HTTP client and server running a pared down version of HTTP/1.0

Problem Statement: In this assignment you will be asked to implement an HTTP client and server running a pared down version of HTTP/1.0. You will have to extend the client and server to make use of some of the application level headers we have studied in class. For those of you who already have socket programming experience, the basic part of this assignment will be fairly easy. I am therefore adding an extra credit component for those students who want a more challenging assignment. This project can be completed in either C/C++, Java, or other languages you prefer.

Tools Used: Eclipse as an IDE for writing the code in JAVA and Wireshark Legacy was used to check the TCP and HTTP requests and responses

Implementation:

CLIENT SIDE: Client_socket.java

Simple HTTP client that is implemented using java programming language that sends GET and PUT request to the server and prints out the server responses along with the file content that is returned from the server.

- Get command line arguments (hostname, port number, HTTP method i.e GET or PUT, path of the file) and construct the request based on the method name in HTTP/1.0.
- Connect to the server via connection-oriented socket.
- Establish socket connection between client and server. (host name & port number required).
- GET/PUT requests over the client-server connection.
- Receives server response and prints it. And this prints the file content as well.
- GET a file from 'real' web server as well as local server i.e localhost.

SERVER SIDE: Server_socket.java

The HTTP server receives request from client, processes it, constructs a valid response and sends back the response over a connection oriented socket. The server handles multiple requests.

- Runs the server on a specific port greater than 2047. As soon as the server starts, it waits for client request and it will handle up to 10 requests from client side and this can be modified later.
- When the client connection is accepted, it reads the HTTP requests.
- Gets references to socket input and output.
- Sends the response code in return i.e "200 OK", "404 Not Found" "200 OK File Created". Also send the file content and content length (Refer Console output below).

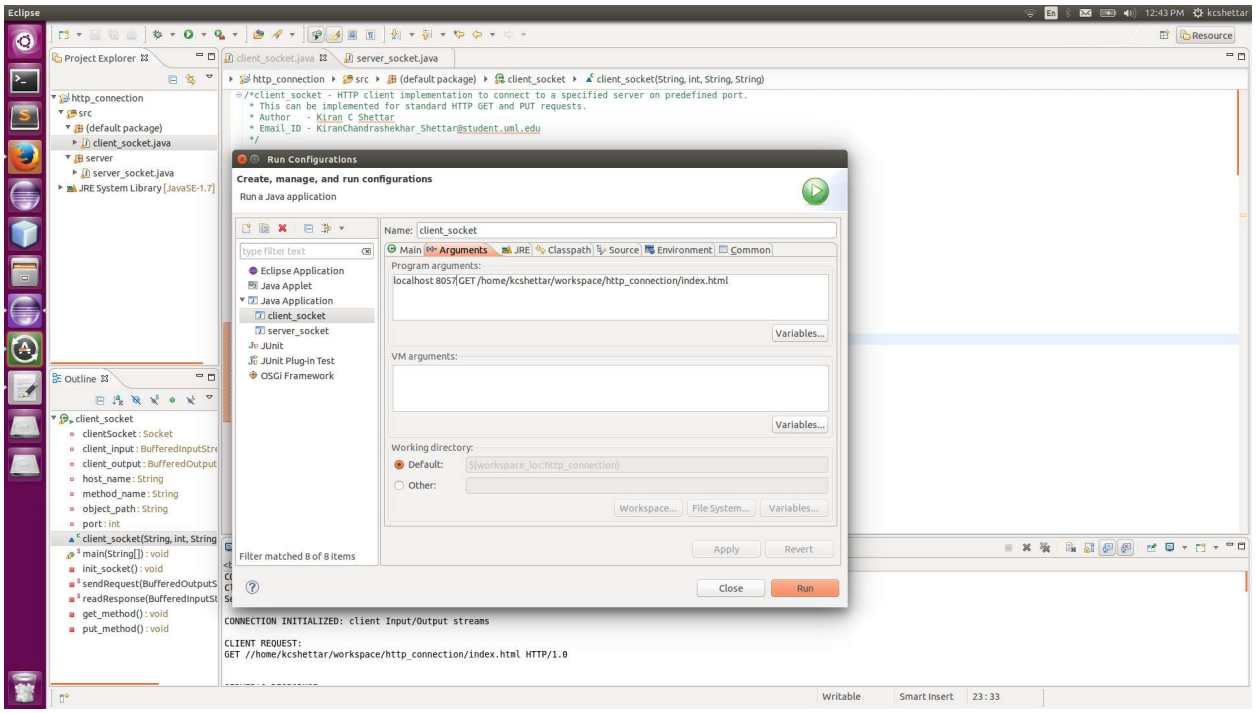
Tradeoffs & Extenstions:

