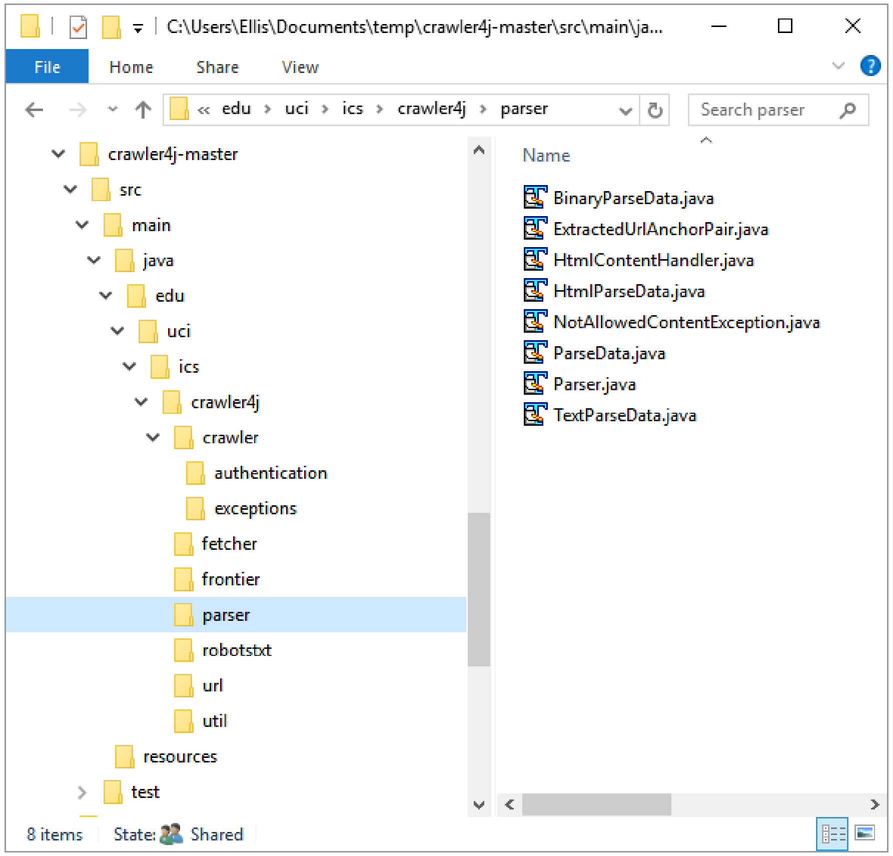
- crawler4j-master
  - src
    - main
      - java
        - edu
          - uci
            - ics
              - crawler4j
                - crawler
                  - authentication
                  - exceptions
                - fetcher
                - frontier
                - parser
                - robotstxt
                - url
                - util
      - resources
    - test

Name
Counters.java
DocIDServer.java
Frontier.java
InProcessPagesDB.java
WebURLTupleBinding.java
WorkQueues.java

