NETWORK ARCHITECTURE -I

FINAL PROJECT REPORT

ON

CHAT APPLICATION

Niharika Sri Sai Ambati(16247289),

Sindhu Mudireddy(16242444),

Sriram Mentey(16251319),

Divya Neelima Putta(16233323).

Project Goal: Our primary aim was to create a chat application between a client and a server using socket programming.

Introduction:

Implementing a chat server application provides a good opportunity for a beginner to design and implement a client server based system. The design is very simple. It is implemented in Java, since it is easy to program in, it precludes the need to deal with low level memory management and includes powerful libraries for sockets and threads.

Scope:

It can be used for private chat over LAN and over Internet. It can be also used r broadcasting notices in an organization. No dedicated server is required if the application is used for Local Area Network(LAN) i.e. any PC can become server.

Tools used:

Eclipse

Languages used:

Java

Methodology/Design:

This Chat Application is based on Client-Server model, where client request from the server and server give response. TCP is used as the transport layer protocol, since it provides reliable delivery which is critical for the given application. TCP does not provide timing guarantee, which is not very important in the given scenario. This protocol ensures that data sent over two points in a network is received in the same order as it is sent. We have created the chat application between a client and the server using the socket programming.

Socket Programming:

Socket:

A socket is the one end-point of a two-way communication link between two programs running over the network. However, one can run the two programs on the same computer. Such communicating programs constitutes a client/server application. The server implements a dedicated logic, called service.

Creating a network connection:

The network connection is initiated by a client program. When it creates a socket for the communication with the server. To create the socket in Java, the client calls the Socket constructor and passes the server address and the specific server port number to it. At this stage, the server must be started on the machine having the specified address and listening for connections on its specific port number. The server uses a specific port dedicated only to listening for connection requests from clients. It cannot use this specific port for data communication with the clients because the server must be able to accept the client connection at any instant. So, its specific port has be dedicated only to listening for new connection requests. The server-side socket associated with specific port is called as server socket. When a connection request arrives on this socket from the client side, the client and the server establish a connection. This connection can be established as follows:

1. When the server receives a connection request on its specific server port, it creates a new socket for it and binds a port number to it.
2. It sends the new port number to the client to inform it that the connection is established.
3. The server goes on now by listening on two ports:

- it waits for new incoming connection requests on its specific port, and
- it reads and writes messages on established connection (on new port) with the accepted client.

The server communicates with the client by reading from and writing to the new port. If other connection requests arrive, the server accepts them in the similar way creating a new port for each new connection. Thus, at any instant, the server must be able to communicate simultaneously with many clients and to wait on the same time for incoming requests on its specific server port. The

communication with each client is done via the sockets created for each communication.

```
SERVER                                 CLIENT
    ,--------------.                      ,--------------.
    |            ,-|                      |              |
    |   accept   | |<---------------------|   create     |
    |            `-|                      |              |
    |            ,-|                      |-.            |
    | read/write | | <------------------->| | read/write |
    |            `-|                      |-'            |
    `--------------'                      `--------------'
```

Chat Application Code in Java:

Client:

```
package server;

import java.io.*;

import java.net.Socket;

import java.net.SocketException;

import java.net.UnknownHostException;

public class ChatSocketClient {

private Socket socket = null;

private InputStream inStream = null;

private OutputStream outStream = null;

public ChatSocketClient() {

}

public void createSocket() {

try {

socket = new Socket("localHost", 3337);

System.out.println("Connected");

inStream = socket.getInputStream();

outStream = socket.getOutputStream();

createReadThread();

createWriteThread();

} catch (UnknownHostException u) {

u.printStackTrace();
```

```
} catch (IOException io) {
io.printStackTrace();
}
}
public void createReadThread() {
Thread readThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
byte[] readBuffer = new byte[200];
int num = inStream.read(readBuffer);
if (num > 0) {
byte[] arrayBytes = new byte[num];
System.arraycopy(readBuffer, 0, arrayBytes, 0, num);
String recvedMessage = new String(arrayBytes, "UTF-8");
System.out.println("Received message from client:" + recvedMessage);
}/* else {
// notify();
}*/
;
//System.arraycopy();
}catch (SocketException se){
System.exit(0);
```

```
} catch (IOException i) {

i.printStackTrace();

}

}

}

};

readThread.setPriority(Thread.MAX_PRIORITY);

readThread.start();

}

public void createWriteThread() {

Thread writeThread = new Thread() {

public void run() {

while (socket.isConnected()) {

try {

BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in));

sleep(100);

String typedMessage = inputReader.readLine();

if (typedMessage != null && typedMessage.length() > 0) {

synchronized (socket) {

outStream.write(typedMessage.getBytes("UTF-8"));

sleep(100);

}

}
```

```
;
//System.arraycopy();
} catch (IOException i) {
i.printStackTrace();
} catch (InterruptedException ie) {
ie.printStackTrace();
}
}
}
};
writeThread.setPriority(Thread.MAX_PRIORITY);
writeThread.start();
}
public static void main(String[] args) throws Exception {
ChatSocketClient myChatClient = new ChatSocketClient();
myChatClient.createSocket();
/*myChatClient.createReadThread();
myChatClient.createWriteThread();*/
}
}
```

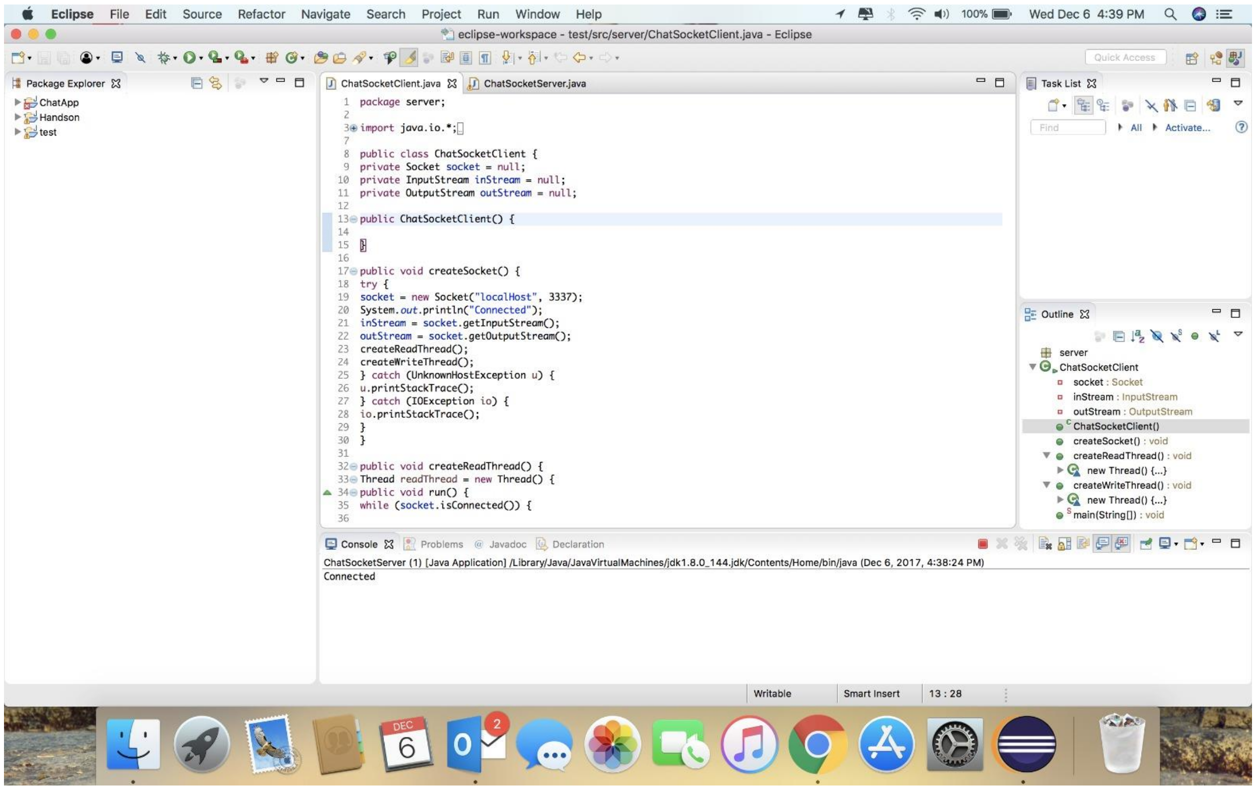