- We can implement more headers in the server response.
- More server responses can be implemented depending on the client request.
- Server usage and client performance can be analyzed.
- We can implement other method requests such as TRACE, CONNECT, POST & DELETE.
- Server implementation - Extend to more file extensions (present - .htm/.html/.jpg).

TEST RESULTS:

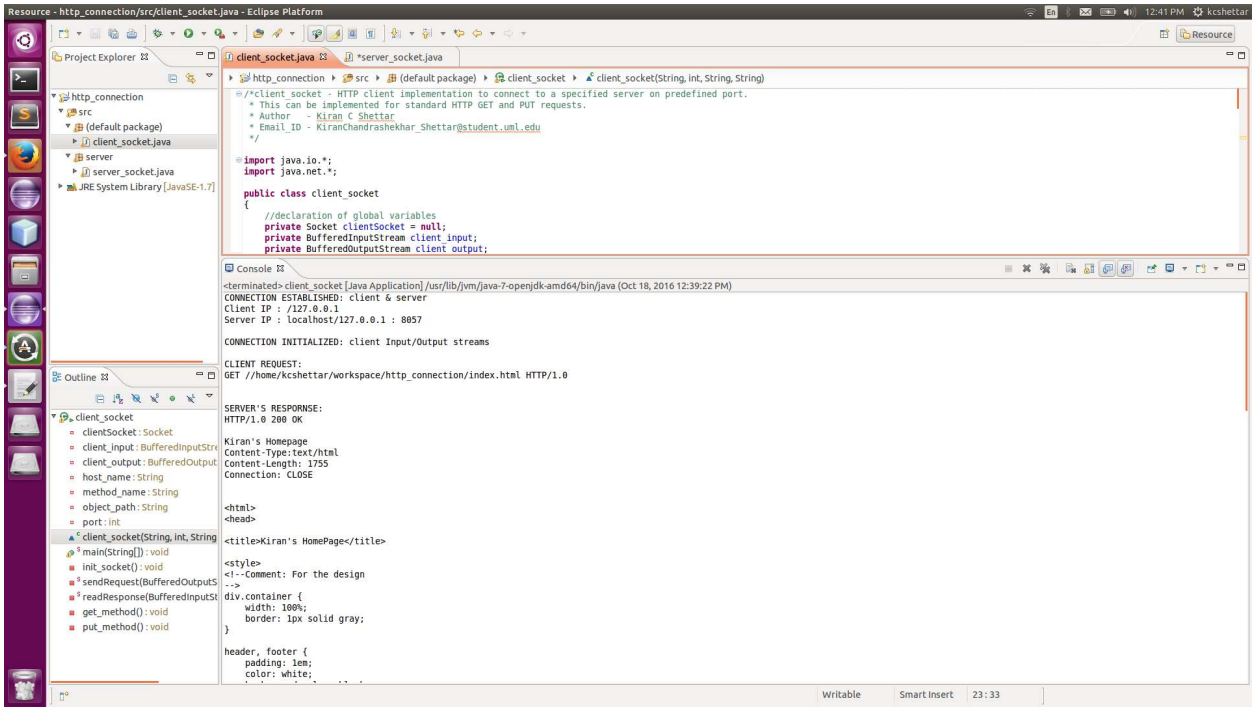
1. ***Eclipse Setup:*** Set the arguments to be passed for client run under:
   Run | Run Configurations| as it is displayed below.

   The arguments that are set as in the screen shot below would result in an HTTP GET request to localhost 127.0.0.1 for index.html on port 8057.



