Clock Synchronization

**Server.java**

import java.io.\*; import java.net.\*;

import java.text.SimpleDateFormat; import java.util.\*;

public class **Server** {

private static final int PORT = 5000;

private static List<ClientHandler> clients = new ArrayList<>();

private static SimpleDateFormat timeFormat = new **SimpleDateFormat**("HH:mm"); public static void **main**(String[] args) {

try (ServerSocket serverSocket = new **ServerSocket**(PORT)) { System.out.**println**("Master Clock Server is running..."); while (clients.**size**() < 3) {

Socket clientSocket = serverSocket.**accept**(); ClientHandler client = new **ClientHandler**(clientSocket); clients.**add**(client);

System.out.**println**("Client connected: " + clientSocket.**getInetAddress**());

}

List<Date> clientTimes = new ArrayList<>(); List<Long> timeOffsets = new ArrayList<>();

Date masterTime = timeFormat.**parse**("03:10"); *// Set master clock time*

*(3:10)*

System.out.**println**("\nMaster Clock Time: " +

timeFormat.**format**(masterTime));

*// Collect client times and compute offsets*

for (ClientHandler client : clients) {

Date clientTime = client.**receiveTime**(); clientTimes.**add**(clientTime);

long offset = (clientTime.**getTime**() - masterTime.**getTime**()) / 60000;

*// Offset in minutes*

timeOffsets.**add**(offset);

}

*// Print old times and offsets* System.out.**println**("\nClient Old Times and Offsets:"); for (int i = 0; i < clientTimes.**size**(); i++) {

System.out.**println**("Client " + (i + 1) + " Time: " + timeFormat.**format**(clientTimes.**get**(i)) +

" | Offset: " + timeOffsets.**get**(i) + " min");

}

*// Calculate new synchronized time*

long sumOffsets = 0;

for (long offset : timeOffsets) { sumOffsets += offset;

60000));

}

long averageOffset = sumOffsets / timeOffsets.**size**(); System.out.**println**("\nSum of Offsets: " + sumOffsets + " min"); System.out.**println**("Average Offset: " + averageOffset + " min");

Date newMasterTime = new **Date**(masterTime.**getTime**() + (averageOffset \*

System.out.**println**("New Synchronized Time: " +

timeFormat.**format**(newMasterTime));

*// Send adjustments to clients*

for (int i = 0; i < clients.**size**(); i++) {

long adjustment = (newMasterTime.**getTime**() - clientTimes.**get**(i).**getTime**()) / 60000;

clients.**get**(i).**sendAdjustment**(adjustment); System.out.**println**("Client " + (i + 1) + " Adjustment: " +

adjustment + " min");

}

} catch (IOException | java.text.ParseException e) { e.**printStackTrace**();

}

}

}

class **ClientHandler** { private Socket socket;

private PrintWriter output; private BufferedReader input;

private SimpleDateFormat timeFormat = new **SimpleDateFormat**("HH:mm"); public **ClientHandler**(Socket socket) throws IOException {

this.socket = socket;

this.output = new **PrintWriter**(socket.**getOutputStream**(), true); this.input = new **BufferedReader**(new

**InputStreamReader**(socket.**getInputStream**()));

}

public Date **receiveTime**() throws IOException { String timeString = input.**readLine**();

try {

return timeFormat.**parse**(timeString);

} catch (Exception e) { return new **Date**();

}

}

public void **sendAdjustment**(long offset) { output.**println**(offset);

}

}

**Clent.java**

import java.io.\*; import java.net.\*;

import java.text.SimpleDateFormat; import java.util.Date;

public class **Client** {

private static final String SERVER\_IP = "localhost"; private static final int SERVER\_PORT = 5000;

private static SimpleDateFormat timeFormat = new **SimpleDateFormat**("HH:mm"); public static void **main**(String[] args) {

try (Socket socket = new **Socket**(SERVER\_IP, SERVER\_PORT)) {

PrintWriter output = new **PrintWriter**(socket.**getOutputStream**(), true); BufferedReader input = new **BufferedReader**(new

**InputStreamReader**(socket.**getInputStream**()));

*// Predefined client times (Manually Set)*

String[] clientTimes = {"03:25", "03:00", "02:50"}; *// 3:25, 3:00, 2:50*

int clientID = (int) (Math.**random**() \* 3); *// Randomly select one* Date localTime = timeFormat.**parse**(clientTimes[clientID]); output.**println**(timeFormat.**format**(localTime)); System.out.**println**("Old Time: " + timeFormat.**format**(localTime));

*// Receive adjustment value*

long adjustment = Long.**parseLong**(input.**readLine**());

Date newTime = new **Date**(localTime.**getTime**() + (adjustment \* 60000));

*// Print adjustment details*

System.out.**println**("Received Adjustment: " + adjustment + " min"); System.out.**println**("New Synchronized Time: " +

timeFormat.**format**(newTime));

} catch (IOException | java.text.ParseException e) { e.**printStackTrace**();

}

}

}