NewsWhere

Assignment Out:	Wed, Feb 18
Design Meeting By:	10pm Mon, Feb 23
Code Due:	11:59pm Mon, Mar 2

---

1 Purpose

In this project, you will have a chance to use and integrate a few things you just learned in class, including

- the server-client model
- Swing (Client GUI)
- Java sockets
- HTML parsing
- multithreading

To implement the client-server model, you will work in pairs. Work can be divided however you wish though we recommend that you have one person work on the server and one person work on the client.

You will again deal with simple XML for input and output. And at the heart of the project, you will design your own algorithm to do something "intelligent". We will only provide the requirements for this project, and it is up you to decide how to satisfy them. Feel free to use any web resource that may help your design.

2 Where's News?

Given an URL to a news article, we want your application to tell in which country the news story happened. For example, a Japan-based reporter wrote a piece of news about the Martians landing in Australia and put it on http://www.TheMostReliableNews.co.jp/headline. With this URL, your application should return "Australia" (instead of Japan, where the news agency and web server might be located).

Your application should be a server that listens to a specific port for client requests. Each client request will contain URLs in a XML format message. Each URL will point to a news story and for each it should figure out in which country the news story happened. To do that you probably would

want to retrieve the web page at the URL and parse it. Sometimes your application may think "it is most likely in country A, but it could also be in country B". So instead of one specific country, we want a list of country-probability pairs, indicating that with proability Pi the news story took place in country Ci. If you are 100 percent sure that it happened in country C, you just return a list containing one pair, C and 1. Your server should then send the lists back to the client, also in XML format.

2.1 Client Side GUI

Your client will have GUI in which a user can input URLs to search. To receive input, your client must at the minimum:

- Have a textbox in the frame GUI (which will hold the Mapwidget component) where you can enter one URL for input. The client will form XML out of this single request and send that to the server.
- Have a textbox to input an already formed XML on the disk. The client will then send that file to the server.

Add-ons to the GUI may result in extra credit (see below).

2.2 Input

Your application will want use java.net.ServerSocket to accept requests from network clients. The port to listen will be specified as an command line argument: "-p port_number". Once connected, the client will send a message in XML, consisting of two tags:

REQUEST the root tag of the document. It has one required attribute, TIMEOUT, which is the maximum amount of time (in ms) the client is willing to wait for your result. It may take a long time to retrieve the web page, or it may take a long time for your algorithm to analyze the news, but in any case your server should give the client a valid response before TIMEOUT.

LOCATION the only type of tag in REQUEST, with one required attribute: URL. For this application, the URL will be an HTTP address.

An example is given below:

```
<REQUEST TIMEOUT=5000>
  <LOCATION URL='http://www.newswherenews.com/07/02/10/a.html' />
  <LOCATION URL='http://www.newswherenews.com/07/02/9/b.html' />
  <LOCATION URL='http://www.anti-newswherenews.com/headlines.html' />
</REQUEST>
```

2.3 Output

For each client request, your application should respond with an XML format message. It has the following tags:

RESPONSE the root tag of the document. It has no attribute.

RESULT the only type of tag in RESPONSE, with one required attribute: URL. The value of URL attribute should be the same as the URL attribute in one LOCATION tag of the input.

WHERE the only type of tag in RESULT. It has two required attributes: COUNTRY and PROBABILITY, indicating the probability that the news (pointed by the URL) happened in a certain country.

For the sample input, a valid response message could be:

```
<RESPONSE>
  <RESULT URL='http://www.newswherenews.com/07/02/9/b.html'>
    <WHERE COUNTRY='U.K.' PROBABILITY=0.8 />
    <WHERE COUNTRY='Australia' PROBABILITY=0.15 />
    <WHERE COUNTRY='New Zealand' PROBABILITY=0.05 />
  </RESULT>
  <RESULT URL='http://www.newswherenews.com/07/02/10/a.html'>
    <WHERE COUNTRY='U.S.A.' PROBABILITY=1 />
  </RESULT>
</RESPONSE>
```

Note that only two results are returned. Maybe anti-newswherenews.com is very slow, or headlines.html does not exist, but in any case your application should respond with a valid XML document before the timeout, with or without the result for this URL. Note also that the order of results is not the same as the input, which is also fine (XML does not care about the order). You want to make sure the sum of probabilities in each RESULT is 1.

