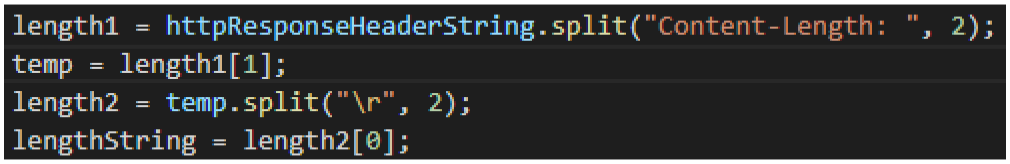
Reading Server response

The server response is read using a scanner to read the socket input stream and stored in the string httpResponseHeaderString. If “\r\n\r\n” is read from the stream to the httpResponseHeaderString at any point, we stop writing to httpResponseHeaderString as “\r\n\r\n” indicates that the end of the response header has been reached.

Parsing Server Response

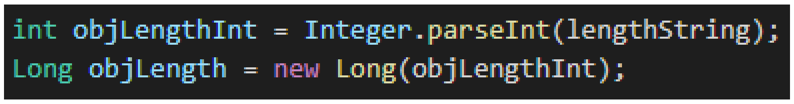
To parse the server response, the split() function is used to first split the response into two strings. The first string contains everything up to and including the “Content-Length: “ part of the response, and the second string contains everything after.

The split() function is then used again to isolate the number that represents the content length.



```
length1 = httpResponseHeaderString.split("Content-Length: ", 2);
temp = length1[1];
length2 = temp.split("\r", 2);
lengthString = length2[0];
```

Then the string lengthString is converted to an integer and stored in the int variable objLengthInt, which finally converted to a long and stored as objLength.



```
int objLengthInt = Integer.parseInt(lengthString);
Long objLength = new Long(objLengthInt);
```

The contains() function was also used to check if the request was properly received by the server by seeing if the response header contained the string “200 OK”.

Reading the Object Body

The object body was read using a scanner. The data from the scanner was then written to a byte array output stream to have the data be in chunks. Finally the byte chunks were written to the file using a file output stream.