Source Code:

**CLIENT CODE:**

**package** client;

**import** java.io.IOException;

**import** java.net.Socket;

**public** **class** Client {

**public** **static** **void** main(String[] args) {

System.***out***.println("My name is [Your Name]");

**for** (**int** i = 1; i <= 80; i++) {

String threadID = "thread" + i;

SubClient subClient = **new** SubClient(threadID);

subClient.start();

}

}

}

**class** SubClient **extends** Thread {

**private** String threadID;

**public** SubClient(String threadID) {

**this**.threadID = threadID;

}

**public** **void** run() {

**boolean** connected = **false**;

**while** (!connected) {

**try** {

Socket socket = **new** Socket("localhost", 8000);

System.***out***.println(threadID + " connected to server");

connected = **true**;

socket.close();

} **catch** (IOException e) {

// Connection refused, keep trying

}

}

}

}

**ITERATIVE SERVER**

**package** server;

**import** java.io.BufferedReader;

**import** java.io.IOException;

**import** java.io.InputStreamReader;

**import** java.io.PrintWriter;

**import** java.net.ServerSocket;

**import** java.net.Socket;

**import** java.text.SimpleDateFormat;

**import** java.util.Date;

**public** **class** IterativeServer {

**public** **static** **void** main(String[] args) {

**try** {

ServerSocket serverSocket = **new** ServerSocket(8000);

**for** (**int** i = 1; i <= 80; i++) {

String threadID = "thread" + i;

**new** IterativeServerThread(serverSocket.accept(), threadID).start();

}

serverSocket.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**class** IterativeServerThread **extends** Thread {

**private** Socket socket;

**private** String threadID;

**public** IterativeServerThread(Socket socket, String threadID) {

**this**.socket = socket;

**this**.threadID = threadID;

}

**public** **void** run() {

**try** {

BufferedReader inputFromClient = **new** BufferedReader(**new** InputStreamReader(socket.getInputStream()));

PrintWriter outputToClient = **new** PrintWriter(socket.getOutputStream(), **true**);

**long** startTime = System.*currentTimeMillis*();

**int** total = 0;

**for** (**int** i = 0; i < 900000; i++) {

total += i;

}

outputToClient.println(total);

SimpleDateFormat formatter = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");

String finishTime = formatter.format(**new** Date());

System.***out***.println(threadID + " start time: " + formatter.format(**new** Date(startTime)));

System.***out***.println(threadID + " finish time: " + finishTime);

System.***out***.println(threadID + " total: " + total);

**long** delay = finishTime- startTime;

System.***out***.println("Thread " + threadID + " delay: " + delay);

outputToClient.println(finishTime);

socket.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

}

**CONCURRENT SERVER**

**package** server;

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

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

**import** java.text.SimpleDateFormat;

**import** java.util.\*;

**import** java.util.concurrent.\*;

**public** **class** ConcurrentServer {

**private** **static** **final** **int** ***NUM\_THREADS*** = 80;

**public** **static** **void** main(String[] args) **throws** IOException {

ServerSocket serverSocket = **new** ServerSocket(8000);

ExecutorService executorService = Executors.*newFixedThreadPool*(***NUM\_THREADS***);

List<Future<Void>> futures = **new** ArrayList<>();

System.***out***.println("Server started");

**long** startTime = System.*currentTimeMillis*();

**for** (**int** i = 0; i < ***NUM\_THREADS***; i++) {

**int** threadID = i + 1;

futures.add(executorService.submit(**new** SubClientHandler(serverSocket.accept(), threadID)));

}

executorService.shutdown();

**try** {

executorService.awaitTermination(Long.***MAX\_VALUE***, TimeUnit.***NANOSECONDS***);

} **catch** (InterruptedException e) {

System.***out***.println("Interrupted: " + e);

}

**long** endTime = System.*currentTimeMillis*();

System.***out***.println("All computations completed");

**for** (**int** i = 0; i < ***NUM\_THREADS***; i++) {

**try** {

futures.get(i).get();

} **catch** (InterruptedException | ExecutionException e) {

System.***out***.println("Exception: " + e);

}

}

**long** delay = endTime - startTime;

System.***out***.println("Total time: " + delay + "ms");

}

**private** **static** **class** SubClientHandler **implements** Callable<Void> {

**private** **final** Socket socket;

**private** **final** **int** threadID;

**public** SubClientHandler(Socket socket, **int** threadID) {

**this**.socket = socket;

**this**.threadID = threadID;

}

@Override

**public** Void call() {

**try** {

System.***out***.println("Thread " + threadID + " connected to server at " + **new** Date());

DataInputStream input = **new** DataInputStream(socket.getInputStream());

DataOutputStream output = **new** DataOutputStream(socket.getOutputStream());

**long** startTime = System.*currentTimeMillis*();

**int** total = 0;

**for** (**int** i = 0; i < 900000; i++) {

total += i;

}

output.writeInt(total);

SimpleDateFormat formatter = **new** SimpleDateFormat("yyyy-MM-dd HH:mm:ss.SSS");

String endTime = formatter.format(**new** Date());

//System.out.println("Thread " + threadID + " computation finished at " + new Date());

System.***out***.println("Thread " + threadID + " start time: " + formatter.format(**new** Date(startTime)));

System.***out***.println("Thread " + threadID + " end time: " + endTime);

**long** delay = finishTime- startTime;

System.***out***.println("Thread " + threadID + " delay: " + delay);

socket.close();

} **catch** (IOException e) {

System.***out***.println("Exception: " + e);

}

**return** **null**;

}

}

}