Name: Rama Teja Repaka SUID:264100460

Internet Programming Assignment-2

----------NOTE----------

Source code file used

JHTTP.java --- Http server
RequestProcessor.java --- used by http server to handle requests
Client.java --- Client class to sent head and post request

---------- Execution Instructions----------

To compile and run JHTTP.java
javac JHTTP.java
java JHTTP . 8090 (. mean current directory)

To compile and run client
javac Client.java
java Client 8090

Question
Implement the HEAD method, by making the necessary changes in RequestProcessor.java.

Answer:

In Simple terms HEAD sends only headers as response to the client. It does not contain body. when request sent to JHTTP is HEAD then control goes to the below specified code. Then the below actions takes place in order

Get the file name from the Request Line like index.html etc
Then check for that file in the server local root directory
IF the file is found
then Read the file and find the size and send the http headers with status code 200 OK
Else if the file is not found
then send the HTTP headers with 404 File not found Error and plain html with status 404

This is the code responsible to handle HEAD requests in Requestprocessor class which in-turn will be used by JHTTP to handle HEAD requests

```
else if (method.equals("HEAD"))
 {

        /* if the method is head
         *  read the file name from the request and
         * check that file exists in the server
         * if exist then send Http 200 status code with headers
         * else send Http 400 error status code with headers
         *
         * */

          String fileName = tokens[1];
          if (fileName.endsWith("/")) fileName += indexFileName;
          String contentType =
              URLConnection.getFileNameMap().getContentTypeFor(fileName);
          if (tokens.length > 2) {
            version = tokens[2];
          }

          File theFile = new File(rootDirectory,
              fileName.substring(1, fileName.length()));

          if (theFile.canRead()
              // Don't let clients outside the document root
              && theFile.getCanonicalPath().startsWith(root)) {
            byte[] theData = Files.readAllBytes(theFile.toPath());
            if (version.startsWith("HTTP/")) { // send a MIME header
              sendHeader(out, "HTTP/1.0 200 OK", contentType, theData.length);
            }


          } else { // can't find the file
            String body = new StringBuilder("<HTML>\r\n")
                .append("<HEAD><TITLE>File Not Found</TITLE>\r\n")
                .append("</HEAD>\r\n")
                .append("<BODY>")
                .append("<H1> HTTP Error: File Not Found</H1>\r\n")
                .append("</BODY></HTML>\r\n").toString();
            if (version.startsWith("HTTP/")) { // send a MIME header
```

```
        //sending the header
        sendHeader(out, "HTTP/1.0 404 File Not Found",
            "text/html; charset=utf-8", body.length());
      }
      out.write(body);
      out.flush();
    }
  }
```

Create a simple Java client that sends a HEAD request to your revised JHTTP server (JHTTP.java & RequestProcessor.java) to test your implementation.

Created the below JAVA client using Http Url connection which sends HEAD request using the url http://127.0.0.1:8090/index.html. Then it displays the headers that are sent by the JHTTP which in-turn means RequestProcessor.java. I handled all the exceptions by using multiple catch blocks clearly as you can clearly see in the code. Below code handles cases for both successful request i mean 200 OK and as well as 404 file not found.

```
public static void headmethod()
        {
                String url = "http://127.0.0.1:8090/index.html";
                String USER_AGENT = "Mozilla/5.0";
                HttpURLConnection conn = null;
                try
                {
                        System.out.println("\nSending 'Head' request to URL : " + url);
                        URL obj = new URL(url);         //creating url object
                        conn = (HttpURLConnection) obj.openConnection();
                        conn.setReadTimeout(5000);  //setting read timeout
                        conn.setRequestMethod("HEAD"); // setting method as HEAD
                        conn.setRequestProperty("User-Agent", USER_AGENT);
                        conn.setDoInput(true);
        conn.setDoOutput(true);
        int responseCode=conn.getResponseCode(); // getting the response code
        System.out.println("\n Response Code: "+ conn.getResponseCode());
        System.out.println("\n Response Msg:  "+conn
                .getResponseMessage());
                        if (responseCode == HttpURLConnection.HTTP_OK)
                        {

                        //storing the headers in the map from the connection
```

```
Map<String, List<String>> map = conn.getHeaderFields();

 System.out.println("\nPrinting Response Header...\n");

for (Map.Entry<String, List<String>> entry : map.entrySet())
{
        System.out.println("Key : " + entry.getKey()
       + " ,Value : " + entry.getValue());
}

   //----------    code for printing the response-------------------

                        BufferedReader in = new BufferedReader(new
InputStreamReader(
                                    conn.getInputStream()));
                        String inputLine;
                        // ---------------------  variable for storing the response ------------------
                        StringBuffer response = new StringBuffer();
                        System.out.println("\n");
                        while ((inputLine = in.readLine()) != null)
                        {
                                response.append(inputLine);
                                response.append("\n");
                        }
                        in.close();
                        System.out.println(response.toString());
                        System.out.println("\nEnd of Head Response \n");
                }
                else
                {
                         //getting the headers and displaying them
                Map<String, List<String>> map = conn.getHeaderFields();

 System.out.println("\nPrinting Response Header...\n");

for (Map.Entry<String, List<String>> entry : map.entrySet())
{
        System.out.println("Key : " + entry.getKey()
       + " ,Value : " + entry.getValue());
}
                        System.out.println("\nBad HEAD request\n");
    }
```

```
        }
        catch (MalformedURLException e)
        {
                e.printStackTrace();
        }
        catch (IOException e)
        {
                e.printStackTrace();
        }
        finally
        {
    if (conn != null)
    {
        conn.disconnect(); //disconnecting the client
    }
        }}
```