2. ***GET request:*** `localhost 8057 GET PATHNAME/index.html`

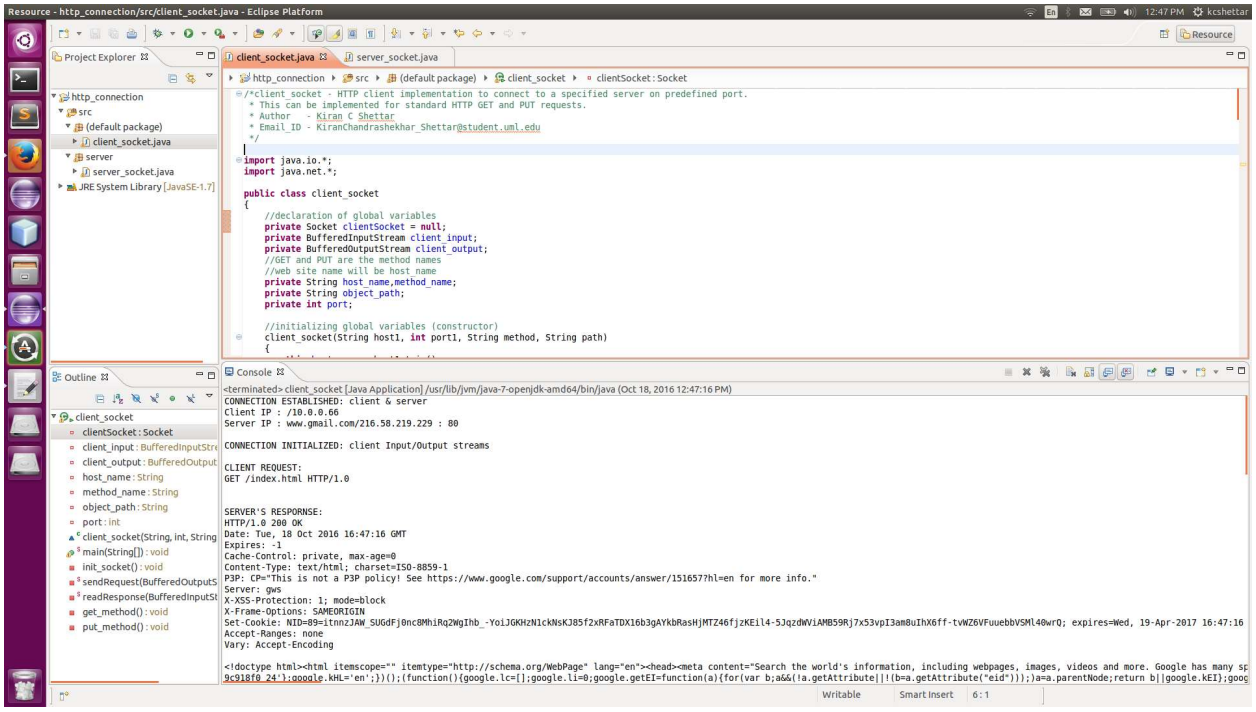
The arguments that are set as in the screen shot below would result in an HTTP GET request to localhost 127.0.0.1 for index.html on port number 8057.




```
Console Output: /_____GET_LOCALHOST_____/
SERVER: Waiting for client requests on port number 8057
CONNECTION ESTABLISHED: client & server
Client IP : /127.0.0.1
Server IP : localhost/127.0.0.1 : 8057
CONNECTION INITIALIZED: client Input/Output streams
CLIENT REQUEST:
GET //home/kcshettar/workspace/http_connection/index.html HTTP/1.0
SERVER'S RESPORNSE:
HTTP/1.0 200 OK
Kiran's Homepage
Content-Type:text/html
Content-Length: 1755
Connection: CLOSE
<html>
<head>
<title>Kiran's HomePage</title>
<style>
<!--MODIFIED
-->
</body>
</html>
```

