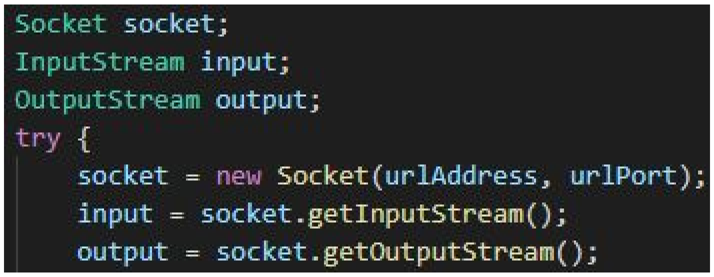
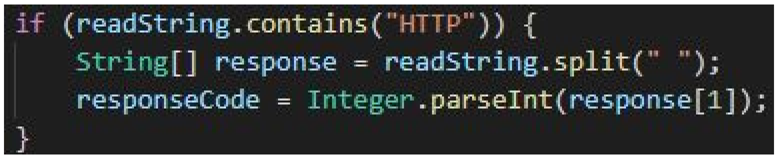
Computer Science 441: Assignment 1

Name: Jack Yang   UCID: 30062393

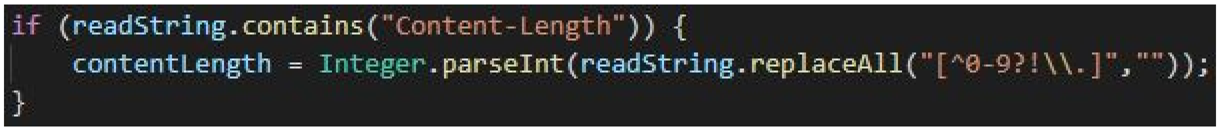
To send the GET request and read the server response, I used regular InputStream and OutputStream. To read the header of the server response, single bytes are read and stored in a byte buffer *buff* from the input socket and is also copied to a line byte buffer *buff1* of size 8K as a maximum of the header size. *Count* keeps track of where to copy the byte to in *buff1* and what the length is to convert to a String afterwards. If the byte read is ‘\r’ and followed by ‘\n’ then it is the end of the line and the whole line is converted to a String to be parsed. The only two things parsed are; the http response code, and the content length. If the line contains the String “HTTP” then the line also contains the response code. If the line contains the String “Content-Length” then the line contains the size of the body.



```
Socket socket;
InputStream input;
OutputStream output;
try {
    socket = new Socket(urlAddress, urlPort);
    input = socket.getInputStream();
    output = socket.getOutputStream();
```



```
if (readString.contains("HTTP")) {
    String[] response = readString.split(" ");
    responseCode = Integer.parseInt(response[1]);
}
```



```
if (readString.contains("Content-Length")) {
    contentLength = Integer.parseInt(readString.replaceAll("[^0-9?!\\.]",""));
}
```

After, if the response code is correct (200 - 299) then the filename is parsed from the urlPath String by taking the last string when split by “/”. Using the InputStream, the body is read and then written to the FileOutputStream similar to assignment 1.