Implement the POST method, by making the necessary changes in RequestProcessor.java

Clients send username and password parameters in request body for authentication purpose. In post method added a logic which retrieves the username and password from the request body and validates with the local database. (In this context i have created Map which holds pairs of usernames and passwords).

If the validation is successful then Requestprocessor sends the HTTP headers with status code 200 OK and finally request file contents.

Code used to handle 200 OK status as you can see clearly how the response is being sent.

```
if (version.startsWith("HTTP/")) { // send a MIME header
        sendHeader(out, "HTTP/1.0 200 OK", contentType, theData.length);
      }

    raw.write(theData);
    raw.flush();
```

If the validation is successful but not file that exists then Requestprocessor sends the HTTP headers with status code 404 File not found and HTML message file not found with status 404.

Code used to handle 404 FILE NOT FOUND as you can see clearly how the response is being sent.

```
else { // can't find the file
        String errormsg = new StringBuilder("<HTML>\r\n")
            .append("<HEAD><TITLE>File Not Found</TITLE>\r\n")
            .append("</HEAD>\r\n")
            .append("<BODY>")
            .append("<H1>HTTP Error 404: File Not Found</H1>\r\n")
            .append("</BODY></HTML>\r\n").toString();

        if (version.startsWith("HTTP/")) {
          // send a MIME header
          sendHeader(out, "HTTP/1.0 404 File Not Found",
              "text/html; charset=utf-8", errormsg.length());
        }
        out.write(errormsg);
        out.flush();
      }
```

If the validation is unsuccessful then the Requestprocessor sends the HTTP headers with status code 401 authorized and HTML message with unauthorized username or password message with the status 401.

Code used to handle INVALID username as you can see clearly how the response is being sent.

```
else
    {
      //case for handling Invalid username
     logger.info("Invalid Username");
     String errormsg = new StringBuilder("<HTML>\r\n")
     .append("<HEAD><TITLE></TITLE>\r\n")
     .append("</HEAD>\r\n")
     .append("<BODY>")
     .append("<H1>Invalid Username</H1>\r\n")
```

```
.append("</BODY></HTML>\r\n").toString();

// sending the header
sendHeader(out, "HTTP/1.0 401 Unauthorized",
  "text/html; charset=utf-8", errormsg.length());

// sending the content errormsg
out.write(errormsg);
out.flush();

}
```

Code used to handle INVALID password as you can see clearly how the response is being sent.

```
else
    {
     logger.info("Invalid password");
     String errormsg = new StringBuilder("<HTML>\r\n")
      .append("<HEAD><TITLE></TITLE>\r\n")
      .append("</HEAD>\r\n")
      .append("<BODY>")
      .append("<H1>Invalid Password</H1>\r\n")
      .append("</BODY></HTML>\r\n").toString();

    // send a MIME header
    sendHeader(out, "HTTP/1.0 401 Unauthorized",
        "text/html; charset=utf-8", errormsg.length());

   //sending the content errormsg
   out.write(errormsg);
   out.flush();

    }
```