3. *GET request:* `www.gmail.com 80 GET index.html`

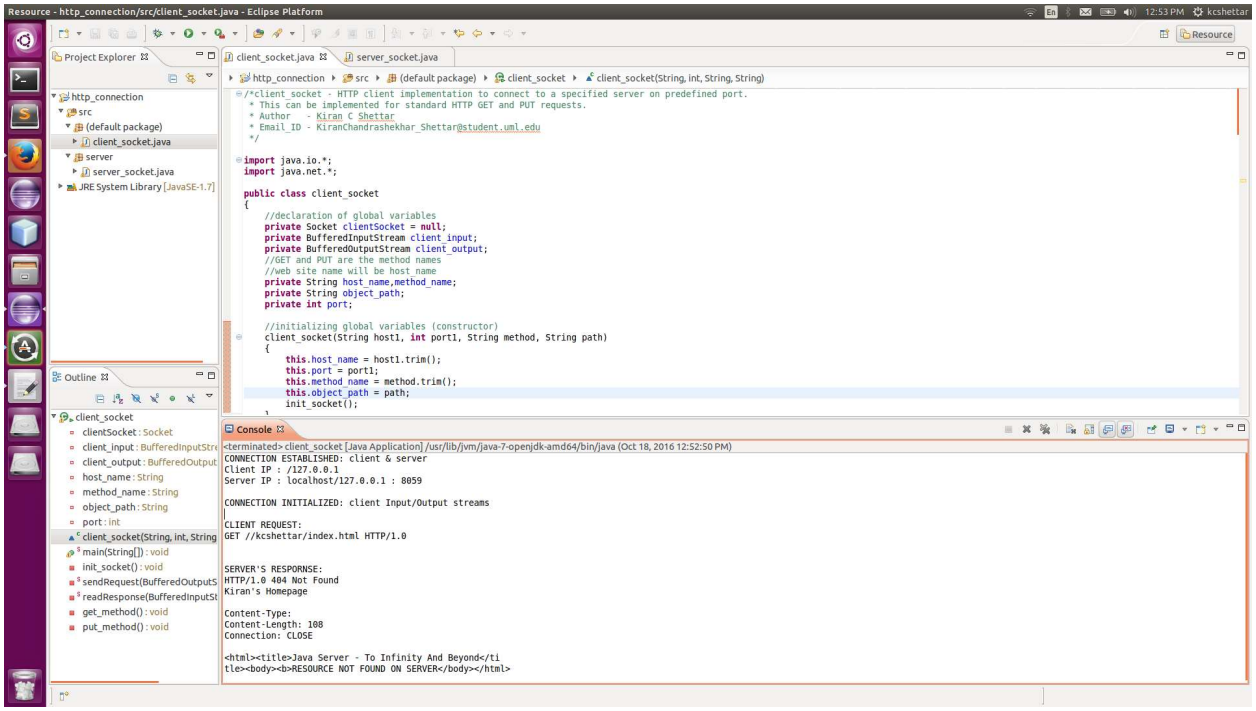
The arguments that are set as in the screen shot below would result in an HTTP GET request to www.gmail.com for index.html on global port number 80.




```
Console Output: /_____GET_GLOBAL_____/
CONNECTION ESTABLISHED: client & server
Client IP : /10.0.0.66
Server IP : www.gmail.com/216.58.219.229 : 80
CONNECTION INITIALIZED: client Input/Output streams
CLIENT REQUEST:
GET /index.html HTTP/1.0
SERVER'S RESPORNSE:
HTTP/1.0 200 OK
Date: Tue, 18 Oct 2016 16:47:16 GMT
Expires: -1
Cache-Control: private, max-age=0
Content-Type: text/html; charset=ISO-8859-1
P3P: CP="This is not a P3P policy! See
https://www.google.com/support/accounts/answer/151657?hl=en for more info."
Server: gws
X-XSS-Protection: 1; mode=block
X-Frame-Options: SAMEORIGIN
Set-Cookie: NID=89=itnnzJAW_SUGdFj0nc8MhiRq2WgIhb_-
YoiJGKHzN1ckNsKJ85f2xRFaTDX16b3gAYkbRasHjMTZ46fjzKEil4-
5JqzdWViAMB59Rj7x53vpI3am8uIhX6ff-tvWZ6VFuuebbVSMl40wrQ; expires=Wed, 19-Apr-2017
16:47:16 GMT; path=/; domain=.google.com; HttpOnly
Accept-Ranges: none
Vary: Accept-Encoding
<!doctype html><html itemscope="" itemtype="http://schema.org/WebPage"
<!--MODIFIED
-->
</script></div></body></html>
```

