Web Client

Computer Network – Project #2

2018008904 이성진

2019-11-12

Web Client: Table of Contents

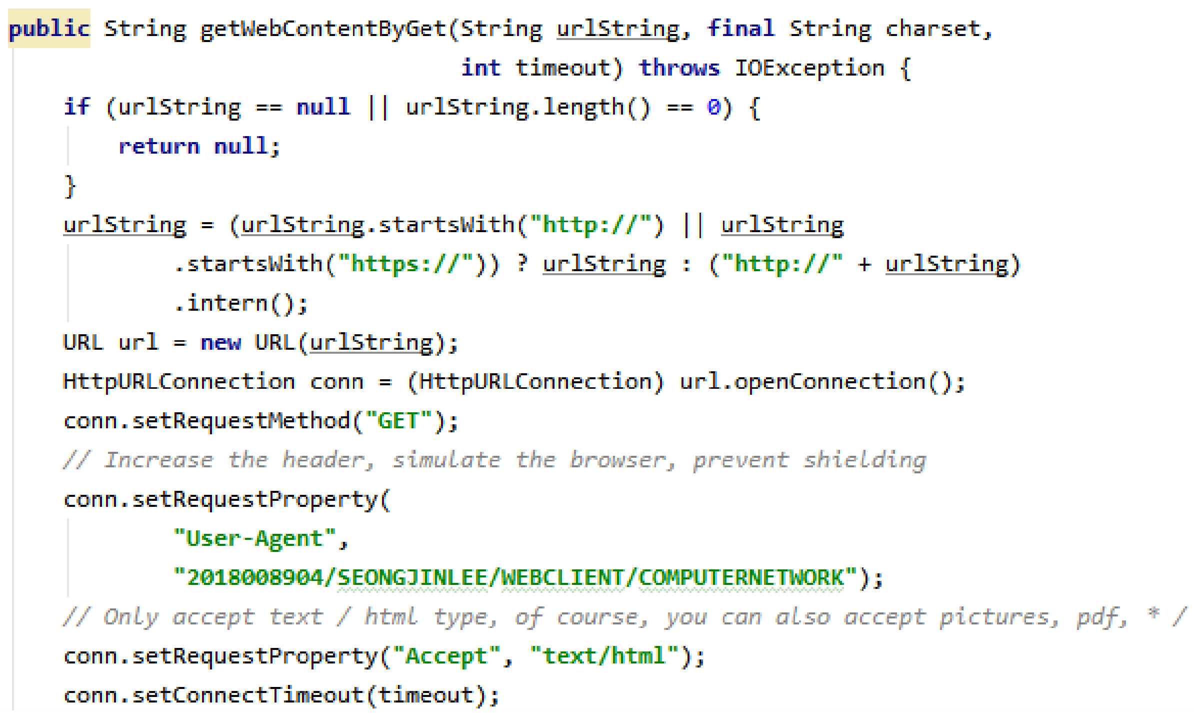
1. Overview

Project #2 uses the contents of the HTTP protocol we learned in class, just like Project #1. The project will implement a client that can send HTTP Get or Post Request to an HTTP server that meets the HTTP 1.1 standard.

2. Source Review

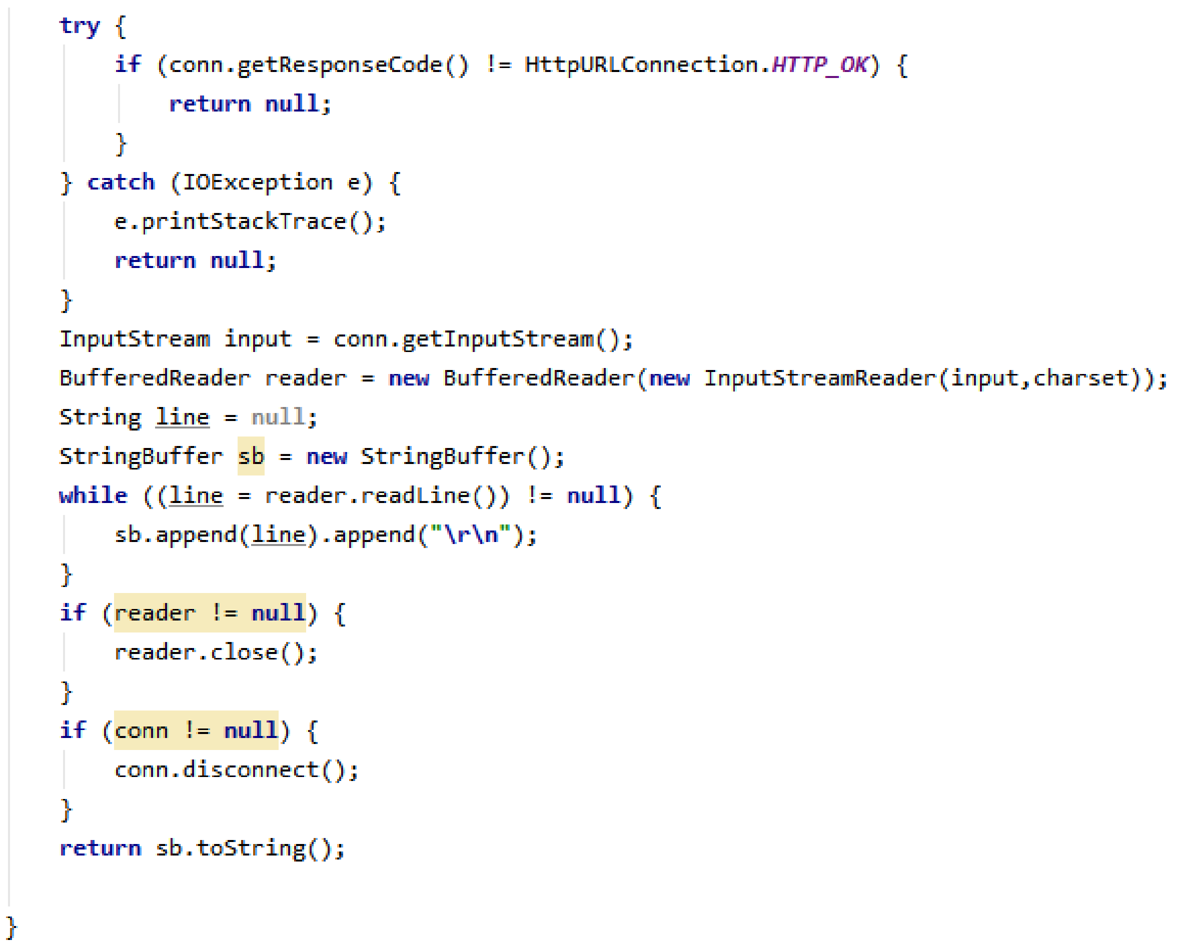
The newly created source code file for this project is one WebClient.java. This class contains methods such as main, getWebContentByGet, getWebContentByPost, printGet, printPost, and so on.