Created a map which is a local database with usernames and passwords and initialized it in the constructor

It is clearly shown below how it is done in the code

```
HashMap<String, String> database = new HashMap<>();

  public RequestProcessor(File rootDirectory,
      String indexFileName, Socket connection)
  {

    database.put("Ram","syracuse");
    database.put("Ravi","syracuse");
    database.put("Venkatesh","syracuse”);
}
```

Below is code that is used for retrieving the Username and password from the Request body and Authenticating with the local database i.e Map.

```
while (true) {
        int c = in.read();
        request.append((char) c);
        if (c == '\r' || c == '\n')
          count++;
        else
          count = 0;
        if (count == 4)
          break;
      }


      Scanner scan = new Scanner(request.toString());
      scan.useDelimiter("Content-Length: ");
      scan.next();
      String str = scan.next();
      scan = new Scanner(str).useDelimiter("\r\n");
      int contentLength = scan.nextInt();
      scan.close();

// main code for retrieving the body that contains username and password

      String body = "";
      for (int i = 0; i < contentLength; i++)
        body += (char) in.read();

      logger.info("Body of the message : " + body);
```

```
String[] userdetails = body.split("[&]");

logger.info("parsing the body to retrieve username and password");

String username = userdetails[0].substring(userdetails[0].indexOf("=")+1);

String password = userdetails[1].substring(userdetails[1].indexOf("=")+1);

logger.info("The username  is "+ username);
logger.info("The password  is "+password);
logger.info("Validating the user with the local database");
username.replaceAll("\\s+","");
if(database.containsKey(username))
{
 String hashpass=database.get(username);
 if(hashpass.contains(password))
 {
   logger.info("Successfully authenticated");
```

Below is code for sending the requested file contents with headers and status 200 OK

```
String fileName = tokens[1];
        if (fileName.endsWith("/")) fileName += indexFileName;
        String contentType =
            URLConnection.getFileNameMap().getContentTypeFor(fileName);
        if (tokens.length > 2)
        {
          version = tokens[2];
        }

        File theFile = new File(rootDirectory,
            fileName.substring(1, fileName.length()));

        if (theFile.canRead()
            // Don't let clients outside the document root
            && theFile.getCanonicalPath().startsWith(root)) {
          byte[] theData = Files.readAllBytes(theFile.toPath());
          if (version.startsWith("HTTP/")) { // send a MIME header
            sendHeader(out, "HTTP/1.0 200 OK", contentType, theData.length);
          }
```

```
        raw.write(theData);
        raw.flush();
```