6 items State: Shared

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

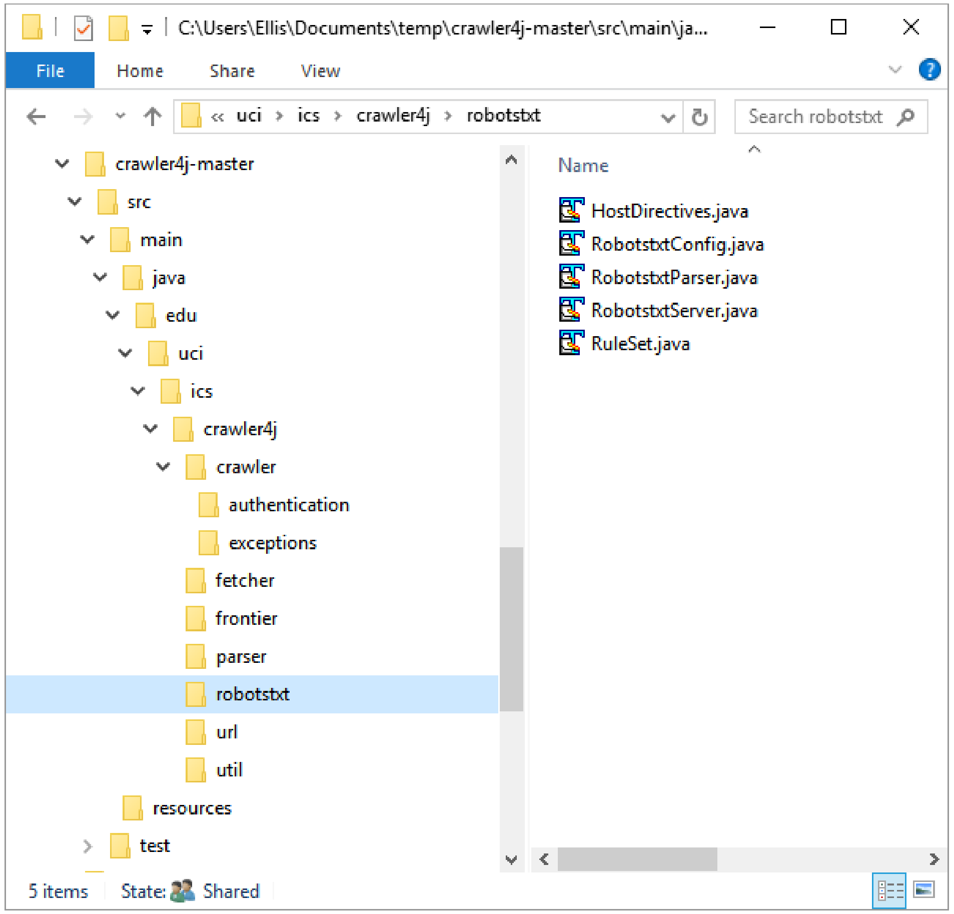

C:\Users\Ellis\Documents\temp\crawler4j-master\src\main\ja...

« edu › uci › ics › crawler4j › parser

```
crawler4j-master
  src
    main
      java
        edu
          uci
            ics
              crawler4j
                crawler
                  authentication
                  exceptions
                fetcher
                frontier
                parser
                robotstxt
                url
                util
      resources
    test
```

Name
- BinaryParseData.java
- ExtractedUrlAnchorPair.java
- HtmlContentHandler.java
- HtmlParseData.java
- NotAllowedContentException.java
- ParseData.java
- Parser.java
- TextParseData.java

8 items State: Shared


C:\Users\Ellis\Documents\temp\crawler4j-master\src\main\ja...