2.1. Get request (getWebContentByGet)



```java
public String getWebContentByGet(String urlString, final String charset,
                                 int timeout) throws IOException {
    if (urlString == null || urlString.length() == 0) {
        return null;
    }
    urlString = (urlString.startsWith("http://") || urlString
            .startsWith("https://")) ? urlString : ("http://" + urlString)
            .intern();
    URL url = new URL(urlString);
    HttpURLConnection conn = (HttpURLConnection) url.openConnection();
    conn.setRequestMethod("GET");
    // Increase the header, simulate the browser, prevent shielding
    conn.setRequestProperty(
            "User-Agent",
            "2018008904/SEONGJINLEE/WEBCLIENT/COMPUTERNETWORK");
    // Only accept text / html type, of course, you can also accept pictures, pdf, * /
    conn.setRequestProperty("Accept", "text/html");
    conn.setConnectTimeout(timeout);
```

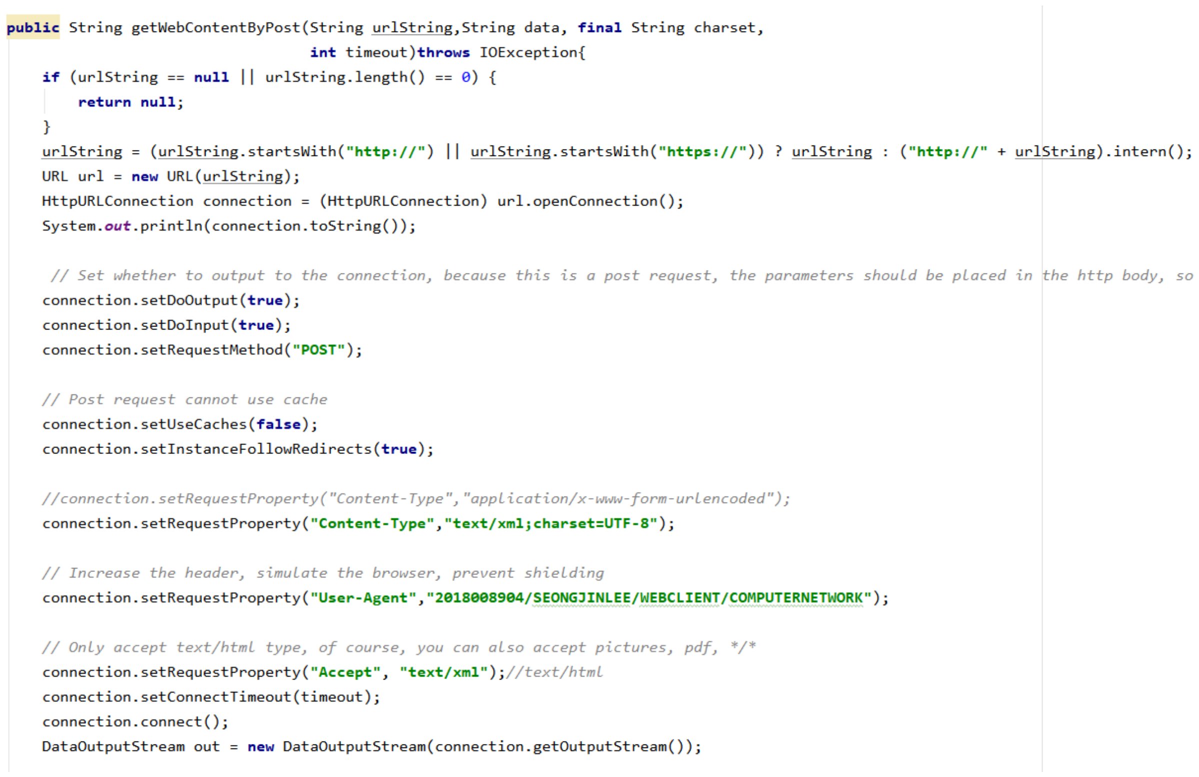
This function, which sends the Get request to the server and returns the response string received for it, first verifies that the entered url starts with http:// or https:// and corrects url if not in the appropriate form. Then, use the URL belonging to Java.net and class such as HttpURLConnection to attempt HTTP connection to the requested url. At this time, multiple header settings of HTTP connection can be done, and "GET" is used as a parameter of setRequestMethod method since the request will be sent. User-Agent was also set to the Student Number/Name/Project Name/Subject format as per the task requirements. In addition, Accept and timeout headers were also set using the internal method of HttpURLConnection.



```java
    try {
        if (conn.getResponseCode() != HttpURLConnection.HTTP_OK) {
            return null;
        }
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
    InputStream input = conn.getInputStream();
    BufferedReader reader = new BufferedReader(new InputStreamReader(input,charset));
    String line = null;
    StringBuffer sb = new StringBuffer();
    while ((line = reader.readLine()) != null) {
        sb.append(line).append("\r\n");
    }
    if (reader != null) {
        reader.close();
    }
    if (conn != null) {
        conn.disconnect();
    }
    return sb.toString();

}
```

When all settings are complete, call the getResponseCode method to verify that the normal response code (200 OK) has been sent from the server. Then, read the response message using InputStream and BufferedReader, save it in StringBuffer, and turn the message sent back to String.