Below is the complete code for POST

```
else if (method.equals("POST"))
    {

      /* if the method is post
       * then parse the request to get the parameters in the body message
       * and validate the parameters from the local map database
       * if success then send the requested file with headers and status 200 Ok
       * if not send the Http 400 error status with message
       */

    StringBuilder request = new StringBuilder();
    int count = 0;
    while (true) {
      int c = in.read();
      request.append((char) c);
      if (c == '\r' || c == '\n')
        count++;
      else
        count = 0;
      if (count == 4)
        break;
    }
    Scanner scan = new Scanner(request.toString());
    scan.useDelimiter("Content-Length: ");
    scan.next();
    String str = scan.next();
    scan = new Scanner(str).useDelimiter("\r\n");
    int contentLength = scan.nextInt();
    scan.close();

    String body = "";
    for (int i = 0; i < contentLength; i++)
      body += (char) in.read();

    logger.info("Body of the message : " + body);

    //using split function to get the username and password from request  body
```

```
String[] userdetails = body.split("[&]");

logger.info("parsing the body to retrive username and password");

String username = userdetails[0].substring(userdetails[0].indexOf("=")+1);

String password = userdetails[1].substring(userdetails[1].indexOf("=")+1);

logger.info("The username  is "+ username);
logger.info("The password  is "+password);
logger.info("Validating the user with the local database");
username.replaceAll("\\s+","");
//condition for checking the valid username
if(database.containsKey(username))
{
 String hashpass=database.get(username);
 //condition for checking the valid password
 if(hashpass.contains(password))
 {
   logger.info("Successfully authenticated");
   String fileName = tokens[1];
       if (fileName.endsWith("/")) fileName += indexFileName;
       String contentType =
           URLConnection.getFileNameMap().getContentTypeFor(fileName);
       if (tokens.length > 2)
       {
         version = tokens[2];
       }

       File theFile = new File(rootDirectory,
           fileName.substring(1, fileName.length()));

       if (theFile.canRead()
           // Don't let clients outside the document root
           && theFile.getCanonicalPath().startsWith(root)) {
         byte[] theData = Files.readAllBytes(theFile.toPath());
         if (version.startsWith("HTTP/")) { // send a MIME header
           sendHeader(out, "HTTP/1.0 200 OK", contentType, theData.length);
         }

       //sending the content that is retrived
       raw.write(theData);
```

```
        raw.flush();
        }
        else { // can't find the file
            String errormsg = new StringBuilder("<HTML>\r\n")
                .append("<HEAD><TITLE>File Not Found</TITLE>\r\n")
                .append("</HEAD>\r\n")
                .append("<BODY>")
                .append("<H1>HTTP Error 404: File Not Found</H1>\r\n")
                .append("</BODY></HTML>\r\n").toString();

            if (version.startsWith("HTTP/")) {
              // send a MIME header
              sendHeader(out, "HTTP/1.0 404 File Not Found",
                  "text/html; charset=utf-8", errormsg.length());
            }
            //sending the message
            out.write(errormsg);
            out.flush();
        }
  }

  else
  {
   logger.info("Invalid password");
   String errormsg = new StringBuilder("<HTML>\r\n")
    .append("<HEAD><TITLE></TITLE>\r\n")
    .append("</HEAD>\r\n")
    .append("<BODY>")
    .append("<H1>Invalid Password</H1>\r\n")
    .append("</BODY></HTML>\r\n").toString();

  // send a MIME header
  sendHeader(out, "HTTP/1.0 401 Unauthorized",
      "text/html; charset=utf-8", errormsg.length());

 //sending the content errormsg
 out.write(errormsg);
 out.flush();

  }

 }
```

```
    else
    {
       //case for handling Invalid username
      logger.info("Invalid Username");
      String errormsg = new StringBuilder("<HTML>\r\n")
      .append("<HEAD><TITLE></TITLE>\r\n")
      .append("</HEAD>\r\n")
      .append("<BODY>")
      .append("<H1>Invalid Username</H1>\r\n")
      .append("</BODY></HTML>\r\n").toString();

      // sending the header
      sendHeader(out, "HTTP/1.0 401 Unauthorized",
        "text/html; charset=utf-8", errormsg.length());

      // sending the content errormsg
      out.write(errormsg);
      out.flush();

      }

   }
```

Thus it clearly satisfies the Requirements.

Create a simple Java client that sends a POST request to your revised JHTTP server (JHTTP.java & RequestProcessor.java) to test your implementation.