4. *GET request:* `localhost 8059 GET /kcshettar/index.html`

The arguments that are set as in the screen shot below would result in an HTTP GET request to localhost 127.0.0.1 for /kcshettar/index.html on port number 8059 where the file does not exist on the specified path. Which results for "404 Not Found".




```
Console Output: /_____GET_LOCAL_____WRONG_PATH_____/
SERVER: Waiting for client requests on port number 8059
CONNECTION ESTABLISHED: client & server
Client IP : /127.0.0.1
Server IP : localhost/127.0.0.1 : 8059
CONNECTION INITIALIZED: client Input/Output streams
CLIENT REQUEST:
GET //kcshettar/index.html HTTP/1.0
SERVER'S RESPORNSE:
HTTP/1.0 404 Not Found
Kiran's Homepage
Content-Type:
Content-Length: 108
Connection: CLOSE
<html><title>Kiran's Homepage</title>
<body><b>RESOURCE NOT FOUND ON SERVER</body></html>
```

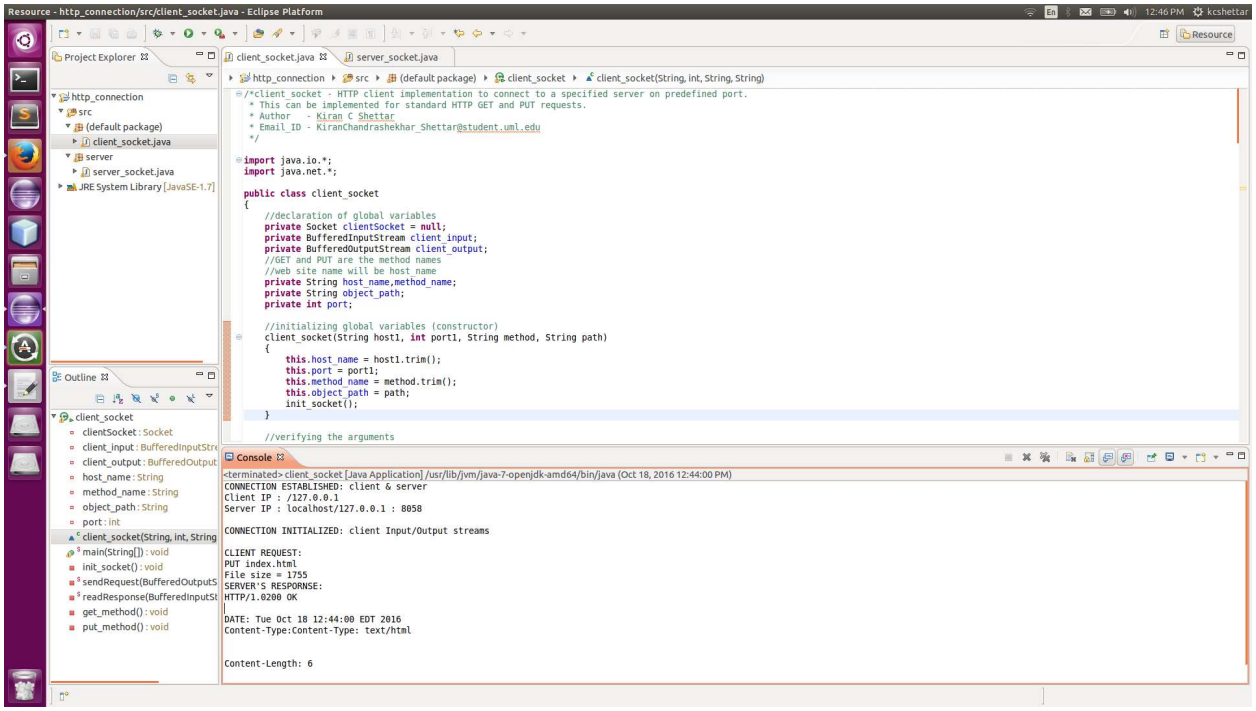
5. ***GET request:*** www.amazon.com 80 GET index.html