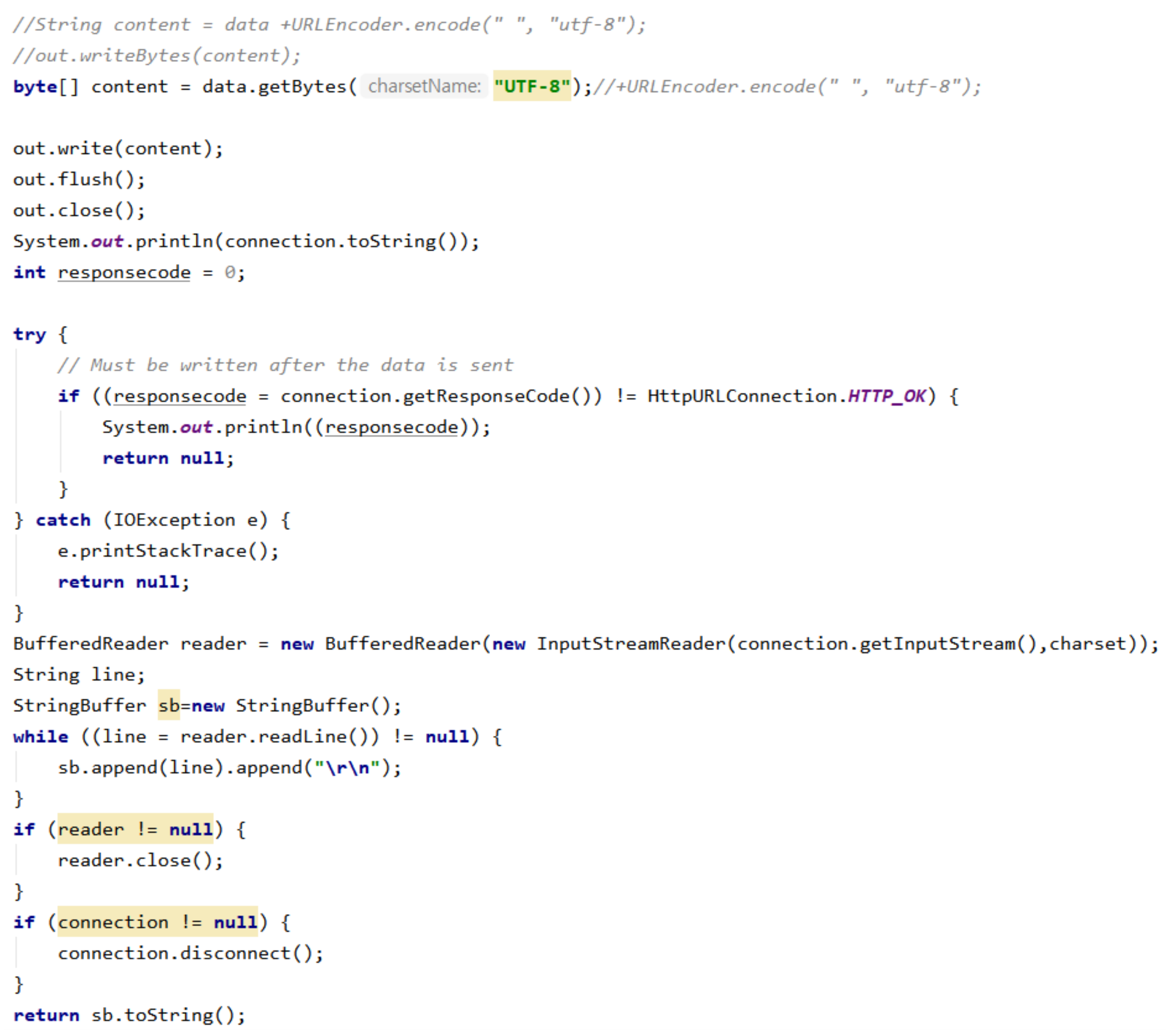
2.2. Post request (getWebContentByPost)



```java
public String getWebContentByPost(String urlString,String data, final String charset,
                                  int timeout)throws IOException{
    if (urlString == null || urlString.length() == 0) {
        return null;
    }
    urlString = (urlString.startsWith("http://") || urlString.startsWith("https://")) ? urlString : ("http://" + urlString).intern();
    URL url = new URL(urlString);
    HttpURLConnection connection = (HttpURLConnection) url.openConnection();
    System.out.println(connection.toString());

     // Set whether to output to the connection, because this is a post request, the parameters should be placed in the http body, so
    connection.setDoOutput(true);
    connection.setDoInput(true);
    connection.setRequestMethod("POST");

    // Post request cannot use cache
    connection.setUseCaches(false);
    connection.setInstanceFollowRedirects(true);

    //connection.setRequestProperty("Content-Type","application/x-www-form-urlencoded");
    connection.setRequestProperty("Content-Type","text/xml;charset=UTF-8");

    // Increase the header, simulate the browser, prevent shielding
    connection.setRequestProperty("User-Agent","2018008904/SEONGJINLEE/WEBCLIENT/COMPUTERNETWORK");

    // Only accept text/html type, of course, you can also accept pictures, pdf, */*
    connection.setRequestProperty("Accept", "text/xml");//text/html
    connection.setConnectTimeout(timeout);
    connection.connect();
    DataOutputStream out = new DataOutputStream(connection.getOutputStream());
```

The method getWebContentByPost created to send a Post request has many similar aspects to the previous getWebContentByGet. A slightly different point is that the Post request characteristic requires the user to send the data, so the string data is also required as a parameter. Set the request header for the requested url using HttpURLConnection similar to getWebContentByGet.

