NAME: TANVEER AHMED SHAIK

STUDENT ID: 1001704423

Requirements for running the assignment: Eclipse ide, jdk1.8.0_191 or above.

Instructions for executing the assignment:

1. Unzip the Shaik_txs4423.zip file.
2. It will have 2 projects with names Server and Clients.
3. Import both the Server and Clients in the Eclipse ide. (You may see complier errors because of missing jars)
4. Then right lick on Server project
   Step 1: Go to properties
   Step 2: Click on Java Build Path
   Step 3: Click on Libraries.
   Step 4: If you see jars are missing (click on those jars and remove them first) then click on Add External Jars.
   Step 4: Browse the unzipped Shaik_txs4423 folder and go to Jars folder.
   Step 5: Select all the jars. Then click on apply and then Ok and close.
   (Note: Without adding jars to the Java Build Path Server project ,code cannot be complied)
5. Repeat the 4th for Clients Project as well. (Note: Without adding jars to the Java Build Path Clients project, code cannot be complied)
6. In Server project go to src/com/assignment and run ServerUi.java as java application.
   a. Server GUI will pop up. Click on Start Server. Then the server will start at "localhost: 8080".
      (Note: There should not be any process running on port 8080 on your machine. Otherwise Address already in use exception will occur.)
   b. After starting the server, a server started message will be appear on Server's GUI.
   c. Additionally, a stop server will also be available for stopping the server and exiting out.
7. In Clients project go to src/com/assignment and run ClientUi.java as java application.
   a. Client Creator GUI will pop up. You will have 3 createclient buttons to create clients. On clicking of create client button a new client GUI will pop up. (All the clients may overlay on each other. So, drag each client separately on the vacant space on your screen)
      (Note: Once a client is created it cannot be recreated until its destroyed. And, a maximum of 3 clients can be created at any point of time.)
   b. Each client will have a "Connect to Server", "Disconnect from Server" buttons to connect and disconnect from server respectively. Client GUI will have a text areas to show messages received and sent.

Server:

1. Server's UI will show all the incoming request to the server in HTTP message format.
2. Server's UI will also show the status of the users i.e. connected or disconnected in real time.
3. Server will process the incoming HTTP request and process it and responds back with appropriate HTTP response.
4. Server will also send the messages from clients to the corresponding recipients in real time.

Client:

1. When the client is created it will randomly get its local time between 2 to 50. And you can see that on the client's UI screen. Now each client randomly picks one client and sends its local time.
2. Sender's UI will be updated with "User i -> User j: local time X" along with, If the message is delivered to the receiver "*Server: Success*" else *"Server: Message not send as the receiver is unavailable"* depending on the response from server i.e. 200 Ok or 404 Not Found.
3. The time gap between sending messages will be 2 to 20 seconds.
4. Recipients client UI will be updated with the incoming messages, first with "*Server: Success*"and then with the sender's name and message and the either with "No adjustment necessary" or "Adjustment necessary: local time adjusted from Y to X+1" depending upon the remote clock and local clock's time as per Lamport's algorithm.

Decisions taken and assumptions:

1. The User Name of the clients are fixed i.e. for Client 1 it is User 1, Client 2 it is User 2 and Client 3 it is User 3. Because we never know what username a user gives each time to clients. So, the usernames are fixed to ease the communication via server.

Crashing situations of server and clients:

1. When the Client closes or crashes and Client creator closes or crashes, before closing or crashing if the client is connected to the server then the client immediately disconnects with the server. Clients crashing does not impact server.
2. When with server closes or crashes it does not impact the clients. When client try to send request to the server. The client's UI will be updated with server unavailable.

Few additional things:

1. In Server UI ,you can see full HTTP messages. First portion will be HTTP request from the client then an additional line i.e. *(Sending message to recipient ....)* and then the HTTP message that is sent to the recipients and then one more additional line i.e. *(Message successfully delivered to recipient....)*. There after HTTP response sent back to the client which sent message to the server.

2. If a client is connected to server and server is closed or crashed and then if the client tries to send a message, then client's UI will be updated with "Server unavailable" message. (please reconnect to the server by clicking connect to server button)
3. If the client is disconnected using the disconnected button in the client UI then local clock of the client will still be running, it will not be stopped.
4. If a client is closed or crashed, then local clock of the client is stopped and when the client is recreated then the local clock is again reinitiated.
5. If a client is connected to server and server is closed or crashed and started again then if the client tries to send a message, then client may still receive "Server unavailable" message and changes its states to normal in UI i.e. Disconnect from server button is disabled and Connect to server button will be enabled (please reconnect to the server by clicking connect to server button). So, it's better to disconnect and connect back again.
6. If all the clients are closed or crashed at once. Then the server UI may show HTTP messages in its own order of each message received this is because concurrent threads will be running by the clients to track the crash events.

References:

1. https://www.dreamincode.net/forums/topic/189336-socket-post-request/
2. http://tutorials.jenkov.com/java-multithreaded-servers/multithreaded-server.html
3. https://www.tutorialspoint.com/http/http_responses.htm
4. https://www.tutorialspoint.com/http/http_requests.htm