Server Connections:

```
package server;

import java.io.*;

import java.net.ServerSocket;

import java.net.Socket;

import java.net.SocketException;

public class ChatSocketServer {

private ServerSocket severSocket = null;

private Socket socket = null;

private InputStream inStream = null;

private OutputStream outStream = null;


public ChatSocketServer() {

}

public void createSocket() {

try {

ServerSocket serverSocket = new ServerSocket(3337);

while (true) {

socket = serverSocket.accept();

inStream = socket.getInputStream();

outStream = socket.getOutputStream();

System.out.println("Connected");

createReadThread();
```

```
createWriteThread();

}

} catch (IOException io) {

io.printStackTrace();

}

}

public void createReadThread() {

Thread readThread = new Thread() {

public void run() {

while (socket.isConnected()) {

try {

byte[] readBuffer = new byte[200];

int num = inStream.read(readBuffer);

if (num > 0) {

byte[] arrayBytes = new byte[num];

System.arraycopy(readBuffer, 0, arrayBytes, 0, num);

String recvedMessage = new String(arrayBytes, "UTF-8");

System.out.println("Received message from server:" + recvedMessage);

} else {

notify();

}

;

//System.arraycopy();
```

```
} catch (SocketException se) {
System.exit(0);
} catch (IOException i) {
i.printStackTrace();
}
}
}
};
readThread.setPriority(Thread.MAX_PRIORITY);
readThread.start();
}

public void createWriteThread() {
Thread writeThread = new Thread() {
public void run() {
while (socket.isConnected()) {
try {
BufferedReader inputReader = new BufferedReader(new InputStreamReader(System.in));
sleep(100);
String typedMessage = inputReader.readLine();
if (typedMessage != null && typedMessage.length() > 0) {
synchronized (socket) {
outStream.write(typedMessage.getBytes("UTF-8"));
sleep(100);

}

}/* else {

notify();

}*/

;

//System.arraycopy();

} catch (IOException i) {

i.printStackTrace();

} catch (InterruptedException ie) {

ie.printStackTrace();

}


}

}

};

writeThread.setPriority(Thread.MAX_PRIORITY);

writeThread.start();

}

public static void main(String[] args) {

ChatSocketServer chatServer = new ChatSocketServer();

chatServer.createSocket();

}}
```