Then, declare one of the DataOutputStream guest houses using the connect method and post the entered data toward the requested url.

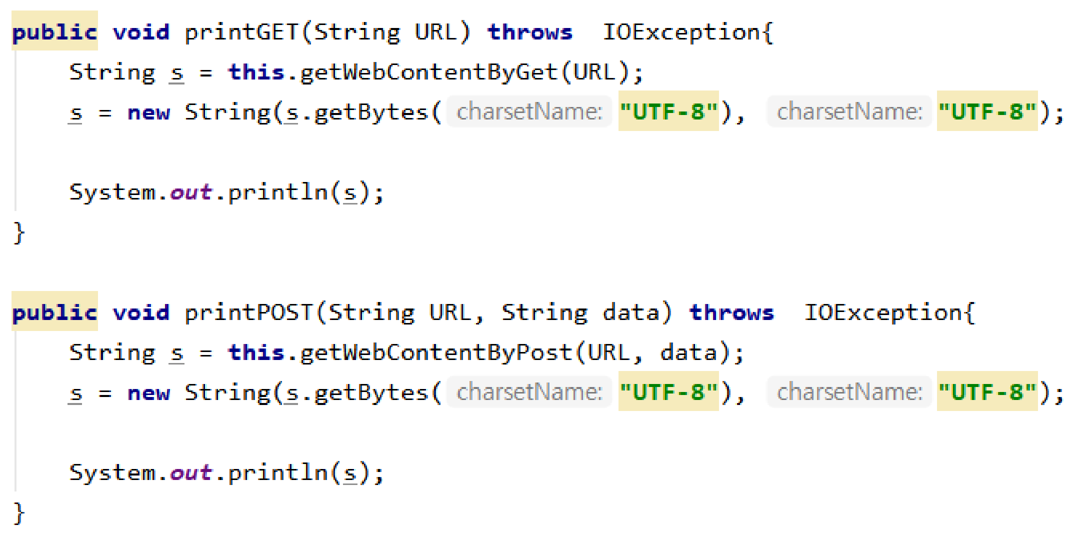


```java
//String content = data +URLEncoder.encode(" ", "utf-8");
//out.writeBytes(content);
byte[] content = data.getBytes( charsetName: "UTF-8");//+URLEncoder.encode(" ", "utf-8");

out.write(content);
out.flush();
out.close();
System.out.println(connection.toString());
int responsecode = 0;

try {
    // Must be written after the data is sent
    if ((responsecode = connection.getResponseCode()) != HttpURLConnection.HTTP_OK) {
        System.out.println((responsecode));
        return null;
    }
} catch (IOException e) {
    e.printStackTrace();
    return null;
}
BufferedReader reader = new BufferedReader(new InputStreamReader(connection.getInputStream(),charset));
String line;
StringBuffer sb=new StringBuffer();
while ((line = reader.readLine()) != null) {
    sb.append(line).append("\r\n");
}
if (reader != null) {
    reader.close();
}
if (connection != null) {
    connection.disconnect();
}
return sb.toString();
```

After then, check if the response code is 200 OK and the response message is delivered, and if confirmed, return message.