« uci › ics › crawler4j › robotstxt

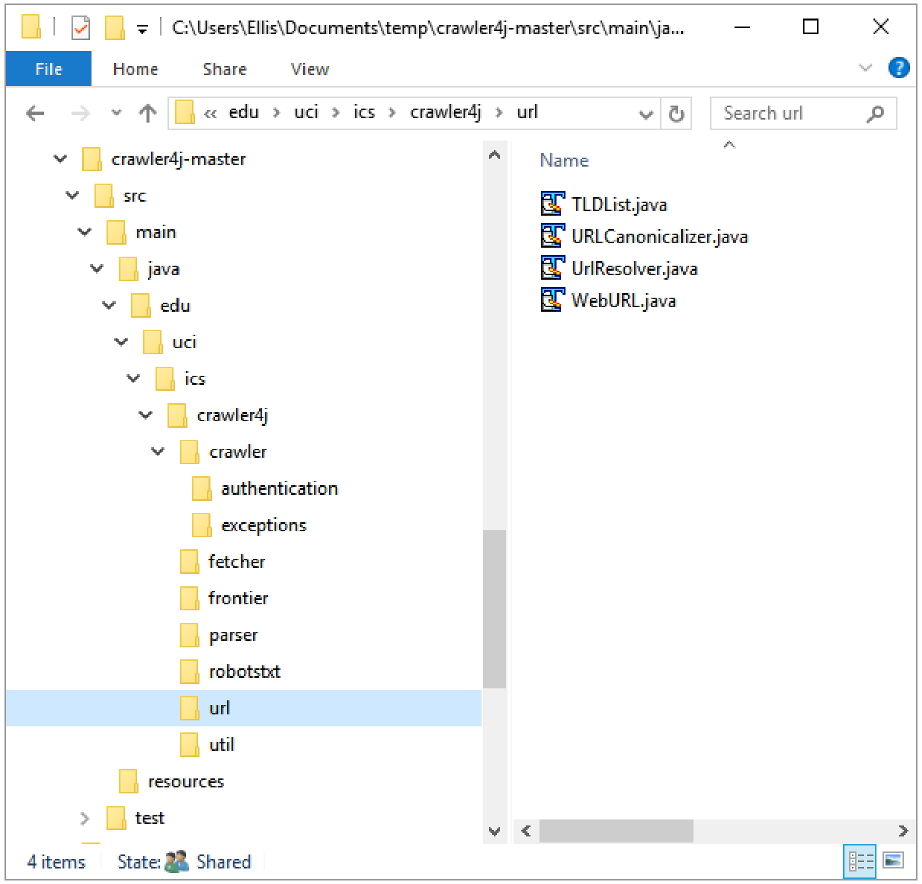
```
crawler4j-master
  src
    main
      java
        edu
          uci
            ics
              crawler4j
                crawler
                  authentication
                  exceptions
                fetcher
                frontier
                parser
                robotstxt
                url
                util
      resources
    test
```

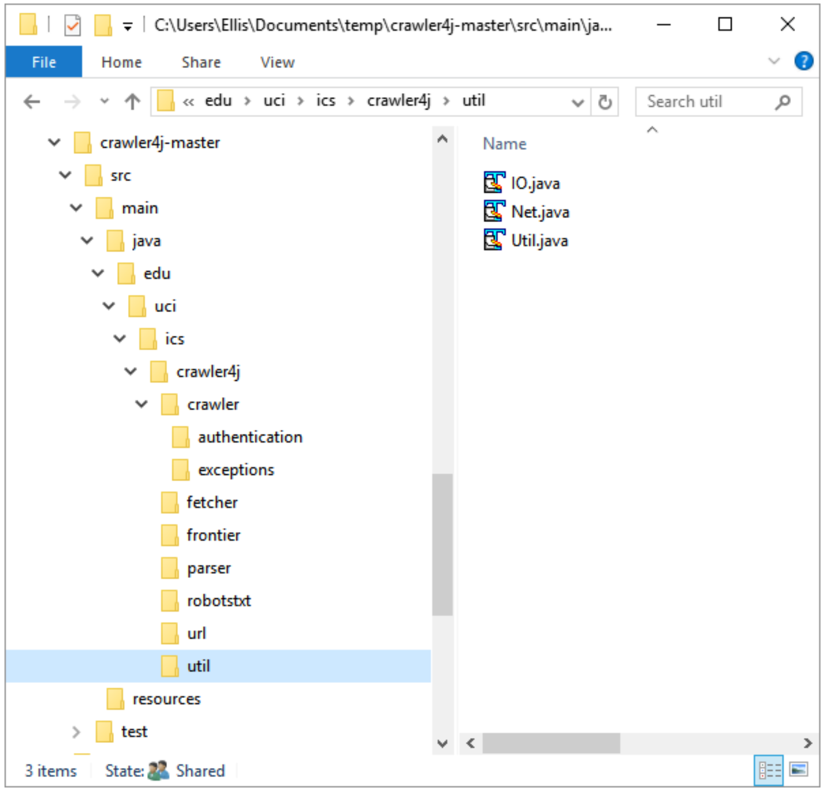
Name
- HostDirectives.java
- RobotstxtConfig.java
- RobotstxtParser.java
- RobotstxtServer.java
- RuleSet.java

