Practical 3 – Networking

Name: ZHAOHUI XU

Matriculation: 180019653

The practical work I have done had successfully passed all the basic tests from the stacscheck. The extensions that I have done will be shown below:

- **Returning of binary images (GIF, JPEG and PNG)**

To test the project whether it is successfully return, I have written relevant method that print out the result, for example: if the server have found the images inside the HTML file, It will print out the information such as “Content-Type: image/jpeg”

```
ConnectionHandler: GET /beer.jpg HTTP/1.1
ConnectionHandler: ... cleaning up and exiting ...
ConnectionHandler:run Stream closed
ConnectionHandler: ... cleaning up and exiting ...
HTTP/1.1 200 OK
Content-Type: image/jpeg
Content-Length: 30582


ConnectionHandler: GET /tp_it.jpg HTTP/1.1
ConnectionHandler: ... cleaning up and exiting ...
ConnectionHandler:run Stream closed
ConnectionHandler: ... cleaning up and exiting ...
```

Figure1: the result shown in console of exclipse

```
private byte[] getRecord(String directory, String filename) throws IOException, UnsupportedEncodingException   {
baOutput = null;
String path = directory + filename;
byte[] content = null;
String feedback = "";
try {
    //test whether file can be find
    br = new  BufferedReader(new FileReader(path));
    br.close();
    content = contentRecord(path);
    feedback += "HTTP/1.1 200 OK\r\n";
    if (filename.contains(".jpg")) {
        feedback += "Content-Type: image/jpeg\r\n";
    } else if (filename.contains(".gif")) {
        feedback += "Content-Type: image/gif\r\n";
    } else if (filename.contains(".png")) {
        feedback += "Content-Type: image/png\r\n";
    } else {
        feedback += "Content-Type: text/html\r\n";
    }
    feedback += "Content-Length: " + content.length + "\r\n";
    feedback += "\r\n";
    System.out.println(feedback);
    //log response
    bw.append(date + " " + feedback);
    bw.newLine();
    baOutput  = new ByteArrayOutputStream();
    baOutput.write(feedback.getBytes("UTF-8"));
    baOutput.write(content);
```

Figure2: the method how to check the image type

- **Multithreading – support multiple concurrent client connection requests up to a specified limit**

In this part, I have used a class “ConnectionCount” to handle the count problem of multi thread, and therefore if the class “Server” has listened the request from the client, it will automatically add the count in “ConnectionCount” class. Similarly, if one client exists, the class will detect and upload the count.

```
main [Java Application] /Library/Java/JavaVirtualMachine
Server started ... listening on port 12345 ..
Server got new connection request from /0:0:0
Number of connection 1
Server got new connection request from /0:0:0
new ConnectionHandler thread started ....
Number of connection 2
new ConnectionHandler thread started ....
HTTP/1.1 200 OK
Content-Type: text/html
Content-Length: 805



ConnectionHandler: GET /page2.html HTTP/1.1
ConnectionHandler: ... cleaning up and exitin
```

Figure3: the result shown in console of exclipse

```
public ServerTest(String directory, int port) {
try {
ss = new ServerSocket(port);
conCounter = new ConnectionCount();
System.out.println("Server started ... listening on port "
//get number of connections
int counter = conCounter.getCounter();
while (counter < maxConnections) {
// will wait until client requests a connection, then retu
Socket conn = ss.accept();
conCounter.add();
```

Figure4: the method how to process the method, “maxConnections” is static number which is 30. if the client is above 30, the client will not be alllowed to access

- **Logging – each time requests are made, log them to a file, indicating date/time request type**

In this part, the method of log response is within the class of “ConnectionHandler”, it will print out the result including date in any situation to log file.

```
25 private OutputStream os; // can send data back to the client on this output stream
26 private ByteArrayOutputStream baOutput;   // The data sent to the output stream is saved in th
27 private Date date = new Date();
28 private BufferedReader br;         // use buffered reader to read client data
29 private BufferedWriter bw;           // use buffered writer to write client data
30 /**
```

Figure5: Initialization

```
113     feedback += "Content-Length: " + content.length + "\r\n";
114     feedback += "\r\n";
115     System.out.println(feedback);
116     //log response
117     bw.append(date + " " + feedback);
118     bw.newLine();
```

```
166     //log response
167     bw.append(date + " " + feedback);
168     bw.newLine();
169     baOutput  = new ByteArrayOutputStream();
170     baOutput.write(feedback.getBytes("UTF-8"));
171     baOutput.write(content);
```

```
179     //log response
180     bw.append(date + " " + feedback);
181     bw.newLine();
```

Figure6, 7, 8: add the log response in GET/HEAD/No Implemented method