**Q9.Java Program to Start One Thread More than Once**

**Public class TestThreadTwice1 extends Thread{**

**Public void run(){**

**System.out.println(“running…”);**

**}**

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

**TestThreadTwice1 t1=new TestThreadTwice1();**

**T1.start();**

**T1.start();**

**}**

**}**

**Q10.Java Program to Check CurrentThread in Multi Threading Concept**

**Answer::**

**/\* Creating three threads using the class Thread and then running them concurrently. \*/**

**Class ThreadA extends Thread{**

**Public void run( ) {**

**For(int I = 1; I <= 5; i++) {**

**System.out.println(“From Thread A with I = “+ -1\*i);**

**}**

**System.out.println(“Exiting from Thread A …”);**

**}**

**}**

**Class ThreadB extends Thread {**

**Public void run( ) {**

**For(int j = 1; j <= 5; j++) {**

**System.out.println(“From Thread B with j= “+2\* j);**

**}**

**System.out.println(“Exiting from Thread B …”);**

**}**

**}**

**Class ThreadC extends Thread{**

**Public void run( ) {**

**For(int k = 1; k <= 5; k++) {**

**System.out.println(“From Thread C with k = “+ (2\*k-1));**

**}**

**System.out.println(“Exiting from Thread C …”);**

**}**

**}**

**Public class Demonstration\_111 {**

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

**ThreadA a = new ThreadA();**

**ThreadB b = new ThreadB();**

**ThreadC c = new ThreadC();**

**a.start();**

**b.start();**

**c.start();**

**System.out.println(“… Multithreading is over “);**

**}**

**}**

**Q11.Java Program to Create a Server with 2 Threads to Communicate with Serveral Client.**

**Import java.io.\*;**

**Import java.util.\*;**

**Import java.net.\*;**

**// Server class**

**Public class Server**

**{**

**// Vector to store active clients**

**Static Vector<ClientHandler> ar = new Vector<>();**

**// counter for clients**

**Static int I = 0;**

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

**{**

**// server is listening on port 1234**

**ServerSocket ss = new ServerSocket(1234);**

**Socket s;**

**// running infinite loop for getting**

**// client request**

**while (true)**

**{**

**// Accept the incoming request**

**s = ss.accept();**

**System.out.println("New client request received : " + s);**

**// obtain input and output streams**

**DataInputStream dis = new DataInputStream(s.getInputStream());**

**DataOutputStream dos = new DataOutputStream(s.getOutputStream());**

**System.out.println("Creating a new handler for this client...");**

**// Create a new handler object for handling this request.**

**ClientHandler mtch = new ClientHandler(s,"client " + i, dis, dos);**

**// Create a new Thread with this object.**

**Thread t = new Thread(mtch);**

**System.out.println("Adding this client to active client list");**

**// add this client to active clients list**

**ar.add(mtch);**

**// start the thread.**

**t.start();**

**// increment i for new client.**

**// i is used for naming only, and can be replaced**

**// by any naming scheme**

**}**

**}**

**}**

**// ClientHandler class**

**class ClientHandler implements Runnable**

**{**

**Scanner scn = new Scanner(System.in);**

**private String name;**

**final DataInputStream dis;**

**final DataOutputStream dos;**

**Socket s;**

**boolean isloggedin;**

**// constructor**

**public ClientHandler(Socket s, String name,**

**DataInputStream dis, DataOutputStream dos) {**

**this.dis = dis;**

**this.dos = dos;**

**this.name = name;**

**this.s = s;**

**this.isloggedin=true;**

**}**

**@Override**

**public void run() {**

**String received;**

**while (true)**

**{**

**try**

**{**

**// receive the string**

**received = dis.readUTF();**

**System.out.println(received);**

**if(received.equals("logout")){**

**this.isloggedin=false;**

**this.s.close();**

**break;**

**}**

**// break the string into message and recipient part**

**StringTokenizer st = new StringTokenizer(received, "#");**

**String MsgToSend = st.nextToken();**

**String recipient = st.nextToken();**

**// search for the recipient in the connected devices list.**

**// ar is the vector storing client of active users**

**for (ClientHandler mc : Server.ar)**

**{**

**// if the recipient is found, write on its**

**// output stream**

**if (mc.name.equals(recipient) && mc.isloggedin==true)**

**{**

**mc.dos.writeUTF(this.name+" : "+MsgToSend);**

**break;**

**}**

**}**

**} catch (IOException e)**

**{**

**e.printStackTrace();**

**}**

**}**

**try**

**{**

**// closing resources**

**this.dis. close();**

**this.dos.close();**

**}catch(IOException e){**

**e.printStackTrace();**

**}**

**Output:**

**New client request received : Socket[addr=/127.0.0.1,port=61818,localport=1234]**

**Creating a new handler for this client...**

**Adding this client to active client list**

**New client request received : Socket[addr=/127.0.0.1,port=61819,localport=1234]**

**Creating a new handler for this client...**

**Adding this client to active client list**

**Q12.Java Program to Create a Client That Receive Message From the Server**

**package**

**com.journaldev.socket;**

**import**

**java.io.IOException;**

**import**

**java.io.ObjectInputStream;**

**import**

**java.io.ObjectOutputStream;**

**import**

**java.lang.ClassNotFoundException;**

**import**

**java.net.ServerSocket;**

**import**

**java.net.Socket;**

**public**

**class**

**SocketServerExample**

**{**

**//static ServerSocket variable**

**private**

**static**

**ServerSocket server;**

**//socket server port on which it will listen**

**private**

**static**

**int**

**port =**

**9876**

**;**

**Public static void main (String args[]) throws IOException, ClassNotFoundException**

**{**

**//create the socket server object**

**server =**

**new**

**ServerSocket(port);**

**//keep listens indefinitely until receives 'exit' call or program terminates**

**while**

**(**

**true**

**){ System.out.println(**

**"Waiting for the client request"**

**);**

**//creating socket and waiting for client connection**

**Socket socket = server.accept();**

**//read from socket to ObjectInputStream object**

**ObjectInputStream ois =**

**new**

**ObjectInputStream(socket.getInputStream());**

**//convert ObjectInputStream object to String**

**String message = (String) ois.readObject();**

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

**//create ObjectOutputStream**

**objectObjectOutputStream**

**oos =newObjectOutputStream(socket.getOutputStream());**

**//write object to Socket**

**oos.writeObject("Hi Client "+message);**

**ois.close();**

**oos.close();**

**socket.close();**

**//terminate the server if clientnoreCase("exit"))break;**

**} System.out.println("Shutting down Socket server!!");**

**//close the Server**

**Socket .close();**

**}**

**}**

**Q13.Java Program of a Multithreaded Implementation of Any Parallelized Divide-Conquer Algorithm**

**Public class ParallelMergeSort extends RecursiveAction {**

**Private final int[] array;**

**Private final int[] helper;**

**Private final int low;**

**Private final int high;**

**Public ParallelMergeSort(final int[] array, final int low, final int high) {**

**This.array = array;**

**Helper = new int[array.length];**

**This.low = low;**

**This.high = high;**

**}**

**@Override**

**Protected void compute() {**

**If (low < high) {**

**Final int middle = (low + high) / 2;**

**Final ParallelMergeSort left =**

**New ParallelMergeSort(array, low, middle);**

**Final ParallelMergeSort right =**

**New ParallelMergeSort(array, middle + 1, high);**

**invokeAll(left, right);**

**merge(array, helper, low, middle, high);**

**}**

**}**

**}**