5 items State: Shared

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





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>

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.viterbi.usc.edu/”);
        /*  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 going

crawling viterbi.usc.edu

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.viterbi.usc.edu/". 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.viterbi.usc.edu/");
     }
```

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 close to 5,000 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 will be much larger than 5,000 rows as it will have numerous repeated URLs

More things to Save

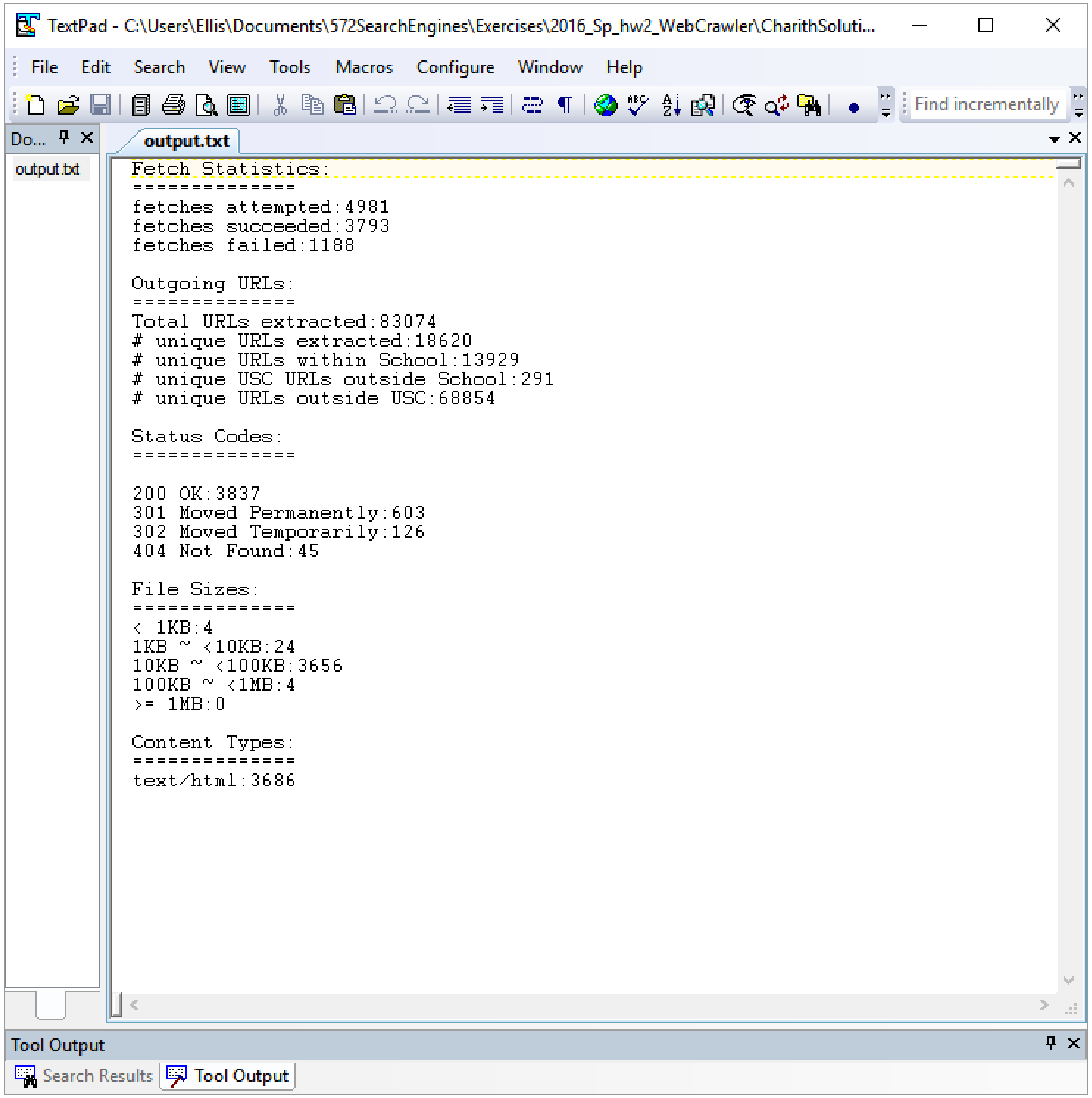
- you should save all of the downloaded web pages, etc. for processing in the next exercise.

