Practical No.: 01

Aim: Develop a program for Mutli-Chat Server Application

Server

File → New Project → Java Application →Project Name: MultiThreadChatServerSync → Finish

Source Code:

MultiThreadChatServerSync.java

```
package multithreadchatserversync;

import java.io.DataInputStream;
import java.io.IOException;
import java.io.PrintStream;
import java.net.ServerSocket;
import java.net.Socket;

/**
 *
 * @author TechnoBoy
 */
public class MultiThreadChatServerSync {

    // The server socket.
  private static ServerSocket serverSocket = null;
  // The client socket.
  private static Socket clientSocket = null;

  // This chat server can accept up to maxClientsCount clients' connections.
  private static final int maxClientsCount = 10;
  private static final clientThread[] threads = new clientThread[maxClientsCount];

    public static void main(String[] args) {
        //The Default Port Number.
        int portNumber=2000;
```

```
        if(args.length<1){
            System.out.println("Usage: java MultiThreadChatServerSync <portNumber>\n"+ "Now using
port number=" + portNumber);
        }
        else{
            portNumber = Integer.parseInt(args[0]);
        }

        //Opening a server socket on the portNumber
        try{
            serverSocket =new ServerSocket(portNumber);
        }
        catch(IOException e)
        {
            System.out.println(e);
        }

        //Creating a client socket for new connection
        while(true)
        {
            try
            {
                clientSocket=serverSocket.accept();
                int i=0;
                for(i=0;i<maxClientsCount;i++)
                {
                    if(threads[i]==null){
                        (threads[i] = new clientThread(clientSocket, threads)).start();
                        break;
                    }
                    if(i==maxClientsCount)
                    {
                        PrintStream os=new PrintStream(clientSocket.getOutputStream());;
```

```
                os.println("Server too busy. Try again later.");
                os.close();
                clientSocket.close();
              }
            }
          }
          catch(IOException e)
          {
            System.out.println(e);
          }
        }
      }

//The Chte client Thread
    private static class clientThread extends Thread{

      private String clientName = null;
      private DataInputStream is = null;
      private PrintStream os = null;
      private Socket clientSocket = null;
      private final clientThread[] threads;
      private int maxClientsCount;

      private clientThread(Socket clientSocket, clientThread[] threads) {
        this.clientSocket = clientSocket;
        this.threads = threads;
        maxClientsCount = threads.length;
      }

      public void run(){
        int maxClientsCount = this.maxClientsCount;
        clientThread[] threads = this.threads;
```

```
try
{
   is = new DataInputStream(clientSocket.getInputStream());
   os = new PrintStream(clientSocket.getOutputStream());
   String name;
   while (true) {
      os.println("Enter your name.");
      name = is.readLine().trim();
      if (name.indexOf('@') == -1) {
      break;
      }
      else {
         os.println("The name should not contain '@' character.");
     }
   }
   /*Welcome the new client.*/
   os.println("Welcome " + name+ " to our chat room.\nTo leave enter /quit in a new line.");
   synchronized(this)
   {
      for(int i=0;i<maxClientsCount;i++)
      {
         if (threads[i] != null && threads[i] == this) {
            clientName = "@" + name;
            break;
         }
      }
      for (int i = 0; i < maxClientsCount; i++)
      {
         if (threads[i] != null && threads[i] != this)
         {
            threads[i].os.println("*** A new user " + name+ " entered the chat room !!! ***");
```

```
        }
    }
}

//Starting the converstation

while(true)
{
    String line = is.readLine();
    if (line.startsWith("/quit"))
    {
        break;
    }

    //If the message is privately sent it to give client
    if(line.startsWith("@"))
    {
        String[] words = line.split("\\s", 2);
        if(words.length > 1 && words[1] != null)
        {
            words[1] = words[1].trim();
            if(!words[1].isEmpty())
            {
                synchronized (this)
                {
                    for (int i = 0; i < maxClientsCount; i++)
                    {
                        if (threads[i] != null && threads[i] != this
                        && threads[i].clientName != null
                        && threads[i].clientName.equals(words[0]))
                        {
                            threads[i].os.println("<" + name + "> " + words[1]);
```

```
                        //Message send private
                        this.os.println(">" + name + "> " + words[1]);
                        break;
                    }
                }
            }
        }
    }
    else
    {
        synchronized (this)
        {
            for (int i = 0; i < maxClientsCount; i++)
            {
                if (threads[i] != null && threads[i].clientName != null)
                {
                    threads[i].os.println("<" + name + "> " + line);
                }
            }
        }
    }
}
//The message is broadcasted pubilcally
synchronized(this)
{
    for (int i = 0; i < maxClientsCount; i++)
    {
        if (threads[i] != null && threads[i] != this&& threads[i].clientName != null)
        {
            threads[i].os.println("*** The user " + name + " is leaving the chat room !!! ***");
        }
```

```
            }
        }
        os.println("*** Bye " + name + " ***");

        //Creating the thread null for new client
        synchronized (this)
        {
            for (int i = 0; i < maxClientsCount; i++)
            {
                if (threads[i] == this)
                {
                    threads[i] = null;
                }
            }
        }

        //Close the all connection

        is.close();
        os.close();
        clientSocket.close();
      }
      catch(IOException e)
      {
        System.out.println(e);
      }
    }
  }
}
```