Once these results are returned, the client should update the mapwidget with the appropriate probabilities. You have to accumulate probabilities from multiple responses by adding the probabilities for each repeating country and normalizing it (the total should always be 1.0)

2.4 Multi-threading

Because you are serving multiple client requests, possibly concurrently, you should have each request served by a thread. There could be more than one URL in a request, so you could also have each URL to be processed by an individual thread. You could have a new thread created whenever work arrives and destroyed when its done, or you could have a pool of persistant worker threads that will be assigned work dynamically. It is up to you to design your multithreading scheme, but you should try to get the best results for each of your clients in time.

Thread pools should be used to get full credit, even if you're not multi-threading page requests. Make sure request time-outs work with the thread pools correctly. When the time-out expires, the server should send the result (whatever it has at that point), and not wait for any other processes to finish. If a thread is in the middle of processing and the time-out expires, you do not have to worry about interrupting that thread, as long as you send the response immediately. Also, make sure that references to threads are removed, so that they are garbage-collected.

2.5 Getting and parsing the web page

The tools (classes) you may need are in java.net and javax.swing.text.html.parser. Go read the javadoc (http://java.sun.com/j2se/1.5.0/docs/api/index.html). Also make sure you deal with bad connections and bad HTMLs. You do not want your NewsWhere server to go down just because someone gave you an incorrect URL or wrote a bad web page. In fact, depending on your approach, you may not need to parse the HTML structure of the web page (you can treat the HTML page as a long string).

2.6 The AI (a.k.a find out where the news happened)

We want you to be creative on this. There will be no standard solution. There is no limitation at all on how you accomplish this. A straight-forward approach may be constructing a dictionary of country names, and count how many times each one appeared in the news article. But what if it is domestic news and it only mentioned city names? What about domestic news of other countries? Sometimes reporters use the capital city to indicate the government or the country. And many articles actually gives the location in the first sentence, for example, "NEW YORK - Blowing snow ...". And what about different ways to call the same country (e.g. U.S., U.S.A., United States, America, etc). Bayesian analysis, anyone?

While we expect some reasonable accuracy, this is not an AI class. We recommend you to use some very simple algorithm, at least in the beginning. After you are done with everything else, you can go back and improve the perfomance of your algorithm. Ideally your design should be very modular, so that the algorithm can be easily replaced by another one.

Note: It is acceptable to do brute-force string comparison in your AI. However, if it takes over 30 seconds to process a single page, then you're doing something wrong.

3 Design and Testing

Please have a design and testing plan ready for your design meeting. As usual, please come to a TA on hours no later than the date stated at the top of this handout. If you rather do it earlier, great!

You should consider writing your own client to test your server. We may later provide our testing client, but don't count on it. Like other cs32 projects, feel free to share tests / test client with your fellow students.

4 Grading

Grades will be based on the design, the functionality, and your code. Correctness, concurrency, accuracy and error-handling will be tested. Extra credits may be given to very creative algorithms, highly modular and extensible designs, or very cleanly written code.

Examples of Extra Credit include:

- Format your GUI so that you can dynamically add or remove multiple URLs to the request.
- Have the client do a crawl on the given URLs. That means the client will load up the page, search for links there, and accumulate them to the XML request.
- Multi-threading each page request per client. If you do this, definitely use thread pools as well. If the time-out expires, that connection should send back whatever response it has immediately.

5 Handin

Make a README file which explains any quirks in your code, as well as any special instructions about running your program. You should also mention any design decisions you made which you found troubling. Your main class should be named "NewsWhereMain". Handin with the following command: /course/cs032/bin/cs032_handin newswhere.