**Karachi Institute of Economics & Technology**

*College of Computing & Information Sciences*



**Network Programming**

**Lab Manual**

**Name**: Muhammad Fasih Rohan

**Student Id:** 62725

**Semester**: Fall 2022

**Class Id:** 110858

**Instructor:** Miss Misbah Anwer

**Contents**

* [Lab 1 (OOP Pillars) 3](#_bookmark0)

[Lab 1 (Button on WinForms) 7](#_bookmark0)

[Lab 1 (Hello world on web) 8](#_bookmark0)

* [Lab 2 (Client – Server Connection) 10](#_bookmark5)
* [Lab 3 (Client – Server Communication) 12](#_bookmark5)
* [Lab 4 (Client - Server Communication using TCP on WinForms) 15](#_bookmark11)
* [Lab 5 (Listing 3.1) 18](#_bookmark11)

[Lab 5 (Listing 3.2) 19](#_bookmark11)

* [Lab 6 (Listing 3.3) 21](#_bookmark11)

[Lab 6 (Listing 3.4) 22](#_bookmark11)

[Lab 6 (Listing 5.1 & 5.2) 24](#_bookmark11)

* [Lab 7 (Listing 5.3 & 5.4) 27](#_bookmark11)

[Lab 7 (Listing 5.5 & 5.6) 30](#_bookmark11)

[Lab 7 (Listing 5.7 & 5.8) 34](#_bookmark11)

[Lab 7 (Listing 5.10 & 5.11) 39](#_bookmark11)

* [Lab 8 (Listing 6.1 & 6.2) 44](#_bookmark11)

[Lab 8 (Listing 6.1 & 6.3) 46](#_bookmark11)

[Lab 8 (Listing 6.4 & 6.5) 49](#_bookmark11)

[Lab 8 (Listing 6.1 & 6.7) 52](#_bookmark11)

[Lab 8 (Listing 6.1 & 6.9) 55](#_bookmark11)

* [Lab 9 (Asynchronous)](#_bookmark11) 56

***Lab # 1 OOP Pillars:***

**Code:**

using System;

namespace NetworkProgrammingTask\_1

{

static class Program

static void Main()

{

OneBook OB = new OneBook();

OB.Price();

TwoBook TB = new TwoBook();

TB.Price();

ThreeBook THB = new ThreeBook();

THB.Price();

Books BK = new Books("Tariq Book Town");

Books BK1 = new Books("EnglishBook", "ScienceBook", "MathsBook");

Economy E = new Economy();

E.Bill();

Business B = new Business();

B.Bill();

VIP V = new VIP();

V.Bill();

}

}

class Books

{

private string \_EnglishBook;

private string \_ScienceBook;

private string \_MathsBook;

private string \_UrduBook;

public Books(string EnglishBook)

{

this.\_EnglishBook = EnglishBook;

if (EnglishBook == \_EnglishBook)

{ Console.WriteLine("English Book costs Rs.50"); }

}

public Books(string ScienceBook, string MathsBook, string UrduBook)

{

this.\_ScienceBook = ScienceBook;

this.\_MathsBook = MathsBook;

this.\_UrduBook = UrduBook;

if (\_ScienceBook == ScienceBook)

{ Console.WriteLine("Science Book costs Rs.50");}

if (MathsBook == \_MathsBook)

{Console.WriteLine("Maths Book costs Rs.80"); }

if (UrduBook == \_UrduBook)

{Console.WriteLine("Urdu Book costs Rs.100"); }

}

}

public abstract class BoookShop

{

public abstract double Price();

protected double oneBookCharges = 8000;

}

public class OneBook : BoookShop

{

public override double Price()

{ return oneBookCharges \* 1; }

}

public class TwoBook : BoookShop

{

public override double Price()

{ return oneBookCharges \* 2; }

}

public class ThreeBook : BoookShop

{

public override double Price()

{ return oneBookCharges \* 3; }

}

class CustomerType

{

protected double BillCharge = 20000;

public virtual double Bill()

{ return BillCharge \* 0; }

}

class Economy : CustomerType

{

public override double Bill()

{return BillCharge \* 1;}

}

class Business : CustomerType

{

public override double Bill()

{ return BillCharge \* 2; }

}

class VIP : CustomerType

{

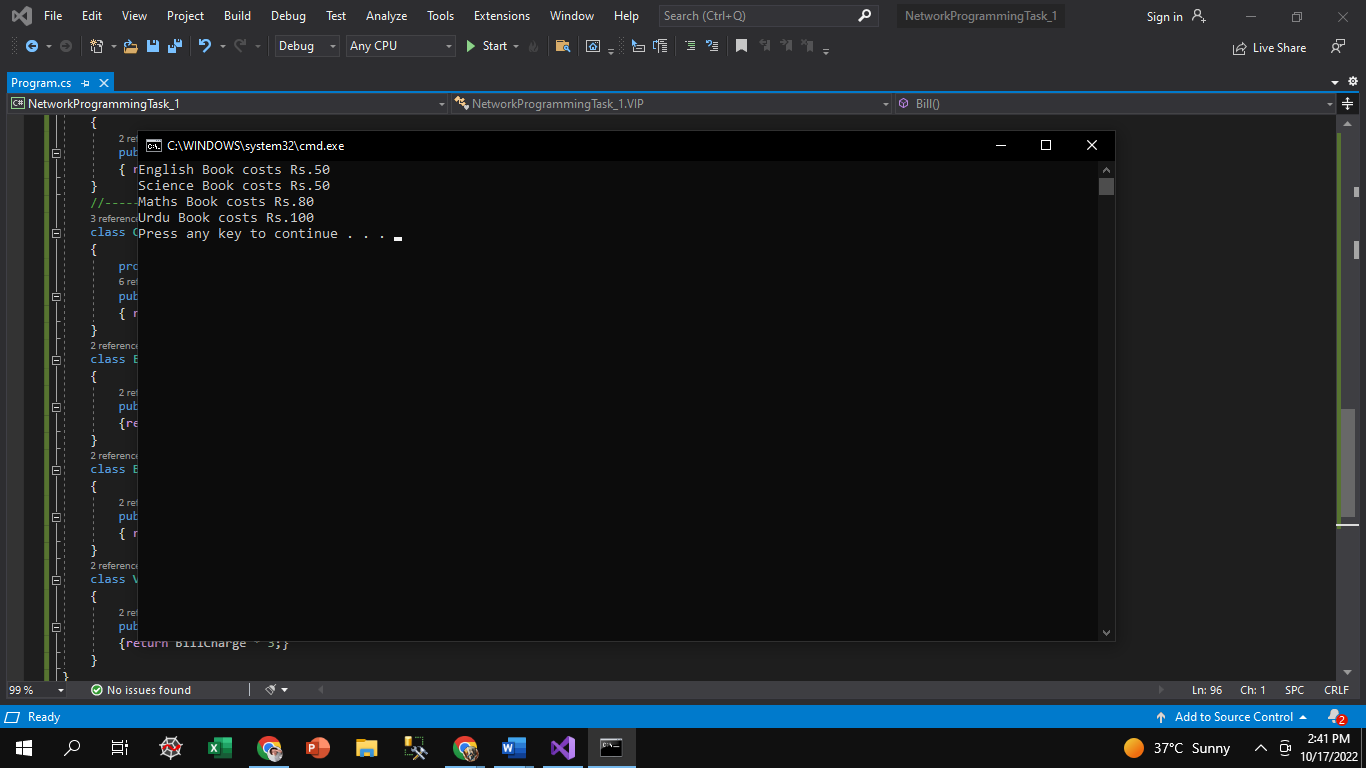
public override double Bill()

{return BillCharge \* 3;}

}

}

***Output:***



***Lab # 2: Client Server Connection***

**Server Code:**

Console.WriteLine("##### Server Side #####");

IPAddress ip = IPAddress.Loopback;

Console.WriteLine("Your IP is: " + ip);

IPEndPoint ep = new IPEndPoint(ip, 2023);

Socket svr = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

svr.Bind(ep);

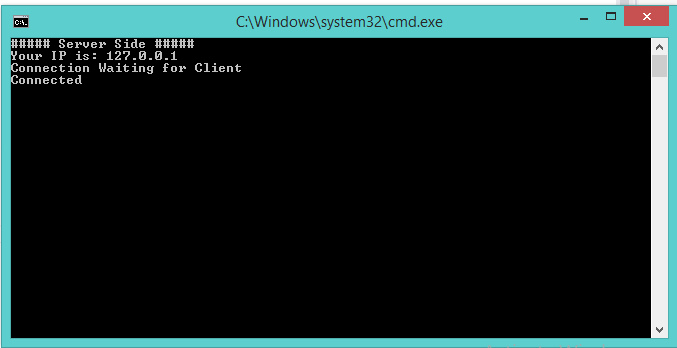
svr.Listen(1);

Console.WriteLine("Connection Waiting for Client ");

svr.Accept();

Console.WriteLine("Connected ");

**Output:**



**Client Code:**

Console.WriteLine("##### Client Side #####" );

IPAddress ip = IPAddress.Loopback;

Console.WriteLine("Your IP is: " + ip);

IPEndPoint ep = new IPEndPoint(ip,2023);

Socket svr = new Socket(ip.AddressFamily,SocketType.Stream,ProtocolType.Tcp);

svr.Connect(ep);

Console.WriteLine("Connection in progress ");

**Output:**



***Lab # 3: Client Server Communication***

**Server Code:**

Console.WriteLine("##### Server Side #####");

IPAddress ip = IPAddress.Loopback;

Console.WriteLine("Your IP is: " + ip);

IPEndPoint ep = new IPEndPoint(ip, 2023);

Socket svr = new Socket(ip.AddressFamily, SocketType.Stream, ProtocolType.Tcp);

svr.Bind(ep);

svr.Listen(1);

Console.WriteLine("Connection Waiting for Client ");

svr.Accept();

Console.WriteLine("Connected ");

