# Summary

The project was to simulate a Web Browser using sockets and parsing the bytes received from the sockets. A socket is connected to an URL and the bytes are received using BufferedReader after sending the GET request to the socket. As they are received, the html code is parsed and shown as the output. For images, a separate GET request was sent was the bytes were received using DataInputStream and they are saved in a file with a image extension.

## Simulation of the program

I’ve simulated the program with the help of Sockets. I used JAVA as the programming language and NetBeans as client in Windows, eclipse in Linux and terminal in UNIX. I’ve created one class WebBrowser which extends JFrame. I’ve implemented both command line and GUI in the same class. If the arguments passed is zero, then the GUI pops up. If an argument is passed, then GUI is hidden and command line works.

For the GUI, I’ve added JFrame, a GO button, text bar as address bar, jEditorPane as View Area and an action Listener to listen to check when GO is clicked. When the GUI first pops up, the user is required to enter the URL in the address bar. After the URL is entered and GO is clicked, the URL is passed to loadURL method where the url is checked whether its image or not. If not, then it is passed to connectSocket method where the socket is created to the URL. If the URL sent is a image, then the URL is parsed to get the filename and the URL is used to connect to socket and the received bytes are saved to a file which is later used to display in the GUI.

In case of command line browser, all the methods used by the GUI method is used. The only difference is that the GUI would be set to not visible. All the line that are received from the socket are parsed. The parsed line are sent to the command line are using System.out.print command. Since the images cannot be displayed in UNIX terminal, a text is displayed which contains “Image: <filename>”.

## Difficulties faced

I encountered lots of difficulties in this project. First one was connecting to the socket. Connection to the socket did not take time but sending the correct GET request took a lot of my time. I finally figured out that the page sent after GET should not contain the host and that host should be sent in the next line as “Host: <host”

The next difficulty I faced was parsing the html. For the most part, I used regular expressions to reduce searching for tags and just searching for anything which looked like a tag such as ‘<\_\_\_>’. With this I was able to remove almost every tag and get the text between the tags. I searched for tags which leave leading and trailing empty lines like headings, paragraphs, line, horizontal lines and lists tags. I searched for these tags exclusively and inserted empty lines wherever necessary.

The third difficulty I faced was parsing the <img> tag. Since the images can be of various formats, I had to look at the formats first and then I had to use regular expressions to remove everything in the line except the file name. It was one of the most tedious tasks since there were different ways for the img tags to appear.

The last and most difficult problem I faced was downloading the image properly. When I first tried, the image got downloaded with the headers. Once I removed them using notepad++, the image got rendered properly. So I had to make sure that the image downloaded does not contain the headers. I found from the file that there was an empty line between the header part and the part from where the image starts. So I copied the first portion of the image into a string buffer and searched for \r\n\r\n. Once I got the first hit, I knew the start of the image. From there it was fairly easy. I also found that after the last byte is sent, sever takes some time to send -1 which says that the server does not have anything else to say.

## What I Learnt

I learnt how data is transferred between the sockets. The project also helped me understand how to download image into a file and display it.

# Result

I was able to remove every issue I faced except using a different port number. I tried every possible way but I couldn’t use any other port number other than 80.