Created the below JAVA client using Http Url connection which sends POST request using the url http://127.0.0.1:8090/index.html. Then it displays the headers that are sent by the JHTTP which in-turn means RequestProcessor.java. I handled all the exceptions by using multiple catch blocks clearly as you can clearly see in the code. Below code handles cases like successful request i mean 200 OK , 404 file not found and finally 401 unauthorized(both username and password).

```
public static void postmethod()
        {
                String url = "http://127.0.0.1:8090/index1.html";
                 String USER_AGENT = "Mozilla/5.0";
                DataOutputStream wr;
                try {

                        URL obj = new URL(url);
```

```
                String urlParameters = "username=Ram&password=syracuse";
                HttpURLConnection con = (HttpURLConnection) obj.openConnection();
                // Send post request
                //Setting all the connection properties
                con.setDoOutput(true);
                con.setRequestMethod("POST");
                con.setRequestProperty("User-Agent", USER_AGENT);
                con.setRequestProperty("Accept-Language", "UTF-8");
                wr = new DataOutputStream(con.getOutputStream());
                wr.writeBytes(urlParameters);
            wr.flush();
            wr.close();
            System.out.println("\nSending 'POST' request to URL : " + url);
            System.out.println("\nPost parameters : " + urlParameters);
                int responseCode = con.getResponseCode();
System.out.println("\n Response Code: "+ con.getResponseCode());
System.out.println("\n Response Msg:  "+con
        .getResponseMessage());

                if (responseCode == HttpURLConnection.HTTP_OK)
            {

Map<String, List<String>> map = con.getHeaderFields();

     System.out.println("\nPrinting Response Header...\n");

    for (Map.Entry<String, List<String>> entry : map.entrySet())
    {
            System.out.println("Key : " + entry.getKey()
            + " ,Value : " + entry.getValue());
    }


                System.out.println("Response Code : " + responseCode);

                //code for printing the response

                BufferedReader in = new BufferedReader(
                        new InputStreamReader(con.getInputStream()));
                String inputLine;
                //variable for storing the response
                StringBuffer response = new StringBuffer();
                System.out.println("\n The contents of the file are: \n");
```

```
                while ((inputLine = in.readLine()) != null) {
                        response.append(inputLine);
                        response.append("\n");
                }
                in.close();
                //print response
                System.out.println(response.toString());
        }

        else
                {

            System.out.println("\nBad POST request\n");

            //storing the headers in the map from the connection

            Map<String, List<String>> map = con.getHeaderFields();

         System.out.println("\nPrinting Response Header...\n");

        for (Map.Entry<String, List<String>> entry : map.entrySet())
        {
                System.out.println("Key : " + entry.getKey()
                + " ,Value : " + entry.getValue());
        }
        System.out.println("\n\n");

        //code for getting the error stream

    Scanner scanner = new Scanner(con.getErrorStream());
    while(scanner.hasNext())
    System.out.println(scanner.next());
    scanner.close();
                }

        } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
        }
}
```

Testing Scenarios

Case 1 : correct filename and correct username and password in post method
Correct filename in head method

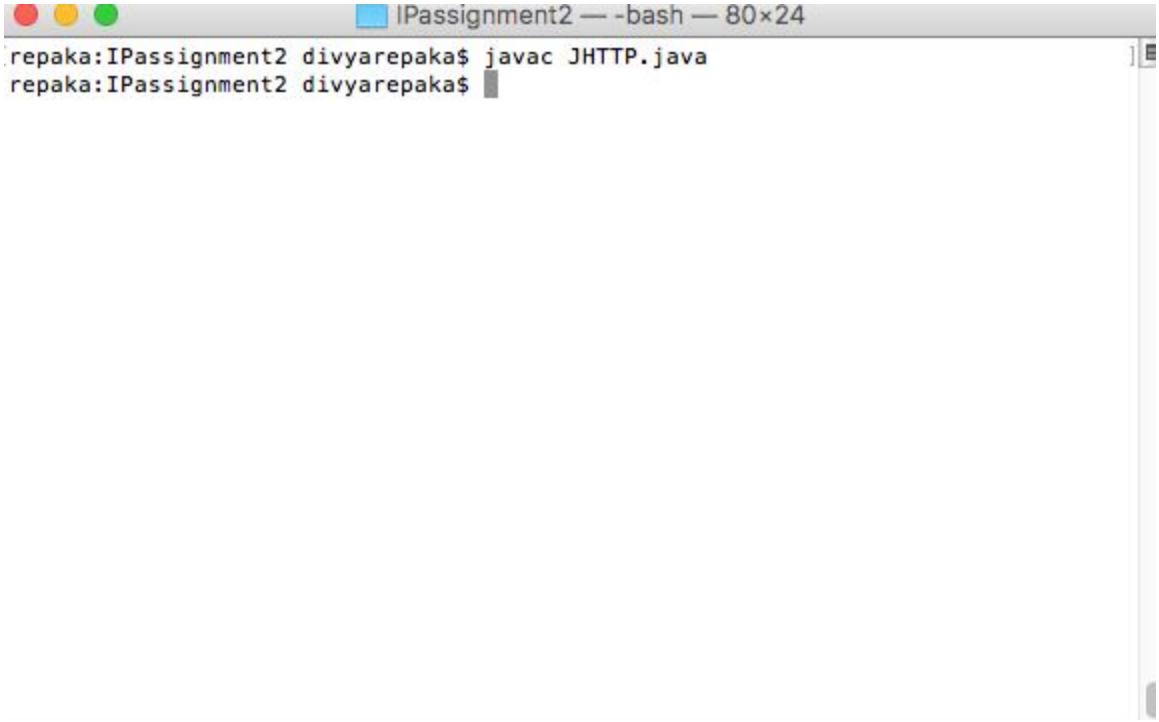
Case 2: In correct filename and correct username and password in post method
In Correct filename in head method

Case 3: In correct username and correct filename in post method

Case 4: In correct Password and correct filename in post method

OUTPUT SCREENSHOTS for all the above scenarios.