The arguments that are set would result in an HTTP GET request to www.amazon.com for index.html on global port number 80.

```
Console Output: /_____GET_GLOBAL_MOVED_PERMANENTLY_____/
CONNECTION ESTABLISHED: client & server
Client IP : /10.0.0.66
Server IP : www.amazon.com/54.239.25.192 : 80
CONNECTION INITIALIZED: client Input/Output streams
CLIENT REQUEST:
GET /index.html HTTP/1.0
SERVER'S RESPORNSE:
HTTP/1.1 301 Moved Permanently
Date: Tue, 18 Oct 2016 16:50:34 GMT
Server: Server
Location: https:///index.html
Content-Length: 227
Keep-Alive: timeout=2, max=14
Connection: Keep-Alive
Content-Type: text/html; charset=iso-8859-1
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>301 Moved Permanently</title>
</head><body>
<h1>Moved Permanently</h1>
<p>The document has moved <a href="https:///index.html">here</a>.</p>
</body></html>
```

6. *PUT request:* `localhost 8058 PUT index.html`

The arguments that are set as in the screen shot below would result in an HTTP PUT request to localhost 127.0.0.1 for index.html on port number 8058 which would create a file in return from the server.




```
Console Output: /_____PUT_LOCALHOST_____/
SERVER: Waiting for client requests on port number 8058
CONNECTION ESTABLISHED: client & server
Client IP : /127.0.0.1
Server IP : localhost/127.0.0.1 : 8058
CONNECTION INITIALIZED: client Input/Output streams
CLIENT REQUEST:
PUT index.html
File size = 1755
SERVER'S RESPORNSE:
HTTP/1.0200 OK
DATE: Tue Oct 18 12:44:00 EDT 2016
Content-Type:Content-Type: text/html
Content-Length: 6
```

FILE NAME | index.html | LOCALHOST | TEMPORARY FILE CREATED

```
<html>
<head>
<title>Kiran's HomePage</title>
<style>
<!--Comment: For the design-->
div.container {
    width: 100%;
    border: 1px solid gray;
}

header, footer {
    padding: 1em;
    color: white;
    background-color: black;
    clear: left;
    text-align: center;
}

naviagate {
    float: left;
    max-width: 160px;
    margin: 0;
    padding: 1em;
}

naviagate ul {
    list-style-type: none;
    padding: 0;
}

naviagate ul a {
    text-decoration: none;
}
article {
    margin-left: 170px;
    border-left: 1px solid gray;
    padding: 1em;
    overflow: hidden;
}

a:link {
    color: green;
    background-color: transparent;
    text-decoration: none;
}

a:hover {
    color: red;
    background-color: transparent;
    text-decoration: underline;
}

a:active {
    color: yellow;
    background-color: transparent;
    text-decoration: underline;
}
</style>
</head>
```

```
<body BGCOLOR="DCDCDC">
<div class="container">
<header>
   <h1>Kiran C Shettar</h1>
</header>

<naviagate>
       <ul>
              <a href="http://kcshettar.my-free.website">HomePage</a>
       </ul>
</naviagate>
<article>
       <h1>Graduate Student</h1>
       <address>Address:<br>90 White Street, #2<br>Lowell, MA, USA - 01854</address>
<symbol>&#9742</symbol> <a href="tel:1-669-262-7951">1-669-262-7951</a>
</article>

<footer>
       <a href="https://www.linkedin.com/in/kiran-shettar-4637b78b">LinkedIn</a>
</footer>

</div>
<p>Contact me:<a
href="mailto:KiranChandrashekhar_Shettar@student.uml.edu?Subject=UML%20Website%20Query
" target="_top"> Send an e-Mail</a></p>

<hr>

<p><a href="tempassn.html">Assignments</a></p>
</body>
</html>
```