CPSC 441
Assignment 1
Design Notes

Majid Ghaderi

Note: 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.

1. How to properly close a socket

You should first close the streams associated with the socket and then close the socket. Alternatively, you can call methods shutdownInput() and shutdownOutput() on your socket object.

2. How to read both binary and text objects form a 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 getInputStream() method to gain access to the byte input stream associate with the socket and then just use the read(byte[]) method. 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 "\r\n".

Even for writing text data to a socket, e.g., HTTP headers, you can create a string object and then call getBytes("US-ASCII") to convert the string to a sequence of bytes that can be written to the low level socket stream.

3. How to parse a URL given as a string

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

4. What is a good size for a byte array to read from or write to a 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.