Design Notes
Assignment 2

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.

Basic GET Request

The high-level structure of my code to get an object using a GET request 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"`. Once you have read the entire header, you can use input stream `read(byte[])` method to read from the socket chunk-by-chunk instead of byte-by-byte (for improved performance).

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.