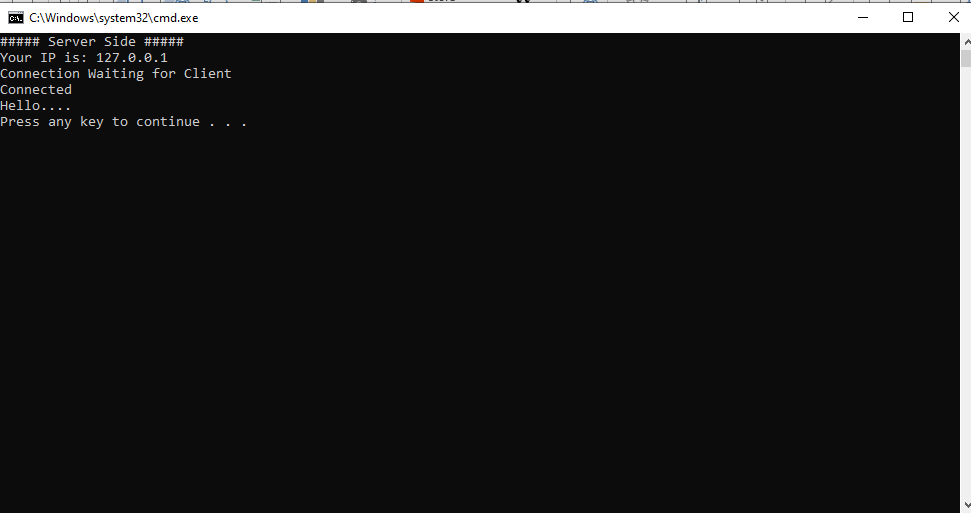
byte[] bt = new byte[100];

Socket sv = svr.Accept();

sv.Receive(bt);

Console.WriteLine(Encoding.ASCII.GetString(bt));

**Output:**



**Client Code:**

Console.WriteLine("##### Client Side #####" );

IPAddress ip = IPAddress.Loopback;

Console.WriteLine("Your IP is: " + ip);

IPEndPoint ep = new IPEndPoint(ip,2023);

Socket sv = new Socket(ip.AddressFamily,SocketType.Stream,ProtocolType.Tcp);

sv.Connect(ep);

Console.WriteLine("Connection in progress ");

string st = "Hello....";

sv.Send(Encoding.ASCII.GetBytes(st));

Console.ReadKey()

**Output:**



**Lab # 4: Client Server Communication using TCP (WinForms)**

**Server Code:**

public partial class Form1 : Form

{ public Form1()

{ InitializeComponent(); }

List<TcpClient> clients = new List<TcpClient>();

TcpListener server;

private void button2\_Click(object sender, EventArgs e)

{ CheckForIllegalCrossThreadCalls = false;

server = new TcpListener(IPAddress.Loopback, 8001);

server.Start(10);

textBox1.AppendText("Server Started!");

Thread t2 = new Thread(AcceptClient);

t2.Start(); }

public void AcceptClient()

{ while (true)

{ TcpClient c = server.AcceptTcpClient();

clients.Add(c);

Thread t = new Thread(asd => ReadMessage(c));

t.Start(); } }

public void ReadMessage(TcpClient client)

{ while (true)

{ NetworkStream stream = client.GetStream();

StreamReader sdr = new StreamReader(stream);

string msg = sdr.ReadLine();

textBox2.AppendText(Environment.NewLine);

textBox2.AppendText("Client: " + msg); } }

private void button3\_Click(object sender, EventArgs e)

{ foreach (var item in clients)

{ textBox1.AppendText(Environment.NewLine);

textBox1.AppendText("Me: " + textBox3.Text);

NetworkStream stream = item.GetStream();

StreamWriter sdr = new StreamWriter(stream);

sdr.WriteLine(textBox3.Text);

sdr.Flush(); } } }

}

**Client Code:**

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Threading;

using System.Windows.Forms;

namespace NP\_Assignment1

{ public partial class Form1 : Form

{ public Form1()

{ InitializeComponent(); }

TcpClient client = new TcpClient();

private void button1\_Click(object sender, EventArgs e)

{ CheckForIllegalCrossThreadCalls = false;

IPEndPoint point = new IPEndPoint(IPAddress.Loopback, 8002);

client = new TcpClient(point);

client.Connect(IPAddress.Loopback, 8001);

textBox2.AppendText("Client Connected");

Thread t = new Thread(ReadMessage);

t.Start(); }

public void ReadMessage()

{ while (true)

{ NetworkStream stream = client.GetStream();

StreamReader sdr = new StreamReader(stream);

string msg = sdr.ReadLine();

textBox2.AppendText(Environment.NewLine);

textBox2.AppendText("Server: " + msg); } }

private void button3\_Click(object sender, EventArgs e)

{ NetworkStream stream = client.GetStream();

StreamWriter sdr = new StreamWriter(stream);

sdr.WriteLine(textBox3.Text);

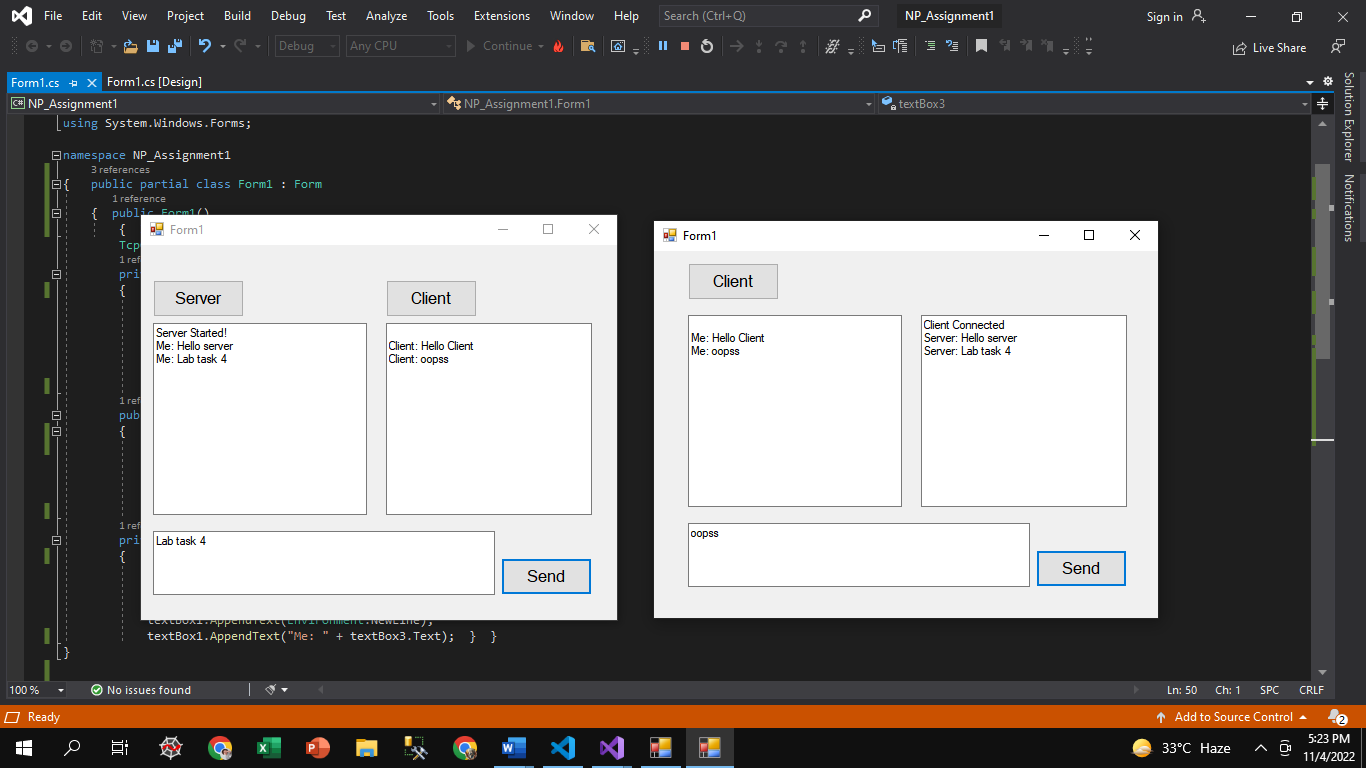
sdr.Flush();

textBox1.AppendText(Environment.NewLine);

textBox1.AppendText("Me: " + textBox3.Text); } }

}

**Output:**



**Lab # 5: Chapter#3 (3.1 & 3.2) Listings**

* *Listing 3.1: The AddressSample.cs program*

**Code:**

using System;

using System.Net;

namespace ConsoleApp69

{ class Program

{ static void Main(string[] args)

{ IPAddress test1 = IPAddress.Parse("192.168.18.11");

IPAddress test2 = IPAddress.Loopback;

IPAddress test3 = IPAddress.Broadcast;

IPAddress test4 = IPAddress.Any;

IPAddress test5 = IPAddress.None;

IPHostEntry ihe = Dns.GetHostEntry(Dns.GetHostName());

IPAddress myself = ihe.AddressList[0];

if (IPAddress.IsLoopback(test2)) { Console.WriteLine("The Loopback address is: {0}", test2.ToString()); }

else { Console.WriteLine("Error obtaining the loopback address"); }

Console.WriteLine("The Local IP address is: {0}", myself.ToString());

if (myself == test2) { Console.WriteLine("The loopback address is the same as local address."); }

else { Console.WriteLine("The loopback address is not the local address."); }

Console.WriteLine("The test address is: {0}", test1.ToString());

Console.WriteLine("Broadcast address: {0}",test3.ToString());

Console.WriteLine("The ANY address is: {0}", test4.ToString());