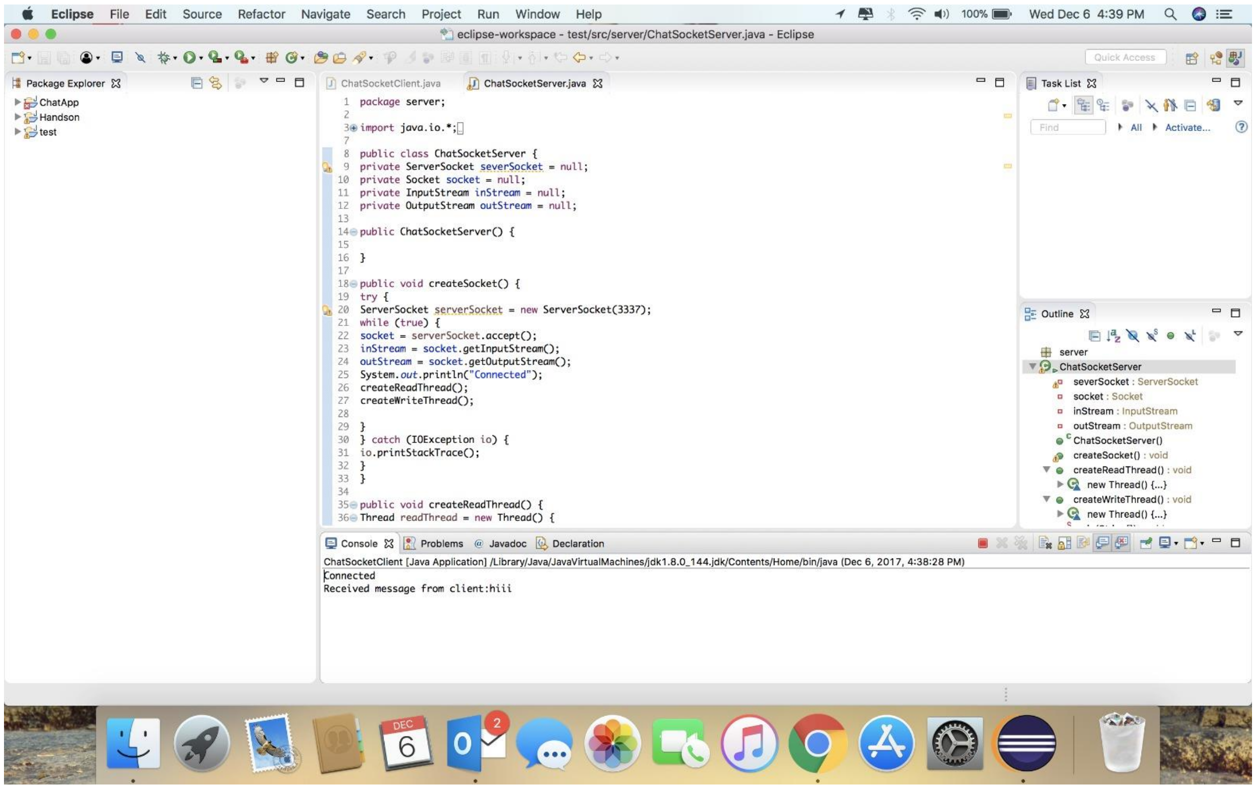
<u>**Output Screenshots:**</u>



eclipse-workspace - test/src/server/ChatSocketClient.java - Eclipse

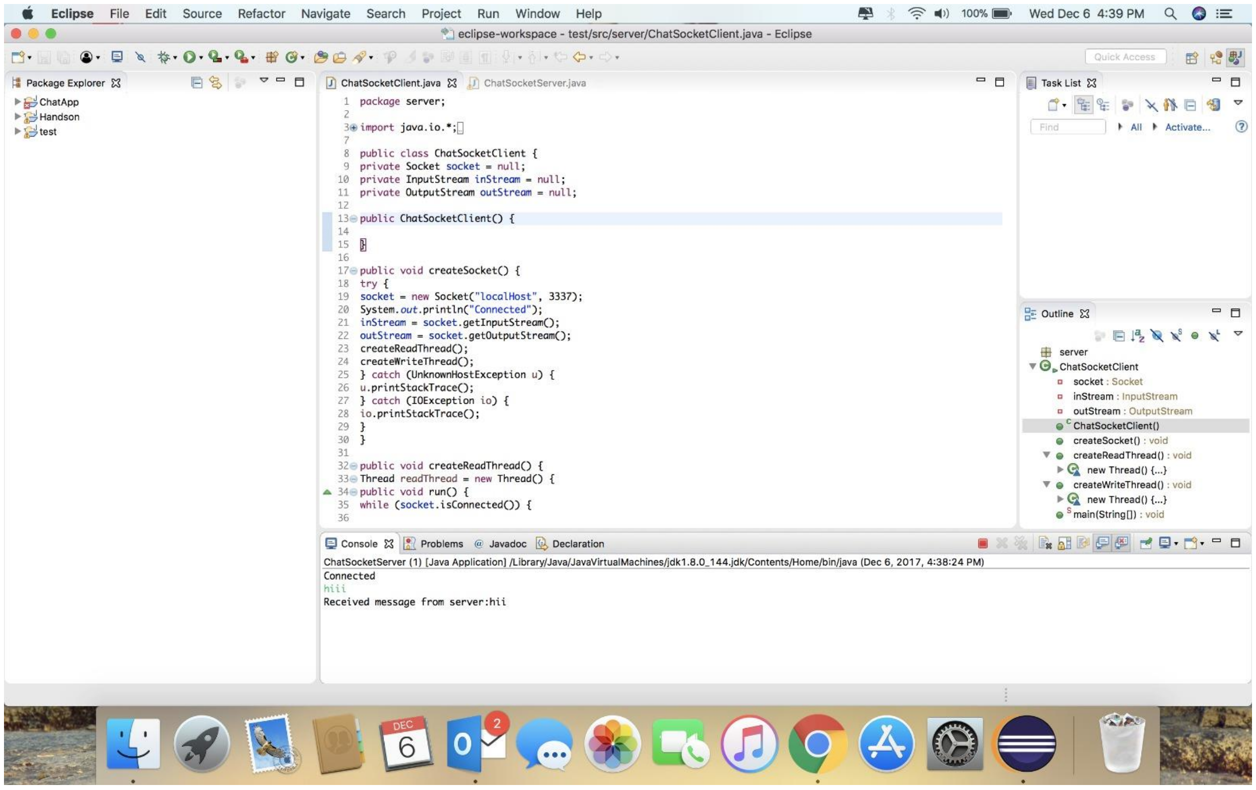
```java
package server;

import java.io.*;

public class ChatSocketClient {
private Socket socket = null;
private InputStream inStream = null;
private OutputStream outStream = null;

public ChatSocketClient() {

}

public void createSocket() {
try {
socket = new Socket("localHost", 3337);
System.out.println("Connected");
inStream = socket.getInputStream();
outStream = socket.getOutputStream();
createReadThread();
createWriteThread();
} catch (UnknownHostException u) {
u.printStackTrace();
} catch (IOException io) {
io.printStackTrace();
}
}

public void createReadThread() {
Thread readThread = new Thread() {
public void run() {
while (socket.isConnected()) {

