**Thread concept: clone, threads of java**

**Subject - Unix Operating System**

**Name – Hemant Sharma**

**PRN – 22610001 Class – TYIT**

**Assignment No – 4(a)**

**Title-** Write a multi-threaded program in java/c for chatting (multi-user and multi-terminal) using threads.

**Objectives:**

1. To learn about threading in Linux/Unix and Java and difference between them.
2. Use of system call/library to write effective programs.

**Theory:**As normal, we will create two Java files, **Server.java** and **Client.java**. Server file contains two classes namely **Server** (public class for creating server) and **ClientHandler** (for handling any client using multi-threading). **Client** file contain only one public class Client (for creating a client).

**Server class:** The steps involved on server side are similar to the article Socket Programming In Java with a slight change to create the thread object after obtaining the streams and port number.

* Establishing the Connection: Server socket object is initialized and inside a while loop a socket object continuously accepts incoming connection.
* Obtaining the Streams: The inputstream object and outputstream object is extracted from the current requests’ socket object.
* Creating a handler object: After obtaining the streams and port number, a new ClientHandler object (the above class) is created with these parameters.
* Invoking the start() method : The start() method is invoked on this newly created thread object.
* ClientHandler class: As we will be using separate threads for each request, lets understand the working and implementation of the ClientHandler class extending Threads. An object of this class will be instantiated each time a request comes.
* First of all, this class extends Thread so that its objects assume all properties of Threads.
* Secondly, the constructor of this class takes three parameters, which can uniquely identify any incoming request, i.e. a Socket, a DataInput Stream to read from and a DataOutputStream to write to. Whenever we receive any request of client, the server extracts its port number, the DataInputStream object and DataOutputStream object and creates a new thread object of this class and invokes start() method on it.
* Inside the run() method of this class, it performs three operations: request the user to specify whether time or date needed, read the answer from input stream object and accordingly write the output on the output stream object.

**Program:**

**Server:**

import java.io.\*;

import java.net.\*;

import java.util.\*;

public class ChatServer {

    private static Set<ClientHandler> clientHandlers = new HashSet<>();

    public static void main(String[] args) {

        int port = 12345;

        try (ServerSocket serverSocket = new ServerSocket(port)) {

            System.out.println("Chat Server is running on port " + port);

            while (true) {

                Socket socket = serverSocket.accept();

                System.out.println("New client connected: " + socket);

                ClientHandler client = new ClientHandler(socket);

                clientHandlers.add(client);

                new Thread(client).start();

            }

        } catch (IOException e) {

            e.printStackTrace();

        }

    }

    static void broadcastMessage(String message, ClientHandler sender) {

        for (ClientHandler client : clientHandlers) {

            if (client != sender) {

                client.sendMessage(message);

            }

        }

    }

    static void removeClient(ClientHandler client) {

        clientHandlers.remove(client);

    }

}

class ClientHandler implements Runnable {

    private Socket socket;

    private PrintWriter out;

    private BufferedReader in;

    public ClientHandler(Socket socket) {

        this.socket = socket;

    }

    @Override

    public void run() {

        try {

            in = new BufferedReader(new InputStreamReader(socket.getInputStream()));

            out = new PrintWriter(socket.getOutputStream(), true);

            out.println("Welcome to the chat!");

            String message;

            while ((message = in.readLine()) != null) {

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

                ChatServer.broadcastMessage(message, this);

            }

        } catch (IOException e) {

            e.printStackTrace();

        } finally {

            ChatServer.removeClient(this);

            try {

                socket.close();

            } catch (IOException e) {

                e.printStackTrace();

            }

        }

    }

    public void sendMessage(String message) {

        out.println(message);

    }

}

**Client:**

import java.io.\*;

import java.net.\*;

public class ChatClient {

    public static void main(String[] args) {

        String hostname = "localhost";

        int port = 12345;

        try (Socket socket = new Socket(hostname, port);

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

             PrintWriter out = new PrintWriter(socket.getOutputStream(), true);