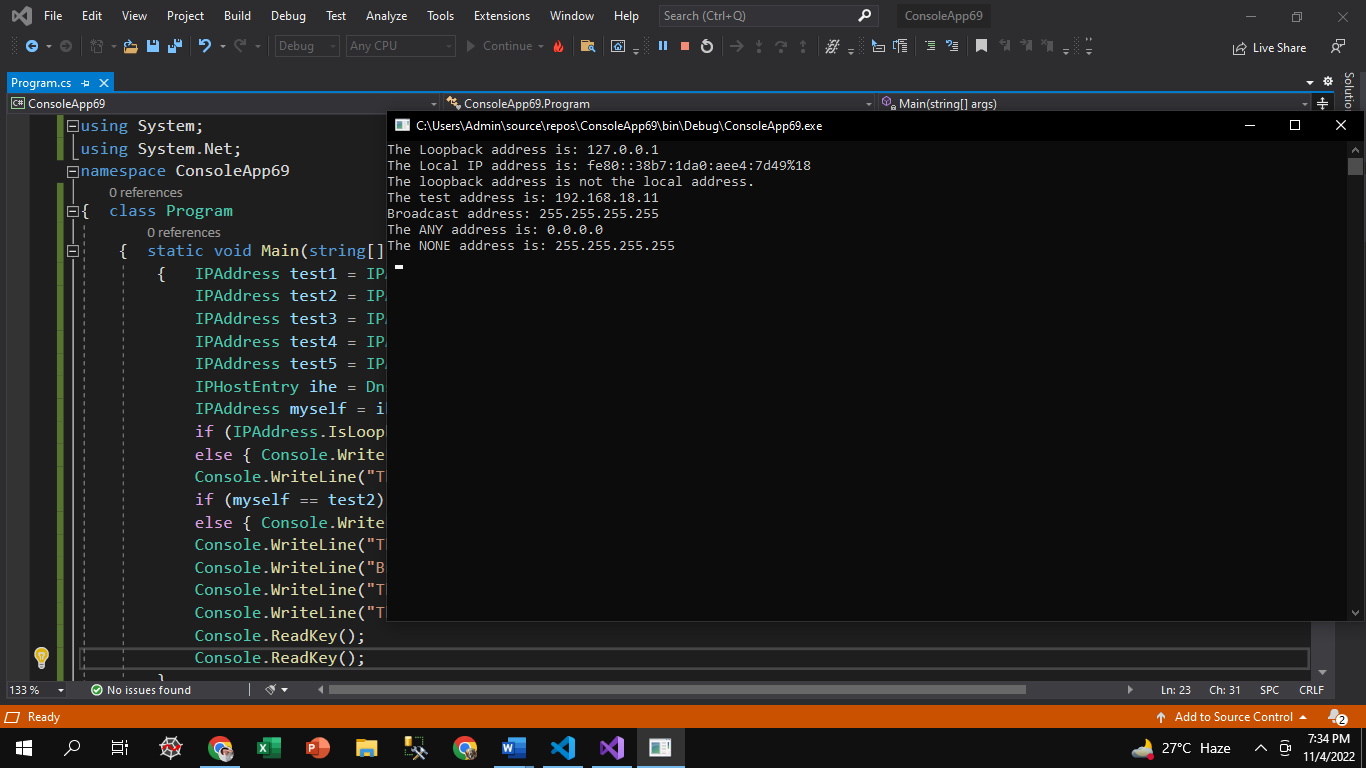
Console.WriteLine("The NONE address is: {0}",test5.ToString());

Console.ReadKey();

}

}

}

**Output:** 

* Listing 3.2: The IPEndPointSample.cs program

**Code:**

IPAddress test1 = IPAddress.Parse("192.168.1.1");

IPEndPoint ie = new IPEndPoint(test1, 8000);

Console.WriteLine("The IPEndPoint is: {0}", ie.ToString());

Console.WriteLine("The AddressFamily is: {0}", ie.AddressFamily);

Console.WriteLine("The address is: {0}, and the port is: {1}", ie.Address, ie.Port);

Console.WriteLine("The min port number is: {0}",IPEndPoint.MinPort);

Console.WriteLine("The max port number is: {0}",IPEndPoint.MaxPort);

ie.Port = 80;

Console.WriteLine("The changed IPEndPoint value is: {0} ", ie.ToString());

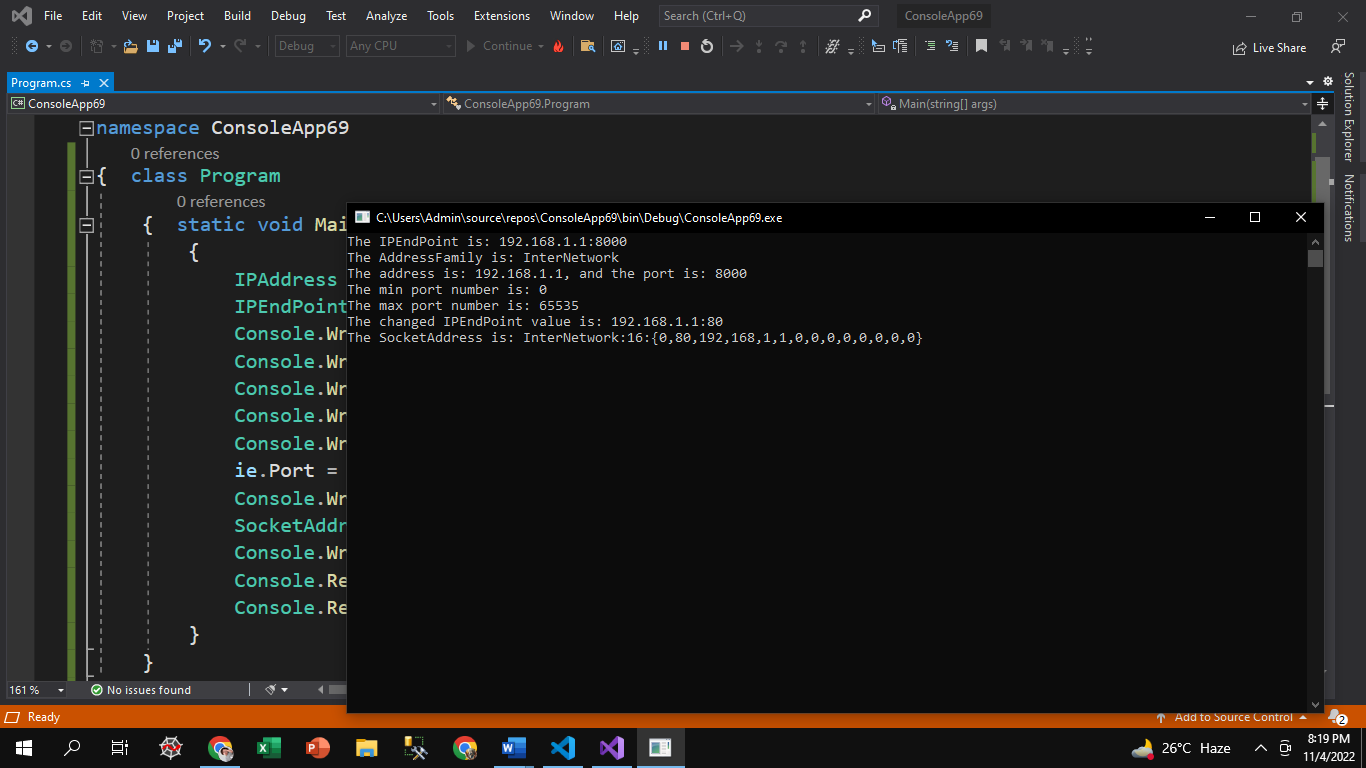
SocketAddress sa = ie.Serialize();

Console.WriteLine("The SocketAddress is: {0}",sa.ToString());

Console.ReadKey();

Console.ReadKey();

**Output:**



**Lab # 6: Chapter#3 (3.3 & 3.4) & Chapter#5 (5.1 & 5.2) Listings**

* Listing 3.3: The SockProp.cs sample socket properties program

**Code:**

IPAddress ia = IPAddress.Parse("127.0.0.1");

IPEndPoint ie = new IPEndPoint(ia, 8000);

Socket test = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

Console.WriteLine("AddressFamily: {0}",test.AddressFamily);

Console.WriteLine("SocketType: {0}",test.SocketType);

Console.WriteLine("ProtocolType: {0}",test.ProtocolType);

Console.WriteLine("Blocking: {0}", test.Blocking);

test.Blocking = false;

Console.WriteLine("new Blocking: {0}", test.Blocking);

Console.WriteLine("Connected: {0}", test.Connected);

test.Bind(ie);

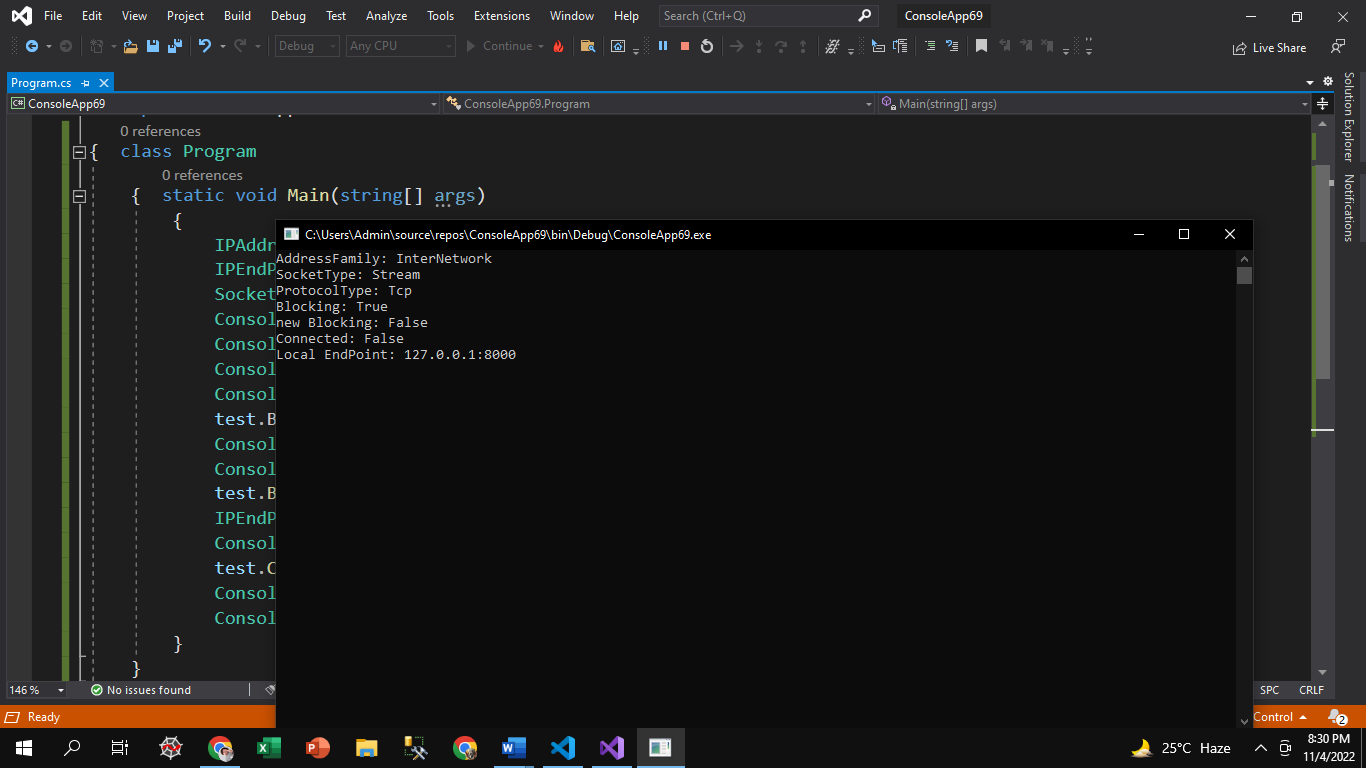
IPEndPoint iep = (IPEndPoint)test.LocalEndPoint;