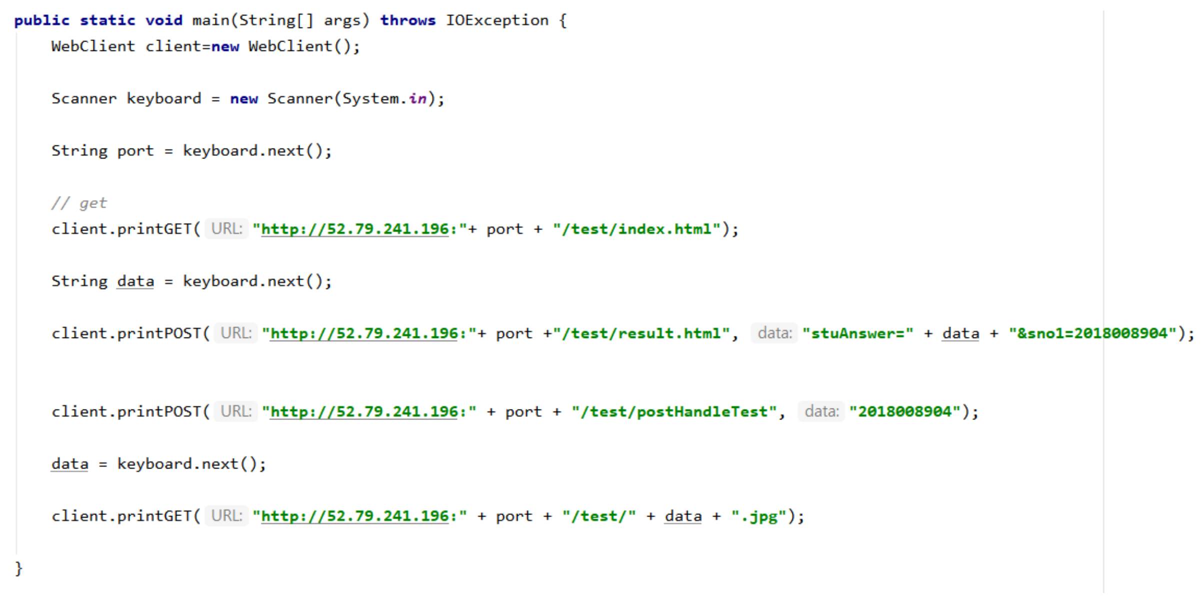
2.3. Print response message

The getWebContentMessage methods take a url and data as a paramater and return a response string. Therefore, in order for the user to see this, we need a function that prints a response message, so we defined the PrintGet and PrintPost functions. It's not a complicated function. It simply sends Get or Post using the received String and then returns a response by calling the System.out.println method when the response is received.



```java
public void printGET(String URL) throws  IOException{
    String s = this.getWebContentByGet(URL);
    s = new String(s.getBytes( charsetName: "UTF-8"),  charsetName: "UTF-8");

    System.out.println(s);
}

public void printPOST(String URL, String data) throws  IOException{
    String s = this.getWebContentByPost(URL, data);
    s = new String(s.getBytes( charsetName: "UTF-8"),  charsetName: "UTF-8");

    System.out.println(s);
}
```

And the main method is like the following. I hard-coded the context of requests required for the auto marking server test. Just enter the port and the correct answer, and it will automatically print out the required request.

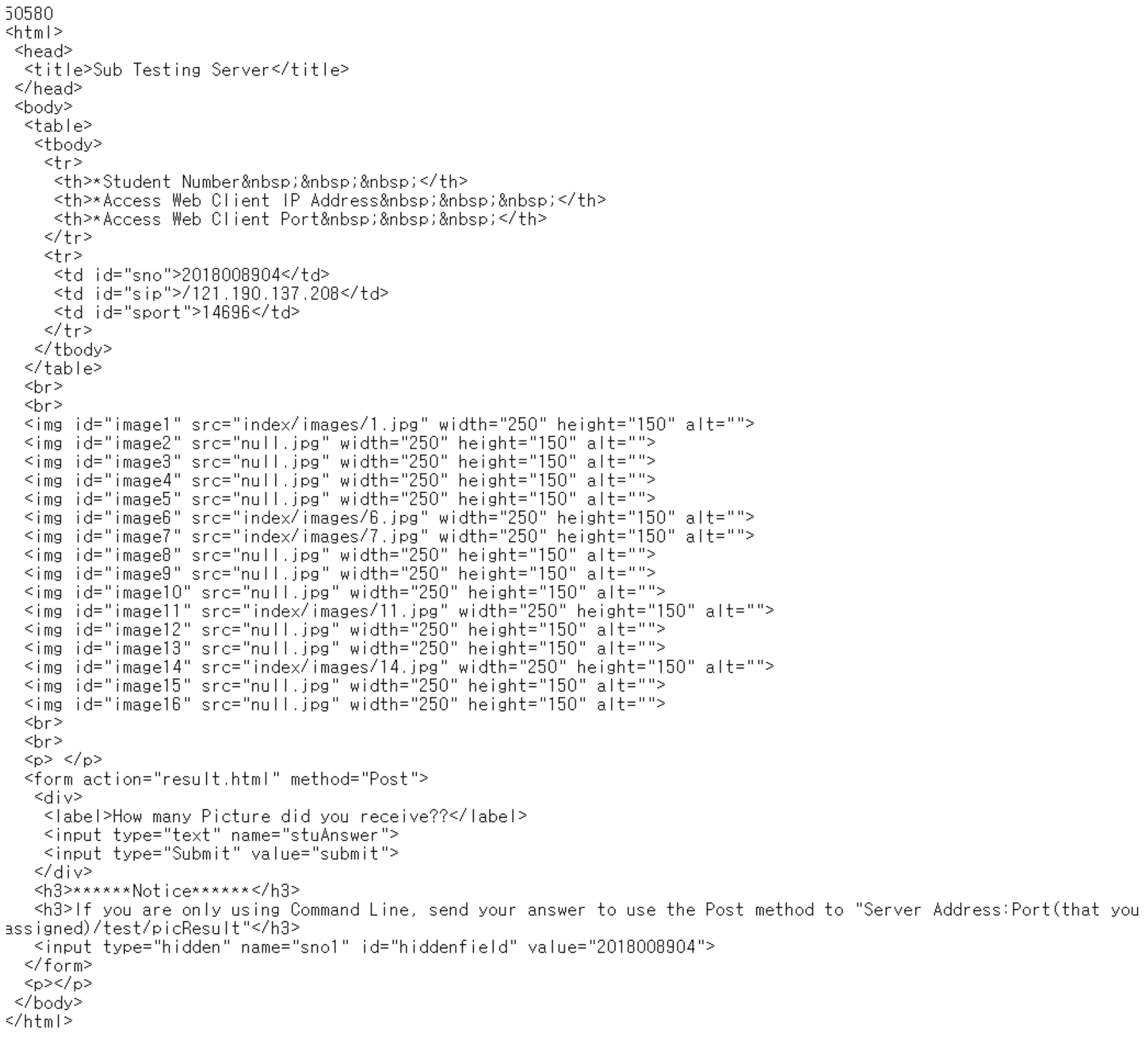


```java
public static void main(String[] args) throws IOException {
    WebClient client=new WebClient();

    Scanner keyboard = new Scanner(System.in);

    String port = keyboard.next();

    // get
    client.printGET( URL: "http://52.79.241.196:"+ port + "/test/index.html");

    String data = keyboard.next();

    client.printPOST( URL: "http://52.79.241.196:"+ port +"/test/result.html",  data: "stuAnswer=" + data + "&sno1=2018008904");


    client.printPOST( URL: "http://52.79.241.196:" + port + "/test/postHandleTest",  data: "2018008904");

    data = keyboard.next();

    client.printGET( URL: "http://52.79.241.196:" + port + "/test/" + data + ".jpg");

}
```

As above, the Get, Post requests are hard-coded to solve the problems that Auto marking server requires. For grading, it was designed to automatically send the required requests when you type in the number of ports and pictures that change each time after running the program.