             BufferedReader in = new BufferedReader(new InputStreamReader(socket.getInputStream()))) {

            System.out.println("Connected to chat server.");

            new Thread(() -> {

                String serverMessage;

                try {

                    while ((serverMessage = in.readLine()) != null) {

                        System.out.println(serverMessage);

                    }

                } catch (IOException e) {

                    e.printStackTrace();

                }

            }).start();

            String userInput;

            while ((userInput = reader.readLine()) != null) {

                out.println(userInput);

            }

        } catch (IOException e) {

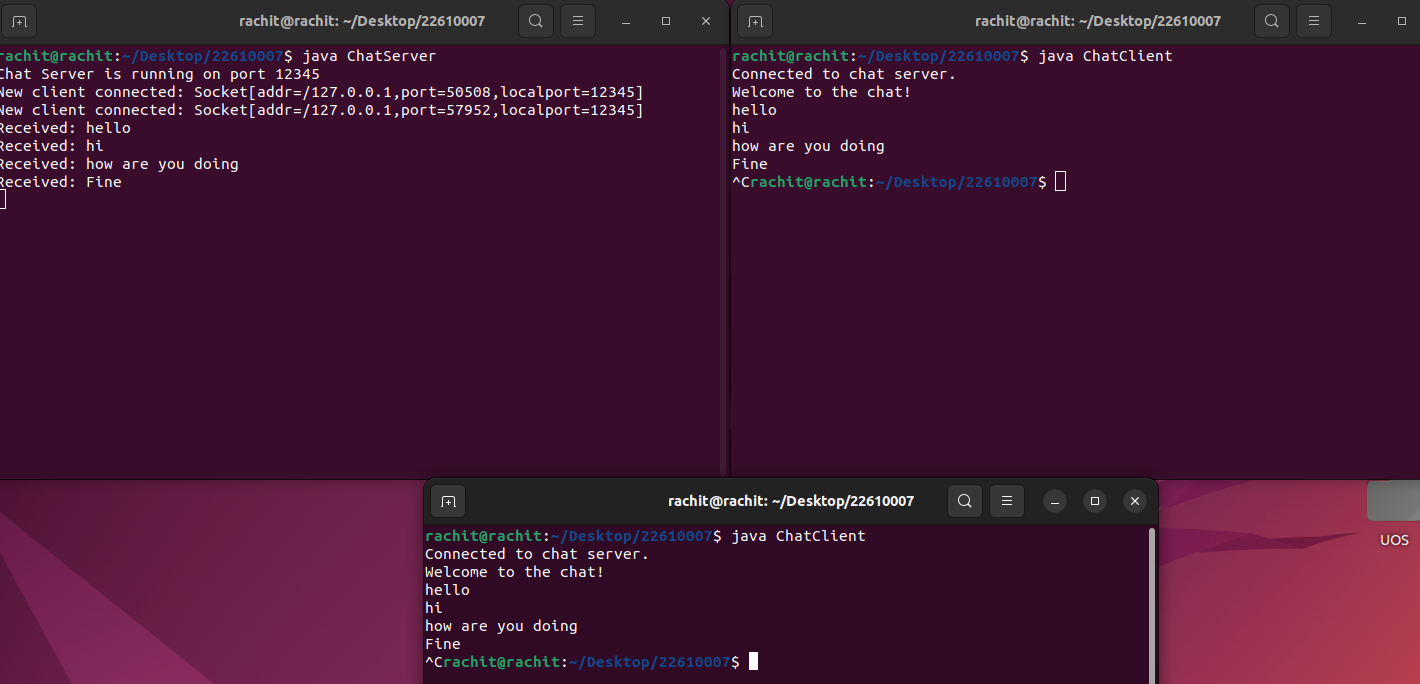
            e.printStackTrace();

        }

    }

}

**Output:**

****

**Conclusion:**

Various concepts and effective programming in Java using threads and sockets was studied. The concept of threading and multithreading understood.