Console.WriteLine("Local EndPoint: {0}",iep.ToString());

test.Close();

Console.ReadKey();

Console.ReadKey();

**Output:** 

**Listing 3.4: The SocketExcept.cs program**

**Code:**

class SocketExcept

{ public static void Main()

{ IPAddress host = IPAddress.Parse("192.168.1.1");

IPEndPoint hostep = new IPEndPoint(host, 8000);

Socket sock = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

try

{ sock.Connect(hostep); }

catch (SocketException e)

{ Console.WriteLine("Problem connecting to host");

Console.WriteLine(e.ToString());

sock.Close();

return; }

try

{ sock.Send(Encoding.ASCII.GetBytes("testing")); }

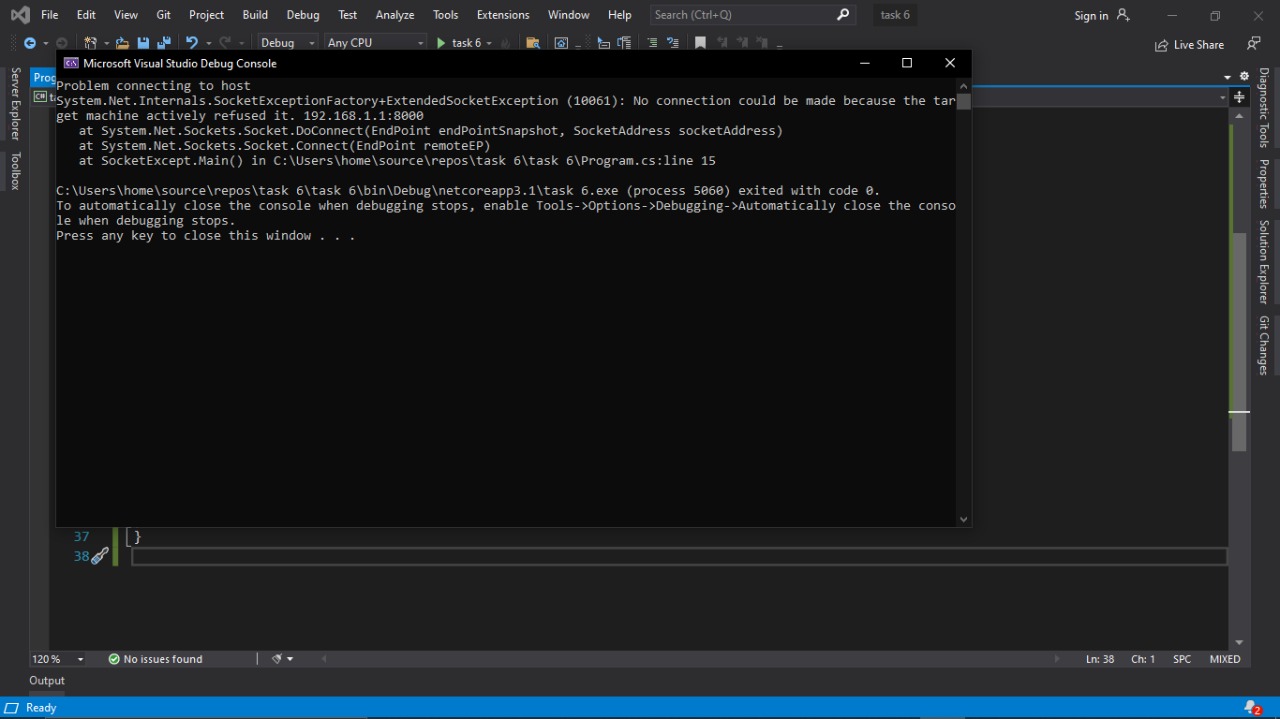
catch (SocketException e)

{ Console.WriteLine("Problem sending data");

Console.WriteLine(e.ToString());

sock.Close();

return;

**Output:** 

* **Listing 5.1 & 5.2: The SimpleTcpSrvr.cs program & The SimpleTcpClient.cs program**

**Server Code:**

using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

namespace ConsoleApp70

{ class Program

{ static void Main(string[] args)

{ int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any,9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

newsock.Bind(ipep);

newsock.Listen(10);

Console.WriteLine("Waiting for a client...");

Socket client = newsock.Accept();

IPEndPoint clientep =(IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port {1}",clientep.Address, clientep.Port);

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

client.Send(data, data.Length,SocketFlags.None);

while (true)

{ data = new byte[1024];

recv = client.Receive(data);

if (recv == 0)

break;

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

client.Send(data, recv, SocketFlags.None);

}

Console.WriteLine("Disconnected from {0}",clientep.Address);

client.Close();

newsock.Close();

}

}

}

**Client Code:**

using System;

using System.Text;

using System.Net;

using System.Net.Sockets;

namespace ConsoleApp71

{ class Program

{ static void Main(string[] args)

{ byte[] data = new byte[1024];

string input, stringData;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

try

{ server.Connect(ipep); }

catch (SocketException e)

{ Console.WriteLine("Unable to connect to server.");

Console.WriteLine(e.ToString());

return; }

int recv = server.Receive(data);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

while (true)

{ input = Console.ReadLine();

if (input == "exit")

break;

server.Send(Encoding.ASCII.GetBytes(input));

data = new byte[1024];

recv = server.Receive(data);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData); }

Console.WriteLine("Disconnecting from server...");

server.Shutdown(SocketShutdown.Both);

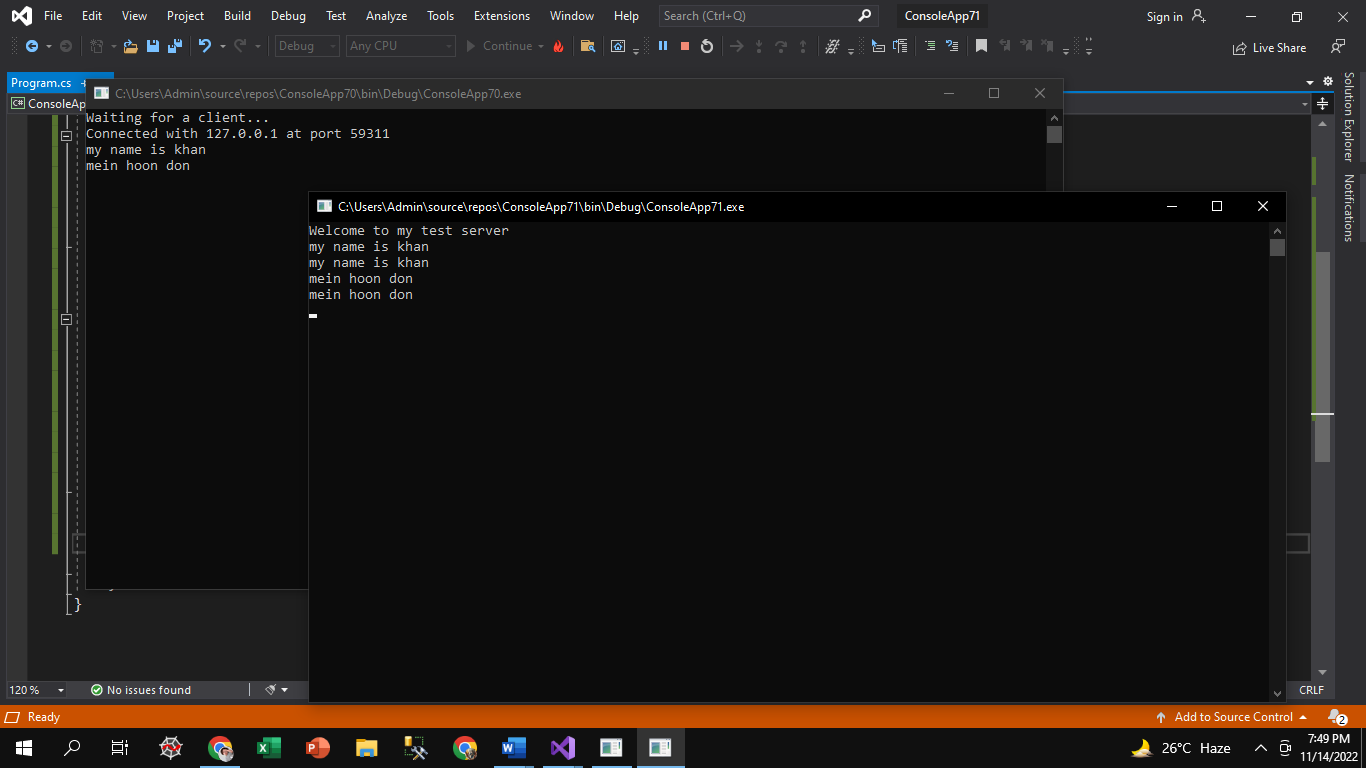
server.Close();