3. Results

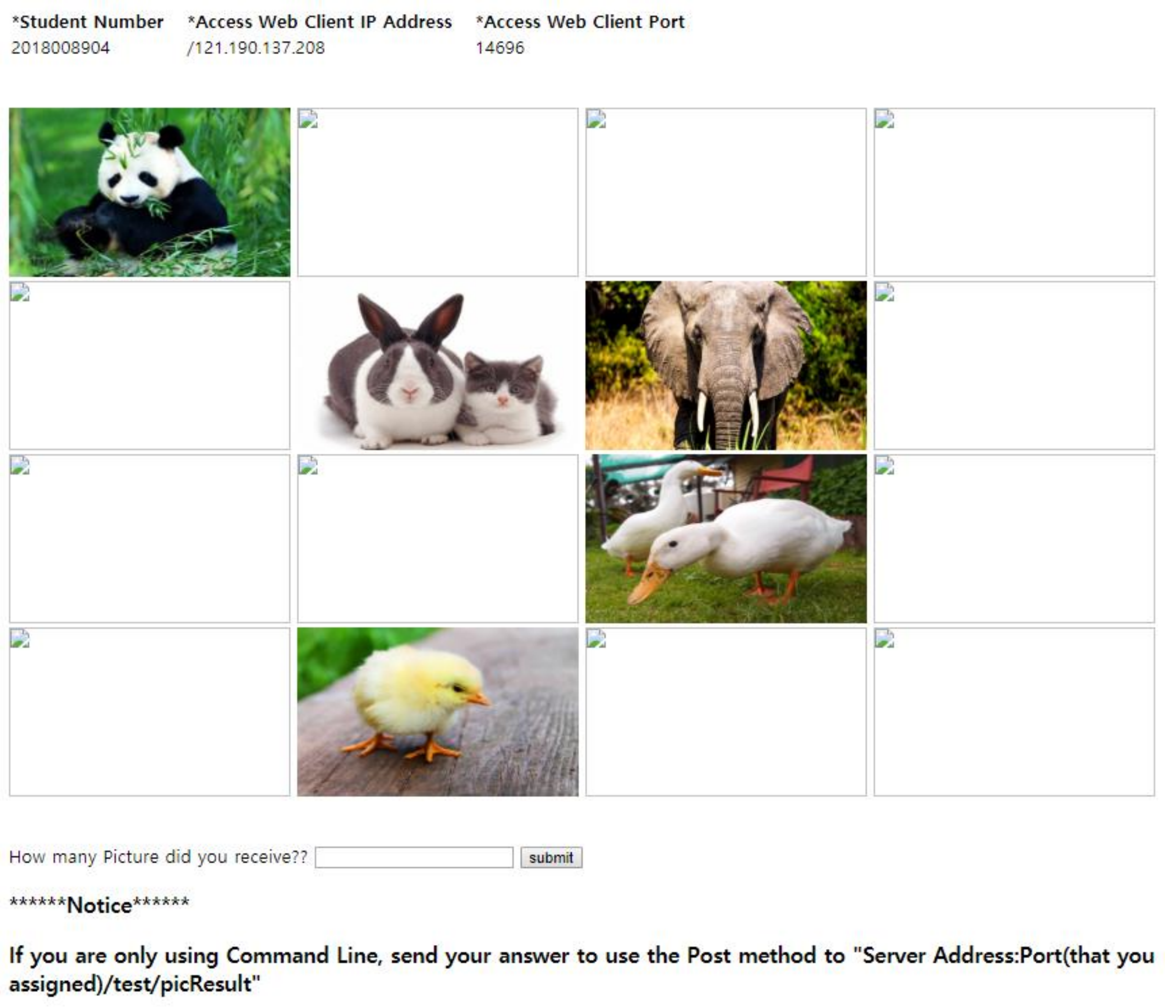
After sending the hard-coded requests and outputting the result, the following result was shown.



```
50580
<html>
 <head>
  <title>Sub Testing Server</title>
 </head>
 <body>
  <table>
   <tbody>
    <tr>
     <th>*Student Number&nbsp;&nbsp;&nbsp;</th>
     <th>*Access Web Client IP Address&nbsp;&nbsp;&nbsp;</th>
     <th>*Access Web Client Port&nbsp;&nbsp;&nbsp;</th>
    </tr>
    <tr>
     <td id="sno">2018008904</td>
     <td id="sip">/121.190.137.208</td>
     <td id="sport">14696</td>
    </tr>
   </tbody>
  </table>
  <br>
  <br>
  <img id="image1" src="index/images/1.jpg" width="250" height="150" alt="">
  <img id="image2" src="null.jpg" width="250" height="150" alt="">
  <img id="image3" src="null.jpg" width="250" height="150" alt="">
  <img id="image4" src="null.jpg" width="250" height="150" alt="">
  <img id="image5" src="null.jpg" width="250" height="150" alt="">
  <img id="image6" src="index/images/6.jpg" width="250" height="150" alt="">
  <img id="image7" src="index/images/7.jpg" width="250" height="150" alt="">
  <img id="image8" src="null.jpg" width="250" height="150" alt="">
  <img id="image9" src="null.jpg" width="250" height="150" alt="">
  <img id="image10" src="null.jpg" width="250" height="150" alt="">
  <img id="image11" src="index/images/11.jpg" width="250" height="150" alt="">
  <img id="image12" src="null.jpg" width="250" height="150" alt="">
  <img id="image13" src="null.jpg" width="250" height="150" alt="">
  <img id="image14" src="index/images/14.jpg" width="250" height="150" alt="">
  <img id="image15" src="null.jpg" width="250" height="150" alt="">
  <img id="image16" src="null.jpg" width="250" height="150" alt="">
  <br>
  <br>
  <p> </p>
  <form action="result.html" method="Post">
   <div>
    <label>How many Picture did you receive??</label>
    <input type="text" name="stuAnswer">
    <input type="Submit" value="submit">
   </div>
   <h3>******Notice******</h3>
   <h3>If you are only using Command Line, send your answer to use the Post method to "Server Address:Port(that you
assigned)/test/picResult"</h3>
   <input type="hidden" name="sno1" id="hiddenfield" value="2018008904">
  </form>
  <p></p>
 </body>
</html>
```

