Design Notes
Assignment 1

Majid Ghaderi

Disclaimer

These notes are based on my own implementation. I do not claim that my implementation is the simplest or the best. Feel free to use or disregard any of these suggestions as you wish.

Tips

- Start with a simple HTTP client without conditional GET.
- Consider text objects first and then extend your code to handle binary objects.
- Once basic GET works then add conditional GET and implement a catalog.

Program Structure

The high-level structure of my `UrlCache.getObject(String url)` method looks like the following:

```
1: open a TCP connection to the server
2: format the GET request and send it to the server
3: while not empty line do
4:     read a header line and process it
5: end while
6: read the body of the response and save it
7: close the socket and clean up
```

Reading Binary and Text form Socket

My suggestion is to read everything from the socket as a sequence of bytes using the low level input stream associated with the socket. That is, call the method `Socket.getInputStream()` to gain access to the byte input stream associate with the socket and then just use method

`InputStream.read()`. It is very easy to convert an array of bytes to a string using one of class String's constructors. You can also write a method to read the header part of the response line-by-line. Just keep reading bytes from the input stream until you see the sequence `"\r\n"`.

Even for writing text data to a socket, *e.g.*, HTTP headers, you can create a string object and then call `String.getBytes("US-ASCII")` to convert the string to a sequence of bytes that can be written to the low level socket stream, which can be obtained using `Socket.getOutputStream()`. Make sure to **flush** the output stream so that the data is actually written to the socket.

Parsing URL

The class String in Java is very powerful. Use method `String.split()` to breakdown the URL to its various components. You can split a string using different delimiters.

Array Size when Reading from Socket

Use a number that is a power of two. Each network packet is about 1500 bytes. So, choose a size that can accommodate a few packets, for example 10*1024 Bytes. You can experiment with this number to see how much effect it has on your download speed.

Saving Catalog

If a HashMap is used to implement the catalog, then `ObjectOutputStream` and `ObjectInputStream` classes can be used to write and read a HashMap object from/to a file.

Converting Last-Modified to Millisecond

You may find class `SimpleDateFormat` with pattern `"EEE, dd MMM yyyy hh:mm:ss zzz"` useful for this purpose. Here is an example:

```
String lastModified = "Thu,   01 Jan 1970 1:00:00 GMT";
SimpleDateFormat format = new SimpleDateFormat("EEE, dd MMM yyyy hh:mm:ss zzz");
Date date = format.parse(lastModified, new ParsePosition(0));
long millis = date.getTime();
```