}

}

}

**Output:**



**Lab # 7: Chapter#5 (5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.10 & 5.11) Listings**

* **Listing 5.3 & 5.4: The BadTcpSrvr.cs program & The BadTcpClient.cs program**

**Server Code:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class BadTcpSrvr

{

public static void Main()

{

int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

newsock.Bind(ipep);

newsock.Listen(10);

Console.WriteLine("Waiting for a client...");

Socket client = newsock.Accept();

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

client.Send(data, data.Length, SocketFlags.None);

IPEndPoint newclient = (IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port {1}", newclient.Address, newclient.Port);

for (int i = 0; i < 5; i++)

{

recv = client.Receive(data);

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

}

Console.WriteLine("Disconnecting from {0}", newclient.Address);

client.Close();

newsock.Close();

Console.ReadKey();

Console.ReadKey();

}

}

**Client Code:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class BadTcpClient

{

public static void Main()

{ byte[] data = new byte[1024];

string stringData;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

try

{ server.Connect(ipep); }

catch (SocketException e)

{ Console.WriteLine("Unable to connect to server.");

Console.WriteLine(e.ToString());

return; }

int recv = server.Receive(data);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

server.Send(Encoding.ASCII.GetBytes("message 1"));

server.Send(Encoding.ASCII.GetBytes("message 2"));

server.Send(Encoding.ASCII.GetBytes("message 3"));

server.Send(Encoding.ASCII.GetBytes("message 4"));

server.Send(Encoding.ASCII.GetBytes("message 5"));

Console.WriteLine("Disconnecting from server...");

server.Shutdown(SocketShutdown.Both);

server.Close();

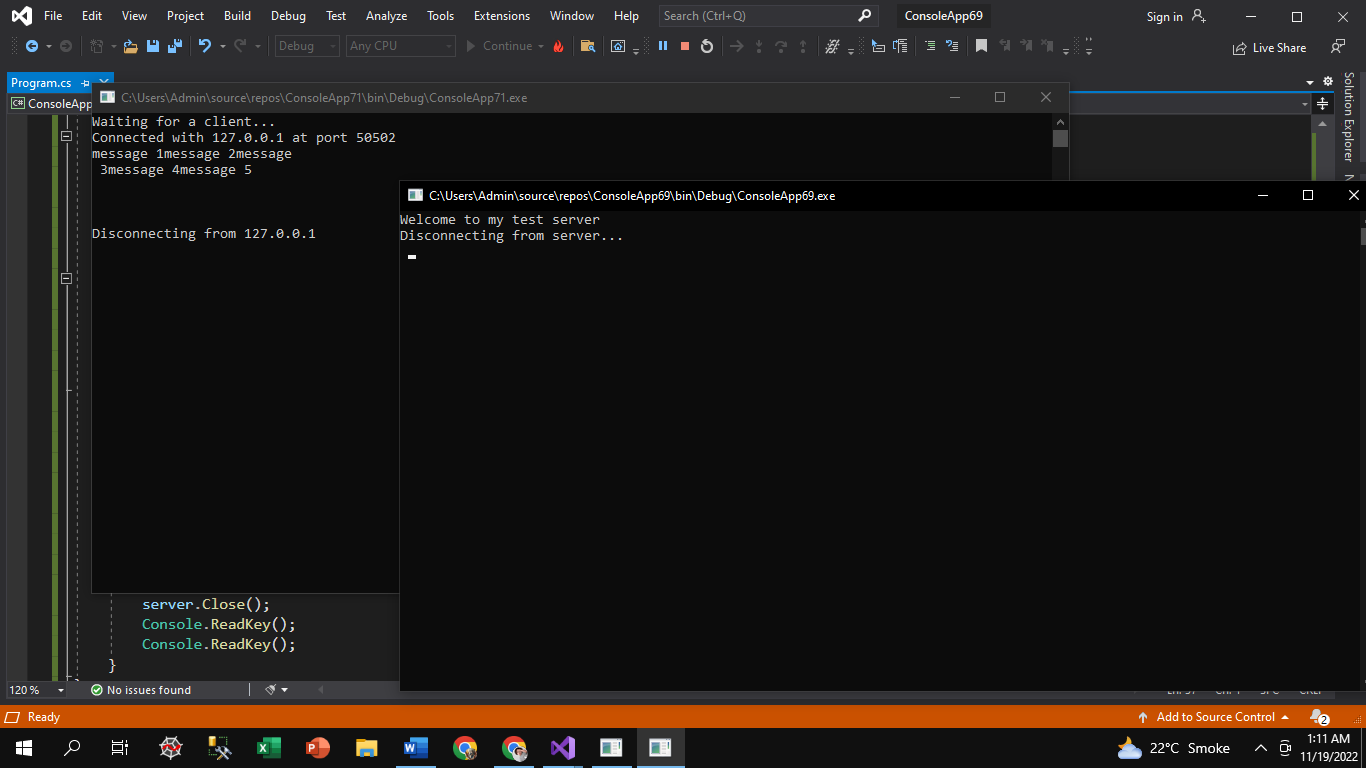
Console.ReadKey();

Console.ReadKey();

}

}

**Output:**



* **Listing 5.5 & 5.6: The FixedTcpSrvr.cs program& The FixedTcpClient.cs program**

**Server Code:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class FixedTcpSrvr

{ private static int SendData(Socket s, byte[] data)

{ int total = 0;

int size = data.Length;

int dataleft = size;

int sent;

while (total < size)

{ sent = s.Send(data, total, dataleft, SocketFlags.None);

total += sent;

dataleft -= sent; }

return total; }

private static byte[] ReceiveData(Socket s, int size)

{ int total = 0;

int dataleft = size;

byte[] data = new byte[size];

int recv;

while (total < size)

{ recv = s.Receive(data, total, dataleft, 0);

if (recv == 0)

{ data = Encoding.ASCII.GetBytes("exit");

break; }

total += recv;

dataleft -= recv; }

return data; }

public static void Main()

{ byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,SocketType.Stream, ProtocolType.Tcp);

newsock.Bind(ipep);

newsock.Listen(10);

Console.WriteLine("Waiting for a client...");

Socket client = newsock.Accept();

IPEndPoint newclient = (IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port {1}", newclient.Address, newclient.Port);

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

int sent = SendData(client, data);

for (int i = 0; i < 5; i++)

{ data = ReceiveData(client, 9);

Console.WriteLine(Encoding.ASCII.GetString(data)); }

Console.WriteLine("Disconnected from {0}", newclient.Address);

client.Close();

newsock.Close();

Console.ReadKey();

Console.ReadKey();

}

}

**Client Code:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class FixedTcpClient

{ private static int SendData(Socket s, byte[] data)

{ int total = 0;

int size = data.Length;

int dataleft = size;

int sent;

while (total < size)

{ sent = s.Send(data, total, dataleft, SocketFlags.None);

total += sent;

dataleft -= sent; }

return total; }

private static byte[] ReceiveData(Socket s, int size)

{ int total = 0;

int dataleft = size;

byte[] data = new byte[size];

int recv;

while (total < size)

{ recv = s.Receive(data, total, dataleft, 0);

if (recv == 0)

{ data = Encoding.ASCII.GetBytes("exit ");

break; }

total += recv;

dataleft -= recv; }

return data; }

public static void Main()

{ byte[] data = new byte[1024];

int sent;

IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

try

{ server.Connect(ipep); }

catch (SocketException e)

{ Console.WriteLine("Unable to connect to server.");

Console.WriteLine(e.ToString());

return; }

int recv = server.Receive(data);

string stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

sent = SendData(server, Encoding.ASCII.GetBytes("message 1"));

sent = SendData(server, Encoding.ASCII.GetBytes("message 2"));

sent = SendData(server, Encoding.ASCII.GetBytes("message 3"));

sent = SendData(server, Encoding.ASCII.GetBytes("message 4"));

sent = SendData(server, Encoding.ASCII.GetBytes("message 5"));

Console.WriteLine("Disconnecting from server...");

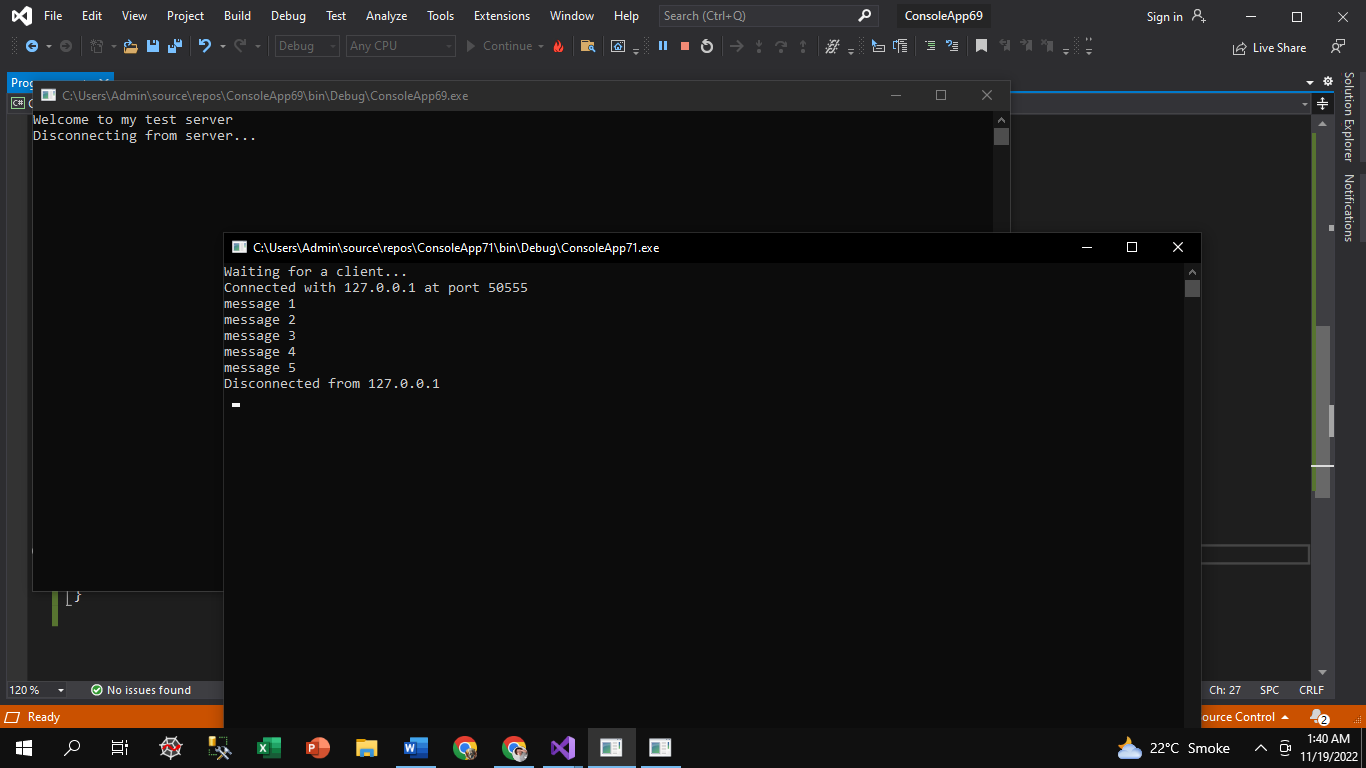
server.Shutdown(SocketShutdown.Both);

server.Close();