Even More 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.
-

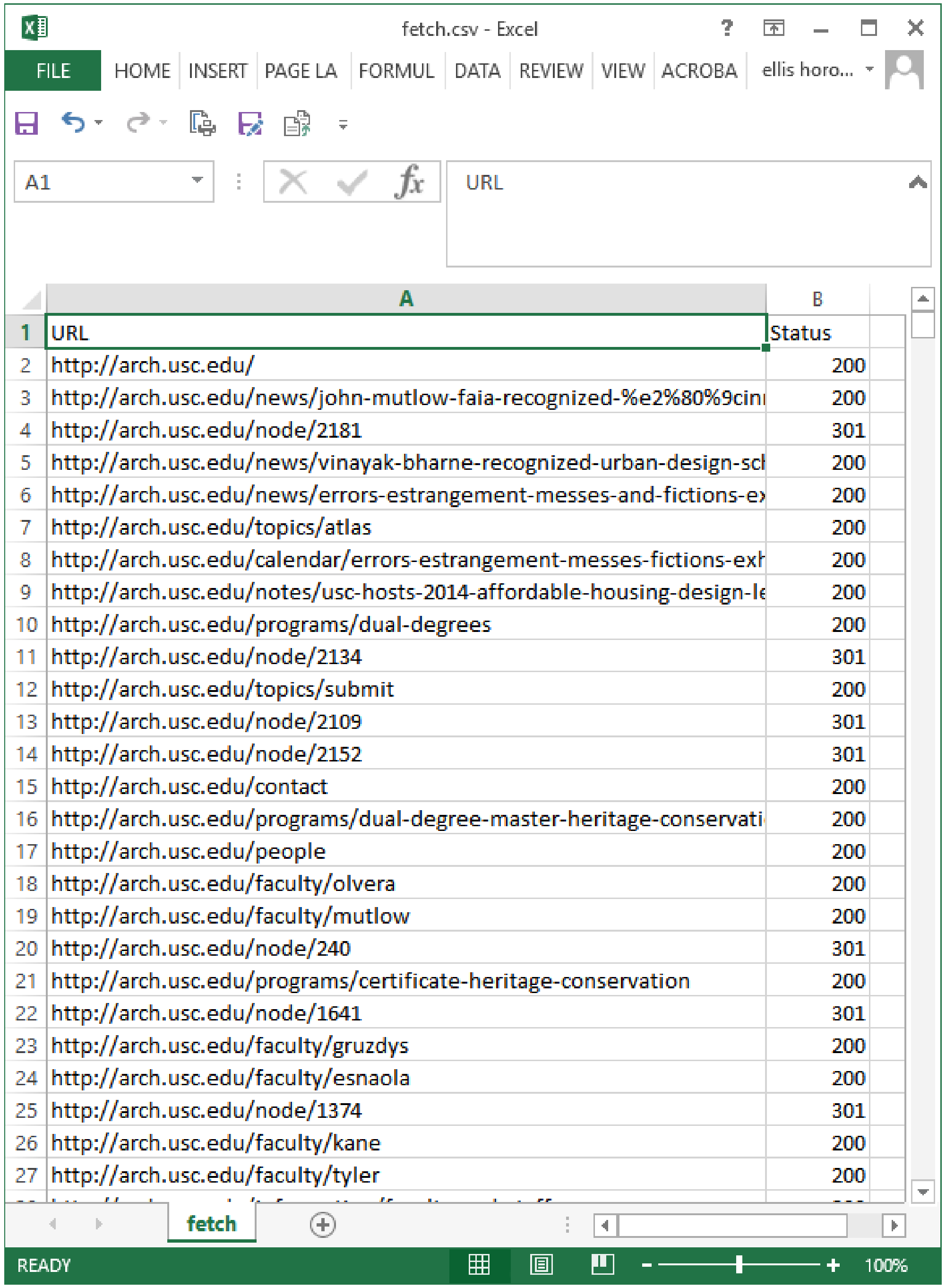
Outgoing URLs

- 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 School:
    The number of *unique* URLs encountered that are associated with the school website,
    i.e. the URL begins with the given root URL of the school.
  - # unique USC URLs outside School:
    The number of *unique* usc.edu URLs encountered that were *not* from the school website.
  - # unique URLs outside USC:
    The number of all other *unique* URLs encountered


TextPad - C:\Users\Ellis\Documents\572SearchEngines\Exercises\2016_Sp_hw2_WebCrawler\CharithSoluti...
File Edit Search View Tools Macros Configure Window Help
Find incrementally
output.txt
Tool Output
Search Results
Tool Output