Client

Right Click on Project → New → Java Class-> Class Name: *MultiThreadChatClient* → package : multithreadchatserversync → Finish

MultiThreadChatClient.java

```
import java.io.DataInputStream;
import java.io.PrintStream;
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.io.IOException;
import java.net.Socket;
import java.net.UnknownHostException;

public class MultiThreadChatClient implements Runnable {

  // The client socket
  private static Socket clientSocket = null;
  // The output stream
  private static PrintStream os = null;
  // The input stream
  private static DataInputStream is = null;

  private static BufferedReader inputLine = null;
  private static boolean closed = false;

  public static void main(String[] args) {

    // The default port.
    int portNumber = 2222;
    // The default host.
    String host = "localhost";
```

```
if (args.length < 2) {
  System.out
      .println("Usage: java MultiThreadChatClient <host> <portNumber>\n"
          + "Now using host=" + host + ", portNumber=" + portNumber);
} else {
  host = args[0];
  portNumber = Integer.valueOf(args[1]).intValue();
}

/*
 * Open a socket on a given host and port. Open input and output streams.
 */
try {
  clientSocket = new Socket(host, portNumber);
  inputLine = new BufferedReader(new InputStreamReader(System.in));
  os = new PrintStream(clientSocket.getOutputStream());
  is = new DataInputStream(clientSocket.getInputStream());
} catch (UnknownHostException e) {
  System.err.println("Don't know about host " + host);
} catch (IOException e) {
  System.err.println("Couldn't get I/O for the connection to the host "
      + host);
}

/*
 * If everything has been initialized then we want to write some data to the
 * socket we have opened a connection to on the port portNumber.
 */
if (clientSocket != null && os != null && is != null) {
  try {

    /* Create a thread to read from the server. */
```

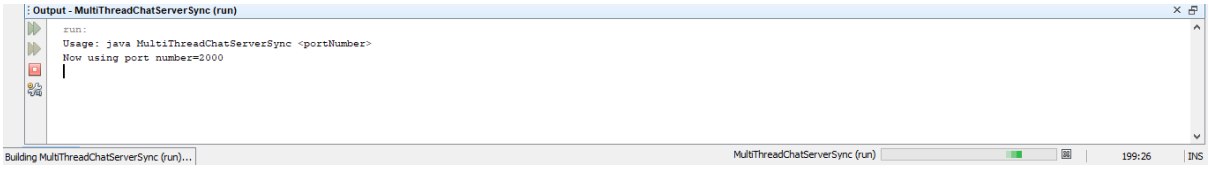
```
      new Thread(new MultiThreadChatClient()).start();
      while (!closed) {
        os.println(inputLine.readLine().trim());
      }
      /*
       * Close the output stream, close the input stream, close the socket.
       */
      os.close();
      is.close();
      clientSocket.close();
    } catch (IOException e) {
      System.err.println("IOException:  " + e);
    }
  }
}

/*
 * Create a thread to read from the server. (non-Javadoc)
 *
 * @see java.lang.Runnable#run()
 */
public void run() {
  /*
   * Keep on reading from the socket till we receive "Bye" from the
   * server. Once we received that then we want to break.
   */
  String responseLine;
  try {
    while ((responseLine = is.readLine()) != null) {
      System.out.println(responseLine);
      if (responseLine.indexOf("*** Bye") != -1)
        break;
```

```
    }
    closed = true;
   } catch (IOException e) {
    System.err.println("IOException:  " + e);
   }
  }
```