}

}

**Output:**



* **Listing 5.7 & 5.8: The VarTcpSrvr.cs program & The VarTcpClient.cs program**

**Server Code:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class VarTcpSrvr

{ private static int SendVarData(Socket s, byte[] data)

{ int total = 0;

int size = data.Length;

int dataleft = size;

int sent;

byte[] datasize = new byte[4];

datasize = BitConverter.GetBytes(size);

sent = s.Send(datasize);

while (total < size)

{ sent = s.Send(data, total, dataleft, SocketFlags.None);

total += sent;

dataleft -= sent; }

return total; }

private static byte[] ReceiveVarData(Socket s)

{ int total = 0;

int recv;

byte[] datasize = new byte[4];

recv = s.Receive(datasize, 0, 4, 0);

int size = BitConverter.ToInt32(datasize, 0);

int dataleft = size;

byte[] data = new byte[size];

while (total < size)

{ recv = s.Receive(data, total, dataleft, 0);

if (recv == 0)

{ data = Encoding.ASCII.GetBytes("exit ");

break; }

total += recv;

dataleft -= recv;

} return data; }

public static void Main()

{ byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

newsock.Bind(ipep);

newsock.Listen(10);

Console.WriteLine("Waiting for a client...");

Socket client = newsock.Accept();

IPEndPoint newclient = (IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port {1}",

newclient.Address, newclient.Port);

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

int sent = SendVarData(client, data);

for (int i = 0; i < 5; i++)

{ data = ReceiveVarData(client);

Console.WriteLine(Encoding.ASCII.GetString(data)); }

Console.WriteLine("Disconnected from {0}", newclient.Address);

client.Close();

newsock.Close();

Console.ReadKey(); }

}

**Client Code:**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class VarTcpClient

{ private static int SendVarData(Socket s, byte[] data)

{ int total = 0;

int size = data.Length;

int dataleft = size;

int sent;

byte[] datasize = new byte[4];

datasize = BitConverter.GetBytes(size);

sent = s.Send(datasize);

while (total < size)

{ sent = s.Send(data, total, dataleft, SocketFlags.None);

total += sent;

dataleft -= sent; }

return total; }

private static byte[] ReceiveVarData(Socket s)

{ int total = 0;

int recv;

byte[] datasize = new byte[4];

recv = s.Receive(datasize, 0, 4, 0);

int size = BitConverter.ToInt32(datasize, 0);

int dataleft = size;

byte[] data = new byte[size];

while (total < size)

{ recv = s.Receive(data, total, dataleft, 0);

if (recv == 0)

{ data = Encoding.ASCII.GetBytes("exit ");

break; }

total += recv;

dataleft -= recv; }

return data; }

public static void Main()

{ byte[] data = new byte[1024];

int sent;

IPEndPoint ipep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

try

{ server.Connect(ipep); }

catch (SocketException e)

{ Console.WriteLine("Unable to connect to server.");

Console.WriteLine(e.ToString());

return; }

data = ReceiveVarData(server);

string stringData = Encoding.ASCII.GetString(data);

Console.WriteLine(stringData);

string message1 = "This is the first test";

string message2 = "A short test";

string message3 = "This string is an even longer test. The quick brown Â\_ fox jumps over the lazy dog.";

string message4 = "a";

string message5 = "The last test";

sent = SendVarData(server, Encoding.ASCII.GetBytes(message1));

sent = SendVarData(server, Encoding.ASCII.GetBytes(message2));

sent = SendVarData(server, Encoding.ASCII.GetBytes(message3));

sent = SendVarData(server, Encoding.ASCII.GetBytes(message4));

sent = SendVarData(server, Encoding.ASCII.GetBytes(message5));

Console.WriteLine("Disconnecting from server...");

server.Shutdown(SocketShutdown.Both);

server.Close();

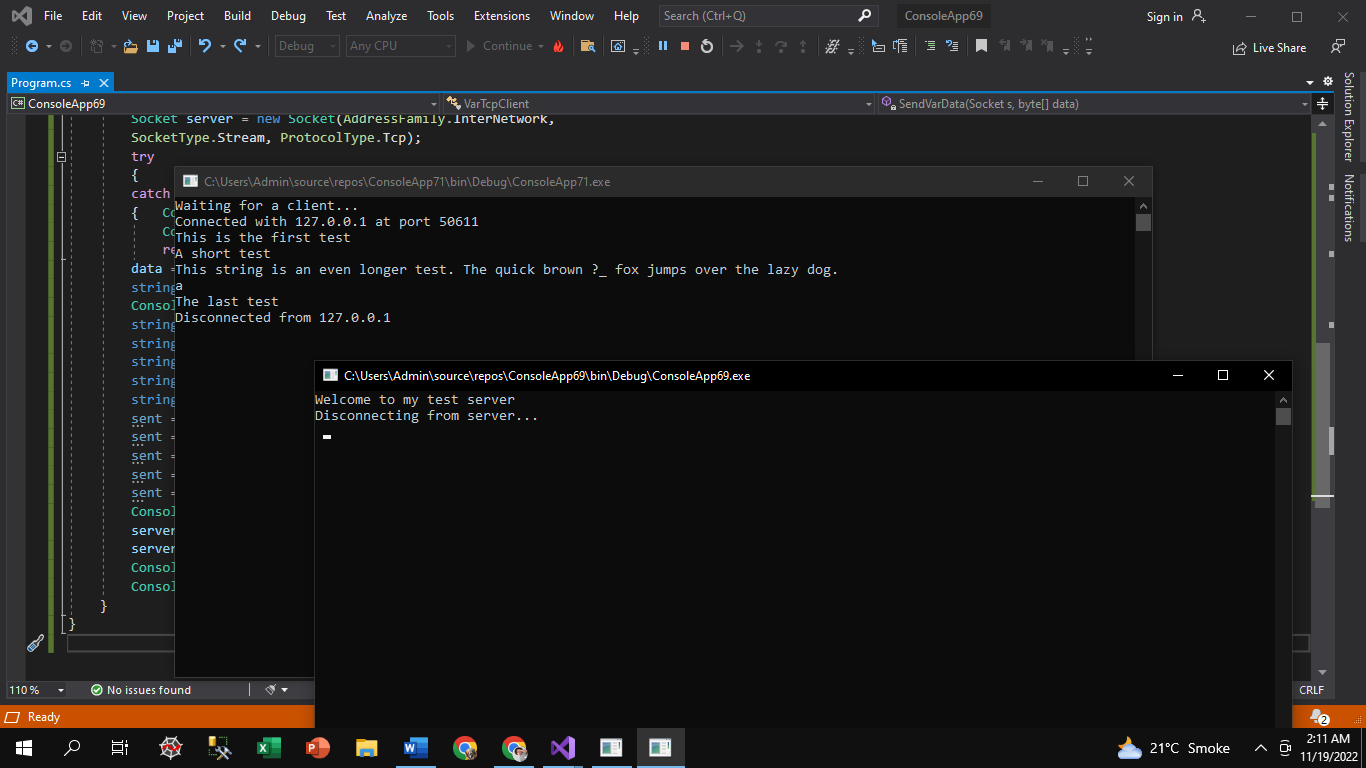
Console.ReadKey();

Console.ReadKey();

}

}

**Output:**



* **Listing 5.10 & 5.11: The StreamTcpSrvr.cs program & The StreamTcpClient.cs program**

**Server Code:**

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

class StreamTcpSrvr

{ public static void Main()

{ string data;

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

newsock.Bind(ipep);

newsock.Listen(10);

Console.WriteLine("Waiting for a client...");

Socket client = newsock.Accept();

IPEndPoint newclient = (IPEndPoint)client.RemoteEndPoint;

Console.WriteLine("Connected with {0} at port {1}",

newclient.Address, newclient.Port);

NetworkStream ns = new NetworkStream(client);

StreamReader sr = new StreamReader(ns);

StreamWriter sw = new StreamWriter(ns);

string welcome = "Welcome to my test server";

sw.WriteLine(welcome);

sw.Flush();

while (true) { try

{ data = sr.ReadLine(); }

catch (IOException)

{ break; }

Console.WriteLine(data);

sw.WriteLine(data);

sw.Flush(); }

Console.WriteLine("Disconnected from {0}", newclient.Address);

sw.Close();

sr.Close();

ns.Close();

Console.ReadKey();

Console.ReadKey();

}

}

**Client Code:**

using System;

using System.IO;

using System.Net;

using System.Net.Sockets;

using System.Text;

class StreamTcpClient

{ public static void Main()

{ string data;

string input;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Stream, ProtocolType.Tcp);

try

{ server.Connect(ipep); }

catch (SocketException e)