Compiled JHTTP.java



Run JHTTP with command line arguments as current directory and port number as shown below

```
IPassignment2 — java JHTTP . 8090 — 80×24
repaka:IPassignment2 divyarepaka$ java JHTTP . 8090
Mar 01, 2018 3:13:04 PM JHTTP start
INFO: Accepting connections on port 8090
Mar 01, 2018 3:13:04 PM JHTTP start
INFO: Document Root: .
```

Now the server is started and waiting for connections.
Compiled and executed Client.java as shown below

It also shows the output of Head request sent and Response it received

```
repaka:IPassignment2 divyarepaka$ javac Client.java
repaka:IPassignment2 divyarepaka$ java Client 8090

 --------------- Testing Head method -----------------

Sending 'Head' request to URL : http://127.0.0.1:8090/index.html

 Response Code: 200

 Response Msg:  OK

Printing Response Header...

Key : null ,Value : [HTTP/1.0 200 OK]
Key : Content-type ,Value : [text/html]
Key : Server ,Value : [JHTTP 2.0]
Key : Content-length ,Value : [123]
Key : Date ,Value : [Fri Mar 02 10:45:11 EST 2018]




End of Head Response


 --------------- End of Head Testing -----------------
```

It shows the output of POST request sent along with username and password and Response it received.

```
--------------- Testing Post method -----------------

Sending 'POST' request to URL : http://127.0.0.1:8090/index.html

Post parameters : username=Ram&password=syracuse

 Response Code: 200

 Response Msg:  OK

Printing Response Header...

Key : null ,Value : [HTTP/1.0 200 OK]
Key : Content-type ,Value : [text/html]
Key : Server ,Value : [JHTTP 2.0]
Key : Content-length ,Value : [123]
Key : Date ,Value : [Thu Mar 01 15:13:35 EST 2018]
Response Code : 200

 The contents of the file are:

<html>
<head>
</head>
<body>
<h1> This is the index page </h1>
<p> This is created for ip assignment </p>
</body>
</html>


 --------------- End of Post Testing -----------------
```

This is the server where you can see what are username and passwords retrieved from request and how they are validated

```
repaka:IPassignment2 divyarepaka$ javac JHTTP.java
repaka:IPassignment2 divyarepaka$ java JHTTP . 8090
Mar 01, 2018 10:42:22 PM JHTTP start
INFO: Accepting connections on port 8090
Mar 01, 2018 10:42:22 PM JHTTP start
INFO: Document Root: .
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: /127.0.0.1:49839 HEAD /index.html HTTP/1.1
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: /127.0.0.1:49840 POST /index.html HTTP/1.1
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: Body of the message : username=Ram&password=syracuse
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: parsing the body to retrive username and password
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: The username  is Ram
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: The password  is syracuse
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: Validating the user with the local database
Mar 01, 2018 10:43:06 PM RequestProcessor run
INFO: Successfully authenticated
```

Now sending a file request from the client side which does not exist in the server.

In the client change the URL to index1.html which does not exist
Then recompile the client and run again

Below screenshot shows the HEAD request , response received

```
repaka:IPassignment2 divyarepaka$ javac Client.java
repaka:IPassignment2 divyarepaka$ java Client 8090

 --------------- Testing Head method -----------------

Sending 'Head' request to URL : http://127.0.0.1:8090/index1.html

 Response Code: 404

 Response Msg:  File Not Found

Printing Response Header...

Key : null ,Value : [HTTP/1.0 404 File Not Found]
Key : Content-type ,Value : [text/html; charset=utf-8]
Key : Server ,Value : [JHTTP 2.0]
Key : Content-length ,Value : [114]
Key : Date ,Value : [Fri Mar 02 10:49:27 EST 2018]

Bad HEAD request
```