When I entered my student number, name, and IP address on the Auto marking server, I got a prompt to go to http://52.79.241.196:50580/test/index.html and check how many pictures were sent. So after running the client program, when I entered port number 50580, the response message was sent in the form of an html file. It was saved and checked through a web browser and found that:

Index.html



*Student Number	*Access Web Client IP Address	*Access Web Client Port
2018008904	/121.190.137.208	14696

How many Picture did you receive??

******Notice******

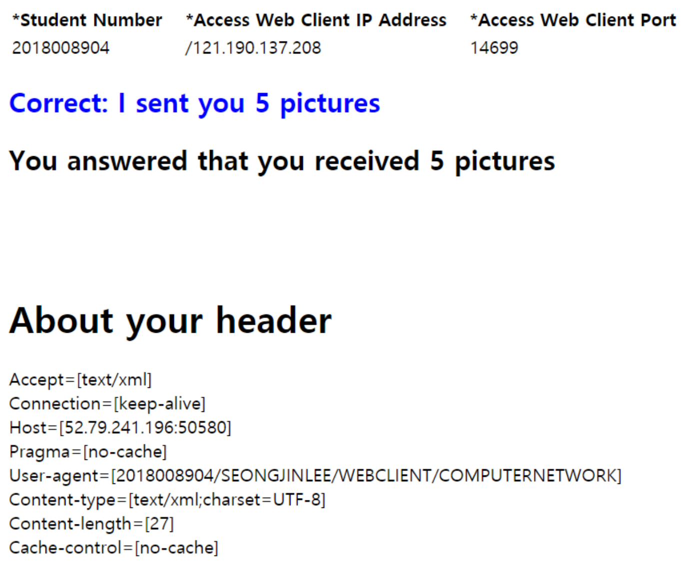
If you are only using Command Line, send your answer to use the Post method to "Server Address:Port(that you assigned)/test/picResult"

After analyzing the contents of Index.html, I was able to notice that the server sent me five images in response. Therefore, the correct answer 5 and the student number were sent using post request in the result.html.

Then the server sent the html code of the webpage in response, indicating that it answered my client correctly and set up the User-Agent header correctly. After passing Mission 1 and 2, I sent my student number to the server using post request at the address given for Mission 3. As a result, the server sent me a number 4033299763 and I could pass Mission 3 too by selecting the correct answer from the scoring page.

```
5
sun.net.www.protocol.http.HttpURLConnection:http://52.79.241.196:50580/test/result.html
sun.net.www.protocol.http.HttpURLConnection:http://52.79.241.196:50580/test/result.html
<html>
 <head>
 </head>
 <body>
  <table>
   <tbody>
    <tr>
     <th>*Student Number&nbsp;&nbsp;&nbsp;</th>
     <th>*Access Web Client IP Address&nbsp;&nbsp;&nbsp;</th>
     <th>*Access Web Client Port&nbsp;&nbsp;&nbsp;</th>
    </tr>
    <tr>
     <td id="sno">2018008904</td>
     <td id="sip">/121.190.137.208</td>
     <td id="sport">14699</td>
    </tr>
   </tbody>
  </table>
  <div>
   <h2 id="sentPic" style="color:blue">Correct: I sent you 5 pictures</h2>
  </div>
  <div>
   <h2 id="ansPic">You answered that you received 5 pictures</h2>
  </div>
  <br>
  <br>
  <br>
  <div>
   <h1>About your header</h1>
  </div>
  <div id="headerTest"> Accept=[text/xml]
   <br>Connection=[keep-alive]
   <br>Host=[52.79.241.196:50580]
   <br>Pragma=[no-cache]
   <br>User-agent=[2018008904/SEONGJINLEE/WEBCLIENT/COMPUTERNETWORK]
   <br>Content-type=[text/xml;charset=UTF-8]
   <br>Content-length=[27]
   <br>Cache-control=[no-cache]
   <br>
  </div>
  <div>
   <h2 id="warning"></h2>
  </div>
 </body>
</html>

sun.net.www.protocol.http.HttpURLConnection:http://52.79.241.196:50580/test/postHandleTest
sun.net.www.protocol.http.HttpURLConnection:http://52.79.241.196:50580/test/postHandleTest
4033299763
```

Result.html



*Student Number	*Access Web Client IP Address	*Access Web Client Port
2018008904	/121.190.137.208	14699