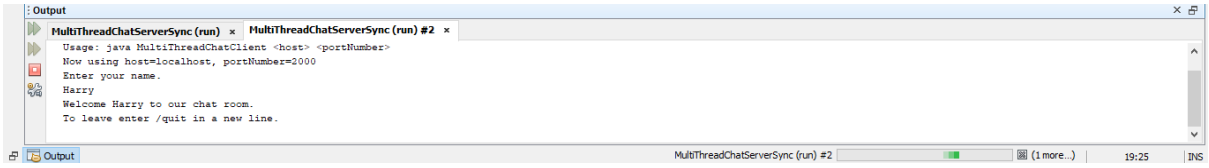
} **Output:**

Run → *MultiThreadChatServerSync.java*



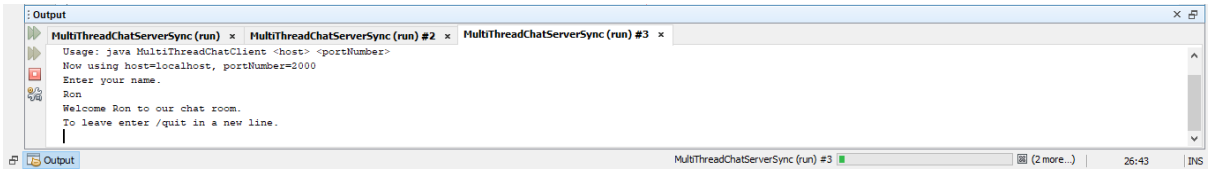
First User

Run → *MultiThreadChatClient.java* →Enter Name



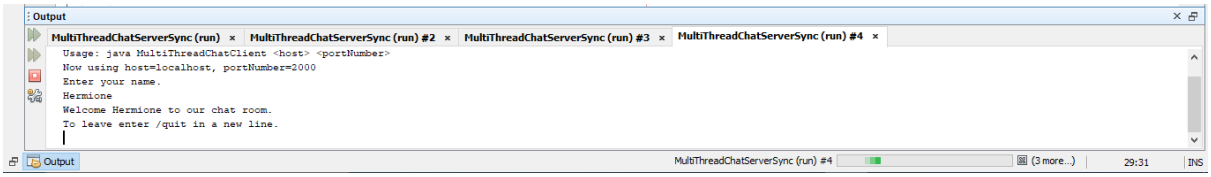
Second User

Run → *MultiThreadChatClient.java* →Enter Name

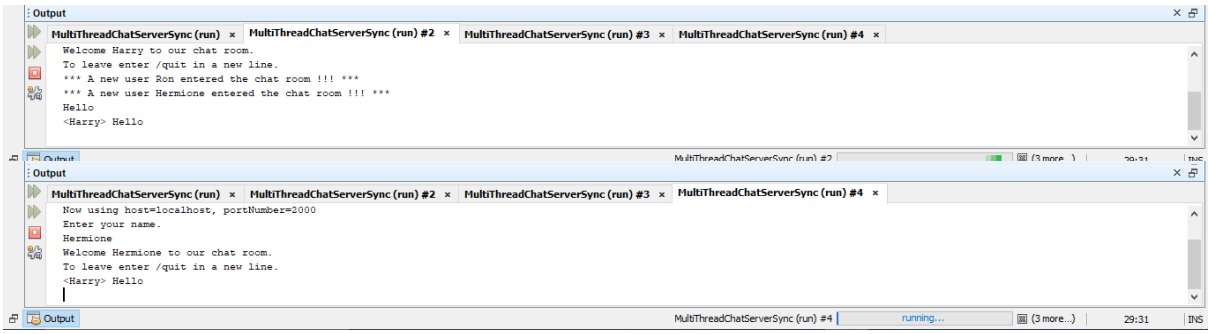


Third User

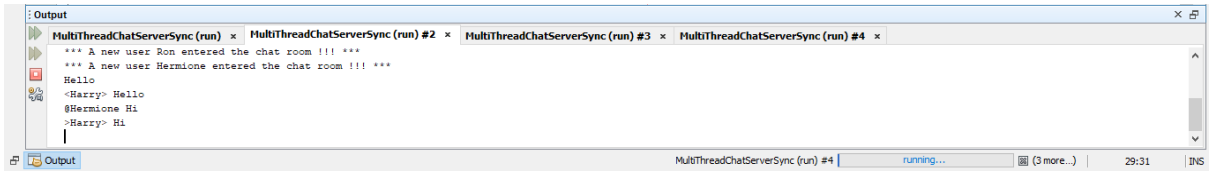
Run → *MultiThreadChatClient.java* →Enter Name

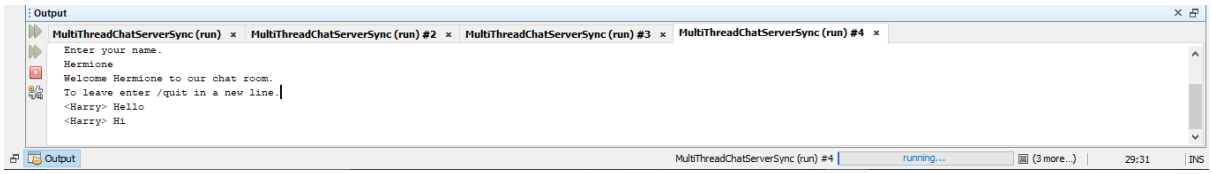


First User sending message to all

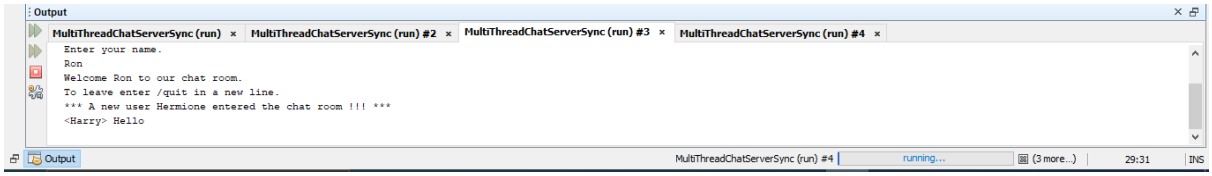


First User sending message to third user





<u>----Second user didn't receive the message---</u>



User leaving