Below Screenshot shows the POST request sent along with url parameters and response received.

```
--------------- Testing Post method -----------------

Sending 'POST' request to URL : http://127.0.0.1:8090/index1.html

Post parameters : username=Ram&password=syracuse

 Response Code: 404

 Response Msg:  File Not Found

Bad POST request


Printing Response Header...

Key : null ,Value : [HTTP/1.0 404 File Not Found]
Key : Content-type ,Value : [text/html; charset=utf-8]
Key : Server ,Value : [JHTTP 2.0]
Key : Content-length ,Value : [117]
Key : Date ,Value : [Thu Mar 01 22:46:19 EST 2018]


<HTML>
<HEAD><TITLE>File
Not
Found</TITLE>
</HEAD>
<BODY><H1>HTTP
Error
404:
File
Not
Found</H1>
</BODY></HTML>

 --------------- End of Post Testing -----------------
```

Now sending a post Request with the invalid username

Changed the urlparameters variable in the code from Ram to RamS. Refer below screen shot

Code :

String urlParameters = "username=Rams&password=syracuse";

In the server side you can see validated the username as invalid

```
repaka:IPassignment2 divyarepaka$ java JHTTP . 8090
Mar 02, 2018 10:05:33 AM JHTTP start
INFO: Accepting connections on port 8090
Mar 02, 2018 10:05:33 AM JHTTP start
INFO: Document Root: .
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: /127.0.0.1:50762 HEAD /index.html HTTP/1.1
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: /127.0.0.1:50763 POST /index.html HTTP/1.1
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: Body of the message : username=Rams&password=syracuse
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: parsing the body to retrive username and password
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: The username  is Rams
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: The password  is syracuse
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: Validating the user with the local database
Mar 02, 2018 10:05:40 AM RequestProcessor run
INFO: Invalid Username
```

Output at the client side

```
--------------- Testing Post method -----------------

Sending 'POST' request to URL : http://127.0.0.1:8090/index.html

Post parameters : username=Rams&password=syracuse

 Response Code: 401

 Response Msg:  Unauthorized

Bad POST request


Printing Response Header...

Key : null ,Value : [HTTP/1.0 401 Unauthorized]
Key : Content-type ,Value : [text/html; charset=utf-8]
Key : Server ,Value : [JHTTP 2.0]
Key : Content-length ,Value : [89]
Key : Date ,Value : [Fri Mar 02 10:05:40 EST 2018]



<HTML>
<HEAD><TITLE></TITLE>
</HEAD>
<BODY><H1>Invalid
Username</H1>
</BODY></HTML>
```

Now sending a post Request with the invalid Password

Changed the urlparameters variable in the code from syracuse to syracuses. Refer below screen shot

Code :

String urlParameters = "username=Ram&password=syracuses";

In the server side you can see validated the password as invalid

```
IPassignment2 — java JHTTP . 8090 — 80×24
repaka:IPassignment2 divyarepaka$ javac JHTTP.java
repaka:IPassignment2 divyarepaka$ java JHTTP . 8090
Mar 02, 2018 10:10:50 AM JHTTP start
INFO: Accepting connections on port 8090
Mar 02, 2018 10:10:50 AM JHTTP start
INFO: Document Root: .
Mar 02, 2018 10:10:55 AM RequestProcessor run
INFO: /127.0.0.1:50772 HEAD /index.html HTTP/1.1
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: /127.0.0.1:50773 POST /index.html HTTP/1.1
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: Body of the message : username=Ram&password=syracuses
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: parsing the body to retrive username and password
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: The username  is Ram
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: The password  is syracuses
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: Validating the user with the local database
Mar 02, 2018 10:10:56 AM RequestProcessor run
INFO: Invalid password
```

Output at the client side

```
--------------- Testing Post method -----------------

Sending 'POST' request to URL : http://127.0.0.1:8090/index.html

Post parameters : username=Ram&password=syracuses

 Response Code: 401

 Response Msg:  Unauthorized

Bad POST request


Printing Response Header...

Key : null ,Value : [HTTP/1.0 401 Unauthorized]
Key : Content-type ,Value : [text/html; charset=utf-8]
Key : Server ,Value : [JHTTP 2.0]
Key : Content-length ,Value : [89]
Key : Date ,Value : [Fri Mar 02 10:10:56 EST 2018]



<HTML>
<HEAD><TITLE></TITLE>
</HEAD>
<BODY><H1>Invalid
Password</H1>
</BODY></HTML>

 --------------- End of Post Testing -----------------
```

Thus it clearly demonstrates all the requirements.