{ Console.WriteLine("Unable to connect to server.");

Console.WriteLine(e.ToString());

return; }

NetworkStream ns = new NetworkStream(server);

StreamReader sr = new StreamReader(ns);

StreamWriter sw = new StreamWriter(ns);

data = sr.ReadLine();

Console.WriteLine(data);

while (true)

{ input = Console.ReadLine();

if (input == "exit")

break;

sw.WriteLine(input);

sw.Flush();

data = sr.ReadLine();

Console.WriteLine(data); }

Console.WriteLine("Disconnecting from server...");

sr.Close();

sw.Close();

ns.Close();

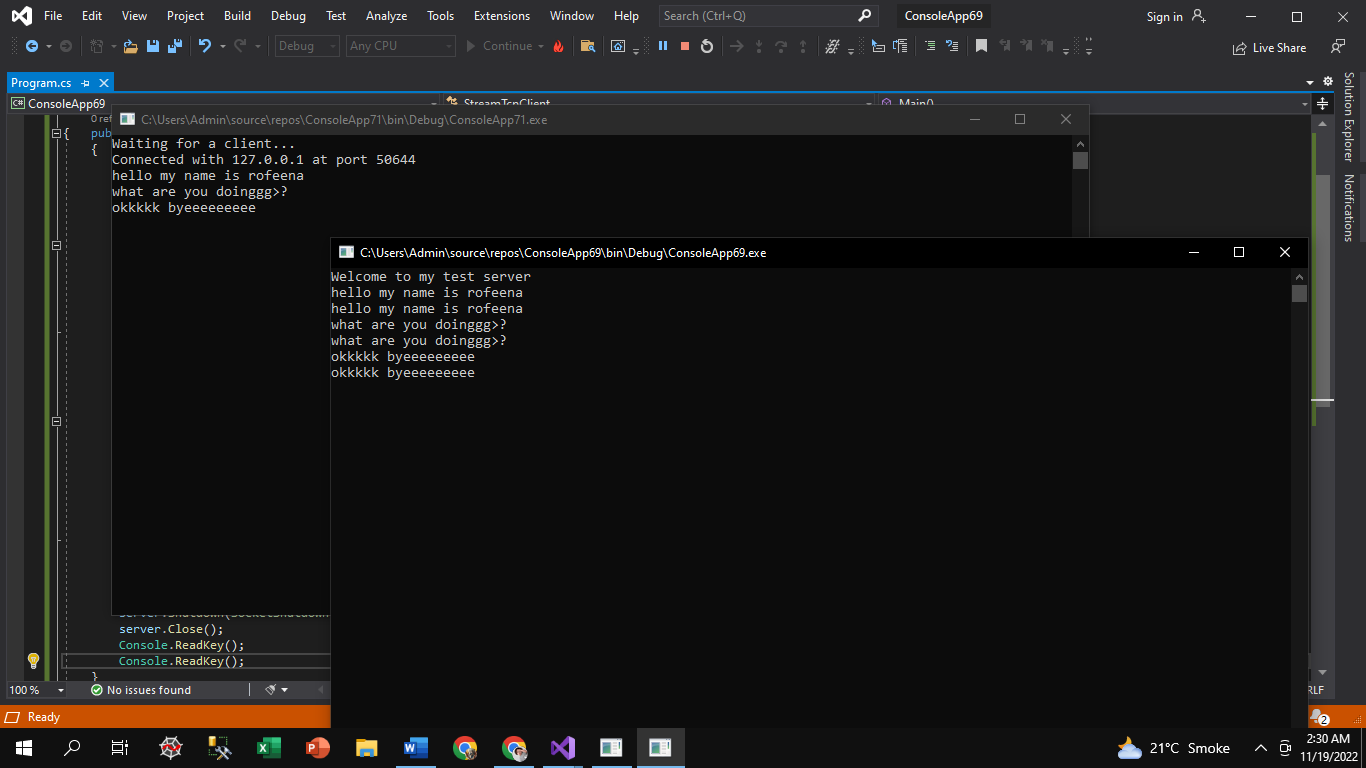
server.Shutdown(SocketShutdown.Both);

server.Close();

}

}

**Output:**



**Lab # 8: Chapter#6 Listings**

* **Listing 6.1 & 6.2: The SimpleUdpServer.cs program & The SimpleUdpClient.cs program**

**Server code (6.1) :**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class SimpleUdpSrvr

{

public static void Main()

{

int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

newsock.Bind(ipep);

Console.WriteLine("Waiting for a client...");

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint Remote = (EndPoint)(sender);

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine("Message received from {0}:", Remote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

newsock.SendTo(data, data.Length, SocketFlags.None, Remote);

while (true)

{

data = new byte[1024];

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

newsock.SendTo(data, recv, SocketFlags.None, Remote);

}

}

}

**Client code (6.2) :**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class SimpleUdpClient

{

public static void Main()

{

byte[] data = new byte[1024];

string input, stringData;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

string welcome = "Hello, are you there?";

data = Encoding.ASCII.GetBytes(welcome);

server.SendTo(data, data.Length, SocketFlags.None, ipep);

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint Remote = (EndPoint)sender;

data = new byte[1024];

int recv = server.ReceiveFrom(data, ref Remote);

Console.WriteLine("Message received from {0}:", Remote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

while (true)

{ input = Console.ReadLine();

if (input == "exit")

break;

server.SendTo(Encoding.ASCII.GetBytes(input), Remote);

data = new byte[1024];

recv = server.ReceiveFrom(data, ref Remote);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

}

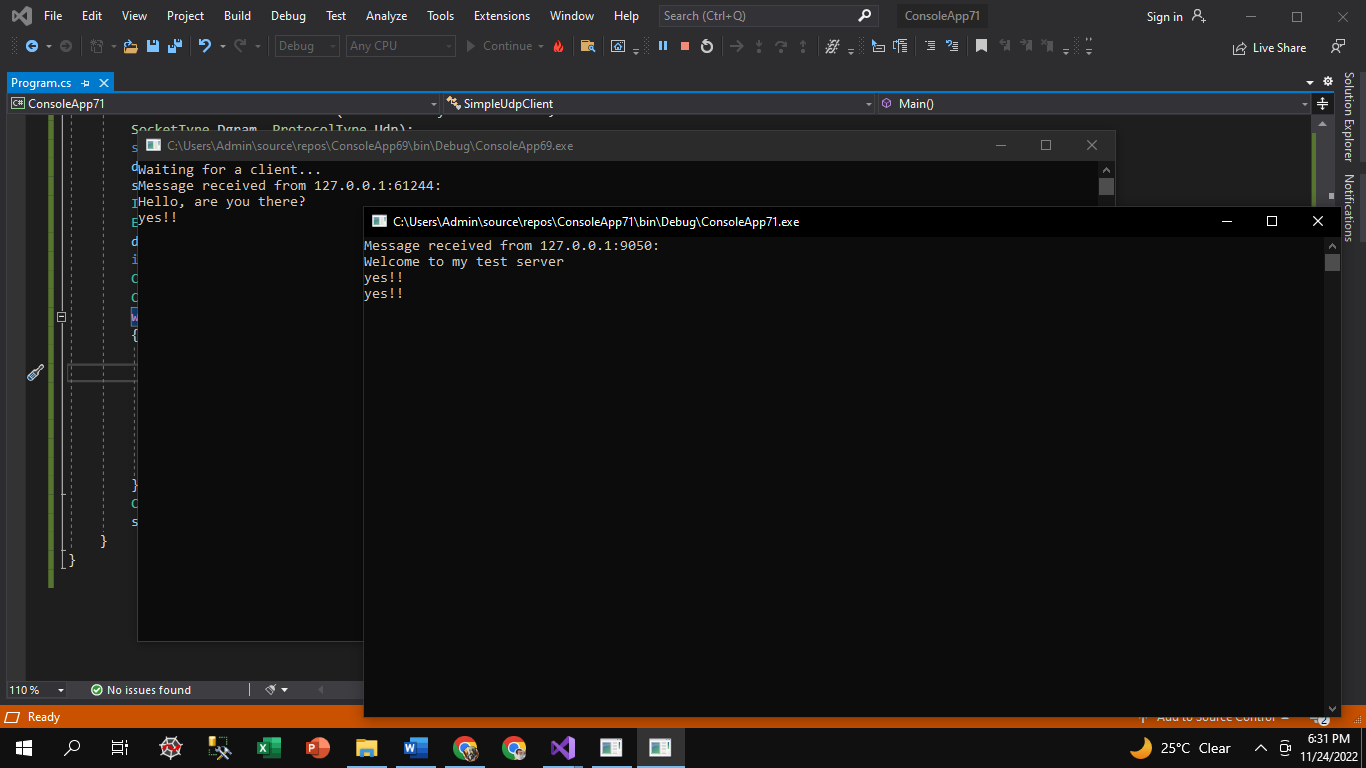
Console.WriteLine("Stopping client");

server.Close();

}

}

**Output:**



* **Listing 6.1 & 6.3: The SimpleUdpServer.cs program & The OddUdpClient.cs program**

**Server code (6.1) :**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class SimpleUdpSrvr

{

public static void Main()

{

int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

newsock.Bind(ipep);

Console.WriteLine("Waiting for a client...");

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint Remote = (EndPoint)(sender);

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine("Message received from {0}:", Remote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

newsock.SendTo(data, data.Length, SocketFlags.None, Remote);

while (true)

{

data = new byte[1024];

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

newsock.SendTo(data, recv, SocketFlags.None, Remote);

}

}

}

**Client code (6.3) :**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class OddUdpClient

{

public static void Main()

{

byte[] data = new byte[1024];

string input, stringData;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

server.Connect(ipep);

string welcome = "Hello, are you there?";

data = Encoding.ASCII.GetBytes(welcome);

server.Send(data);

data = new byte[1024];

int recv = server.Receive(data);

Console.WriteLine("Message received from {0}:", ipep.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

while (true)

{

input = Console.ReadLine();

if (input == "exit")

break;

server.Send(Encoding.ASCII.GetBytes(input));

data = new byte[1024];

recv = server.Receive(data);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