```
Fetch Statistics:
==============
fetches attempted:4981
fetches succeeded:3793
fetches failed:1188

Outgoing URLs:
==============
Total URLs extracted:83074
# unique URLs extracted:18620
# unique URLs within School:13929
# unique USC URLs outside School:291
# unique URLs outside USC:68854

Status Codes:
==============

200 OK:3837
301 Moved Permanently:603
302 Moved Temporarily:126
404 Not Found:45

File Sizes:
==============
< 1KB:4
1KB ~ <10KB:24
10KB ~ <100KB:3656
100KB ~ <1MB:4
>= 1MB:0

Content Types:
==============
text/html:3686
```

Sample statistics Output



	A	B
1	URL	Status
2	http://arch.usc.edu/	200
3	http://arch.usc.edu/news/john-mutlow-faia-recognized-%e2%80%9cin	200
4	http://arch.usc.edu/node/2181	301
5	http://arch.usc.edu/news/vinayak-bharne-recognized-urban-design-sc	200
6	http://arch.usc.edu/news/errors-estrangement-messes-and-fictions-e	200
7	http://arch.usc.edu/topics/atlas	200
8	http://arch.usc.edu/calendar/errors-estrangement-messes-fictions-exh	200
9	http://arch.usc.edu/notes/usc-hosts-2014-affordable-housing-design-le	200
10	http://arch.usc.edu/programs/dual-degrees	200
11	http://arch.usc.edu/node/2134	301
12	http://arch.usc.edu/topics/submit	200
13	http://arch.usc.edu/node/2109	301
14	http://arch.usc.edu/node/2152	301
15	http://arch.usc.edu/contact	200
16	http://arch.usc.edu/programs/dual-degree-master-heritage-conservati	200
17	http://arch.usc.edu/people	200
18	http://arch.usc.edu/faculty/olvera	200
19	http://arch.usc.edu/faculty/mutlow	200
20	http://arch.usc.edu/node/240	301
21	http://arch.usc.edu/programs/certificate-heritage-conservation	200
22	http://arch.usc.edu/node/1641	301
23	http://arch.usc.edu/faculty/gruzdys	200
24	http://arch.usc.edu/faculty/esnaola	200
25	http://arch.usc.edu/node/1374	301
26	http://arch.usc.edu/faculty/kane	200
27	http://arch.usc.edu/faculty/tyler	200

Sample of fetch.csv file

Page Rank

- For Assignment #3 you will need to compute the Page Rank of each page you download in this assignment.
- To do that you will need to keep a record of all URLs contained in a given page. Therefore you should also generate a csv file that includes every successfully downloaded html file, in column 1, and the outgoing URLs that were contained in the page, in subsequent columns 2, 3, 4, etc. You need not submit this file with your assignment #2, but you will need it when you get to assignment #3.