```

```
ChatSocketServer (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Dec 6, 2017, 4:38:24 PM)
Connected
```



```
package server;

import java.io.*;

public class ChatSocketServer {
private ServerSocket severSocket = null;
private Socket socket = null;
private InputStream inStream = null;
private OutputStream outStream = null;

public ChatSocketServer() {

}

public void createSocket() {
try {
ServerSocket serverSocket = new ServerSocket(3337);
while (true) {
socket = serverSocket.accept();
inStream = socket.getInputStream();
outStream = socket.getOutputStream();
System.out.println("Connected");
createReadThread();
createWriteThread();

}
} catch (IOException io) {
io.printStackTrace();
}
}

public void createReadThread() {
Thread readThread = new Thread() {
```

```
ChatSocketClient [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Dec 6, 2017, 4:38:28 PM)
Connected
Received message from client:hiii
```



```
package server;

import java.io.*;

public class ChatSocketClient {
private Socket socket = null;
private InputStream inStream = null;
private OutputStream outStream = null;

public ChatSocketClient() {

}

public void createSocket() {
try {
socket = new Socket("localHost", 3337);
System.out.println("Connected");
inStream = socket.getInputStream();
outStream = socket.getOutputStream();
createReadThread();
createWriteThread();
} catch (UnknownHostException u) {
u.printStackTrace();
} catch (IOException io) {
io.printStackTrace();
}
}

public void createReadThread() {
Thread readThread = new Thread() {
public void run() {
while (socket.isConnected()) {

```

```
ChatSocketServer (1) [Java Application] /Library/Java/JavaVirtualMachines/jdk1.8.0_144.jdk/Contents/Home/bin/java (Dec 6, 2017, 4:38:24 PM)
Connected
hiii
Received message from server:hii
```

References:

https://way2java.com/networking/chat-program-two-way-communication/

https://www.youtube.com/watch?v=uYRTpMGdf1g

http://pirate.shu.edu/~wachsmut/Teaching/CSAS2214/Virtual/Lectures/chat-client-server.html

http://www.dreamincode.net/forums/topic/259777-a-simple-chat-program-with-clientserver-gui-optional/