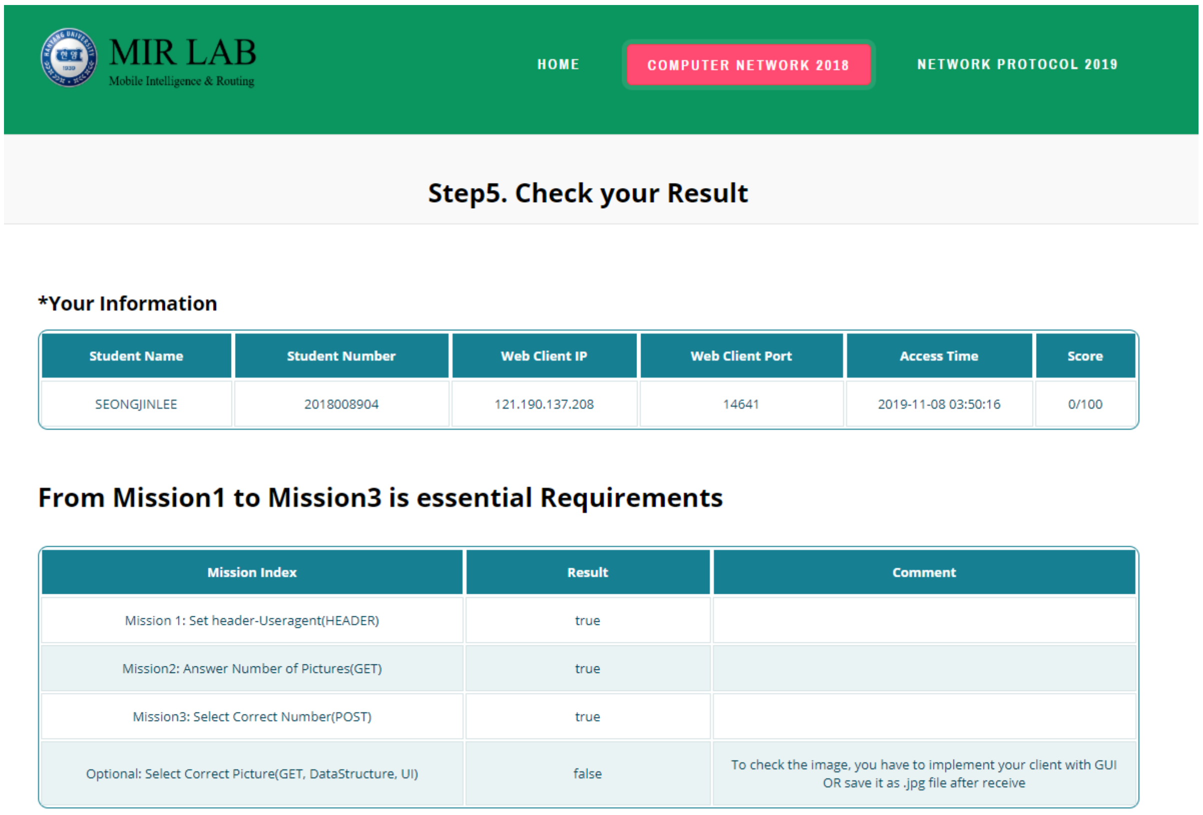
Correct: I sent you 5 pictures

You answered that you received 5 pictures

About your header

Accept=[text/xml]
Connection=[keep-alive]
Host=[52.79.241.196:50580]
Pragma=[no-cache]
User-agent=[2018008904/SEONGJINLEE/WEBCLIENT/COMPUTERNETWORK]
Content-type=[text/xml;charset=UTF-8]
Content-length=[27]
Cache-control=[no-cache]

The second automatic scoring program was also able to meet all goals through sending requests that satisfy the required requests, such as changing the User-Agent header. Scoring results are as follows.



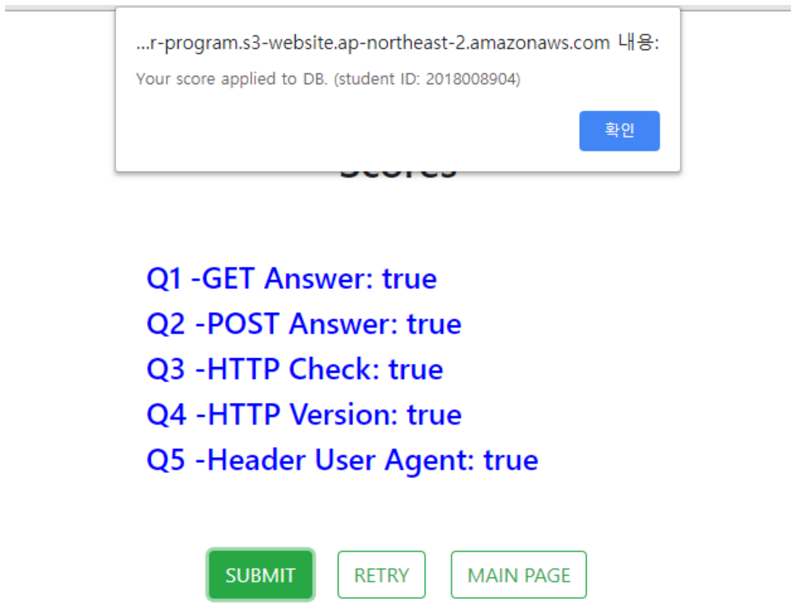
Step5. Check your Result

*Your Information

Student Name	Student Number	Web Client IP	Web Client Port	Access Time	Score
SEONGJINLEE	2018008904	121.190.137.208	14641	2019-11-08 03:50:16	0/100

From Mission1 to Mission3 is essential Requirements

Mission Index	Result	Comment
Mission 1: Set header-Useragent(HEADER)	true	
Mission2: Answer Number of Pictures(GET)	true	
Mission3: Select Correct Number(POST)	true	
Optional: Select Correct Picture(GET, DataStructure, UI)	false	To check the image, you have to implement your client with GUI OR save it as .jpg file after receive



...r-program.s3-website.ap-northeast-2.amazonaws.com 내용:

Your score applied to DB. (student ID: 2018008904)

Q1 -GET Answer: true
Q2 -POST Answer: true
Q3 -HTTP Check: true
Q4 -HTTP Version: true
Q5 -Header User Agent: true