Assignment 2, CPSC 441, Winter 2021, Lachlan Moore

In order to read and parse the response from the server I used a BufferedInputStream as it returns the data from the server as bytes. It is important to receive the data as bytes as receiving the data in another form will corrupt files that have binary data. First I start by making an outer while loop that is set to always true. The first line of this while loop calls a method I made called getLine to a temp String variable. getLine reads individual bytes from the BufferedInputStream and converts them to characters until it reaches a character '\n', signaling a new line has been reached. It puts all characters into one line thus being a line of the header sent by the server. The next bit of the while loop checks if this temp variable equals exactly to "\r\n" (the last line of the header). If it does the while loop breaks, if not we add the temp string to the end of the header string and continue looping.

```
String header = "";
String temp = "";
while(true) {
    temp = getLine(in);
    if (temp.equals("\r\n")) {
        break;
    }
    header = header + temp;
}
System.out.println(header);
```

If the header contains the code '200 Ok' we know the request was good and we open a fileOutputStream. Much like the last assignment we write to this file stream from the InputStream contents after the header, that has already been iterated over. The read() method will return something other than -1 until the end of the stream.

```
// create file to save
FileOutputStream file = new FileOutputStream(fileName);
// grab body data
byte buffer[] = new byte[1024];
int count;
while((count = in.read(buffer)) != -1){
    file.write(buffer, 0, count);
    file.flush();
}
file.close();
```

Another reason I use a BufferedInputStream and fileOutPut stream is so that I can write to a buffer and increase efficiency for larger files.