```
/****************** Copyright 2002 The Hartmann Software Group, LLC
****************/

                            /* Server Side Socket Program */


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

public class Server implements Runnable
{
    private BufferedReader reader;
    private ThreadGroup group;
    private Thread keyboardreader;
    static public boolean bquit = true;

    public Server() throws Exception
    {
        System.out.print(
        "Enter in the server port or \"quit\" to end the program: ");
        reader= new BufferedReader(new InputStreamReader(System.in));
    }

/*******************       Keyboard input information for the Server
(Adminstration) ***********************/

    public void KeyBoardInput() throws IOException
    {
        String input = reader.readLine();

        if(input.equals("quit"))
        {
            System.exit(0);
        }

        ServerSocket ss = new ServerSocket(Integer.parseInt(input));

        System.out.println("Server listening at " + InetAddress.getLocalHost()
                                + " on port " + ss.getLocalPort());

        (new Thread(this)).start();

        group = new ThreadGroup("Client Thread Group");


        while(Server.bquit)
        {
            try
            {
                ss.setSoTimeout(2000);
                Socket s = ss.accept();
                ClientThread ct = new ClientThread(group,s);
            }
            catch(SocketTimeoutException e)
            {
            }
        }

        ss.close();
    }

    public static void main(String[] args)
    {
        try
        {
            Server s = new Server();
```

```
            s.KeyBoardInput();
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

/******************  This method contains the Server keyboard input routines
**********************/

    public void run()
    {
        try
        {
            while(Server.bquit)
            {
                if(reader.readLine().equals("quit"))
                {
                    System.out.println("Shutting Down Server");
                    Server.bquit = false;
                    Thread[] threads = new Thread[group.activeCount()];
                    group.enumerate(threads);

                    for(int i=0; i < threads.length; i++)
                    {
                        System.out.println(group.activeCount());
                        ((ClientThread)threads[i]).close();

                    }

                    break;
                }
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}

/******************        Each Client Connection is a seperate thread
**********************/

class ClientThread extends Thread
{
    private Socket sock;
    private ObjectOutputStream objectout;
    private ObjectInputStream objectin;
    private String name;
    private boolean bname = false;

    public ClientThread(ThreadGroup t, Socket s)
    {
        super(t,"");

        try
        {
            sock = s;
            objectout = new ObjectOutputStream(s.getOutputStream());
            objectin = new ObjectInputStream(s.getInputStream());
            start();
        }
        catch(Exception e)
        {
            e.printStackTrace();
```

```
        }
    }

    public void close() throws Exception
    {
        objectout.writeObject("quit");
    }

/****************** This method is the 'main' method of this thread printing
and broadcasting messages according **********************/

    public void run()
    {
        try
        {
            while(Server.bquit)
            {
                Object ob = objectin.readObject();

                if(ob instanceof String)
                {
                    if(((String)ob).equals("quit"))
                    {
                        objectout.writeObject("Bye");
                        objectout.close();
                        objectin.close();
                        break;
                    }
                    else if(!bname)
                    {
                        if(((String)ob).substring(0,4).equals("Name"))
                        {
                            name = ((String)ob).substring(7);
                            setName(name);
                            bname = true;
                        }
                    }
                    else
                    {
                        System.out.println("Input from " + name + ":  " + ob);

/******************        Broadcasting to messages to other clients
**********************/

                        ClientThread[] threadarray = new ClientThread[(
                        getThreadGroup().activeCount())];
                        getThreadGroup().enumerate(threadarray);

                        for(int i=0; i< threadarray.length; i++)
                        {
                            if(!((ClientThread)threadarray[i]).getName().
                            equals(name))
                            {
                                ((ClientThread)threadarray[i]).getOutputStream
                                ().writeObject("Input from " + name + ":  " +
                                ob);
                            }
                        }
                    }
                }
/******************        Non-String Objects are not accepted
**********************/
                else
                {
                    objectout.writeObject(
                    "This server accepts only String objects");
                }
```

```
            }
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }

    public ObjectOutputStream getOutputStream()
    {
        return objectout;
    }
}
```