}

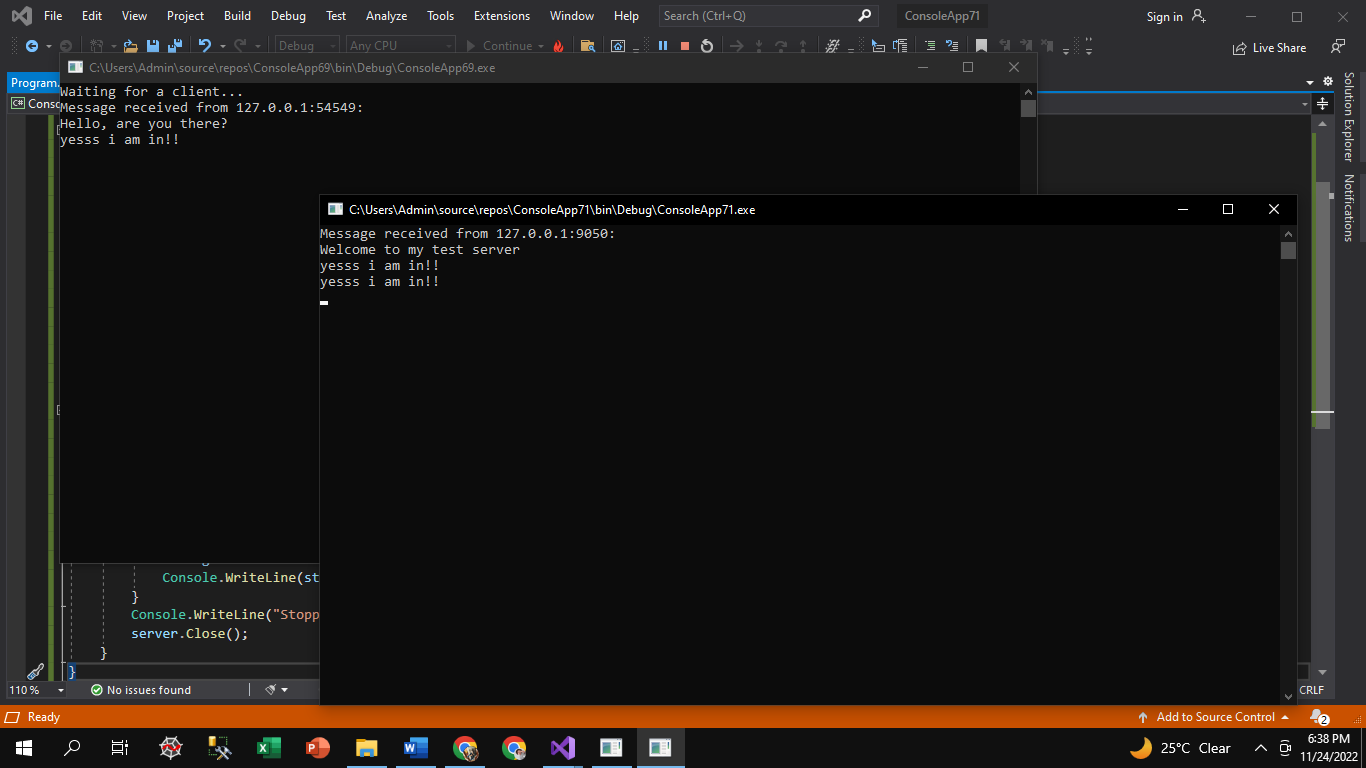
Console.WriteLine("Stopping client");

server.Close();

}

}

**Output:**



* **Listing 6.4 & 6.5: The TestUdpServer.cs program & The TestUdpClient.cs program**

**Server code (6.4):**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class TestUdpSrvr

{

public static void Main()

{

int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

newsock.Bind(ipep);

Console.WriteLine("Waiting for a client...");

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint tmpRemote = (EndPoint)(sender);

recv = newsock.ReceiveFrom(data, ref tmpRemote);

Console.WriteLine("Message received from {0}:", tmpRemote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

newsock.SendTo(data, data.Length, SocketFlags.None, tmpRemote);

for (int i = 0; i < 5; i++)

{

data = new byte[1024];

recv = newsock.ReceiveFrom(data, ref tmpRemote);

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

}

newsock.Close();

}

}

**Client code (6.5):**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class TestUdpClient

{

public static void Main()

{

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

string welcome = "Hello, are you there?";

data = Encoding.ASCII.GetBytes(welcome);

server.SendTo(data, data.Length, SocketFlags.None, ipep);

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint tmpRemote = (EndPoint)sender;

data = new byte[1024];

int recv = server.ReceiveFrom(data, ref tmpRemote);

Console.WriteLine("Message received from {0}:", tmpRemote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

server.SendTo(Encoding.ASCII.GetBytes("message 1"), tmpRemote);

server.SendTo(Encoding.ASCII.GetBytes("message 2"), tmpRemote);

server.SendTo(Encoding.ASCII.GetBytes("message 3"), tmpRemote);

server.SendTo(Encoding.ASCII.GetBytes("message 4"), tmpRemote);

server.SendTo(Encoding.ASCII.GetBytes("message 5"), tmpRemote);

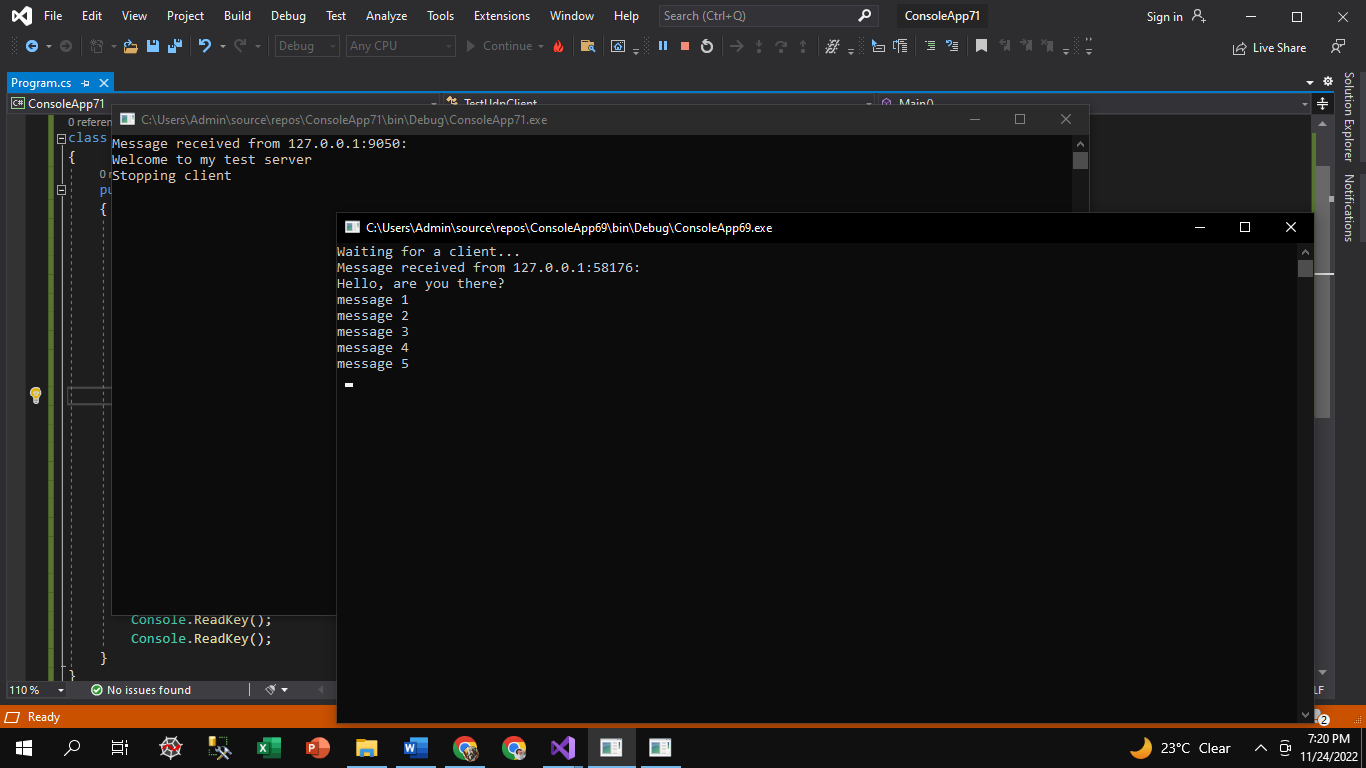
Console.WriteLine("Stopping client");

server.Close();

}

}

**Output:**



* **Listing 6.1 & 6.7: The SimpleUdpServer.cs program & The BetterUdpClient.cs program**

**Server code (6.1):**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class SimpleUdpSrvr

{

public static void Main()

{

int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

newsock.Bind(ipep);

Console.WriteLine("Waiting for a client...");

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint Remote = (EndPoint)(sender);

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine("Message received from {0}:", Remote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

newsock.SendTo(data, data.Length, SocketFlags.None, Remote);

while (true)

{

data = new byte[1024];

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

newsock.SendTo(data, recv, SocketFlags.None, Remote);

}

}

}

**Client code (6.7):**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class BetterdUdpClient

{

public static void Main()

{

byte[] data = new byte[30];

string input, stringData;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

string welcome = "Hello, are you there?";

data = Encoding.ASCII.GetBytes(welcome);

server.SendTo(data, data.Length, SocketFlags.None, ipep);

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint tmpRemote = (EndPoint)sender;

data = new byte[30];

int recv = server.ReceiveFrom(data, ref tmpRemote);

Console.WriteLine("Message received from {0}:", tmpRemote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

int i = 30;

while (true)

{

input = Console.ReadLine();

if (input == "exit")

break;

server.SendTo(Encoding.ASCII.GetBytes(input), tmpRemote);

data = new byte[i];

try

{

recv = server.ReceiveFrom(data, ref tmpRemote);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData);

}

catch (SocketException)

{

Console.WriteLine("WARNING: data lost, retry message.");

i += 10;

}

}

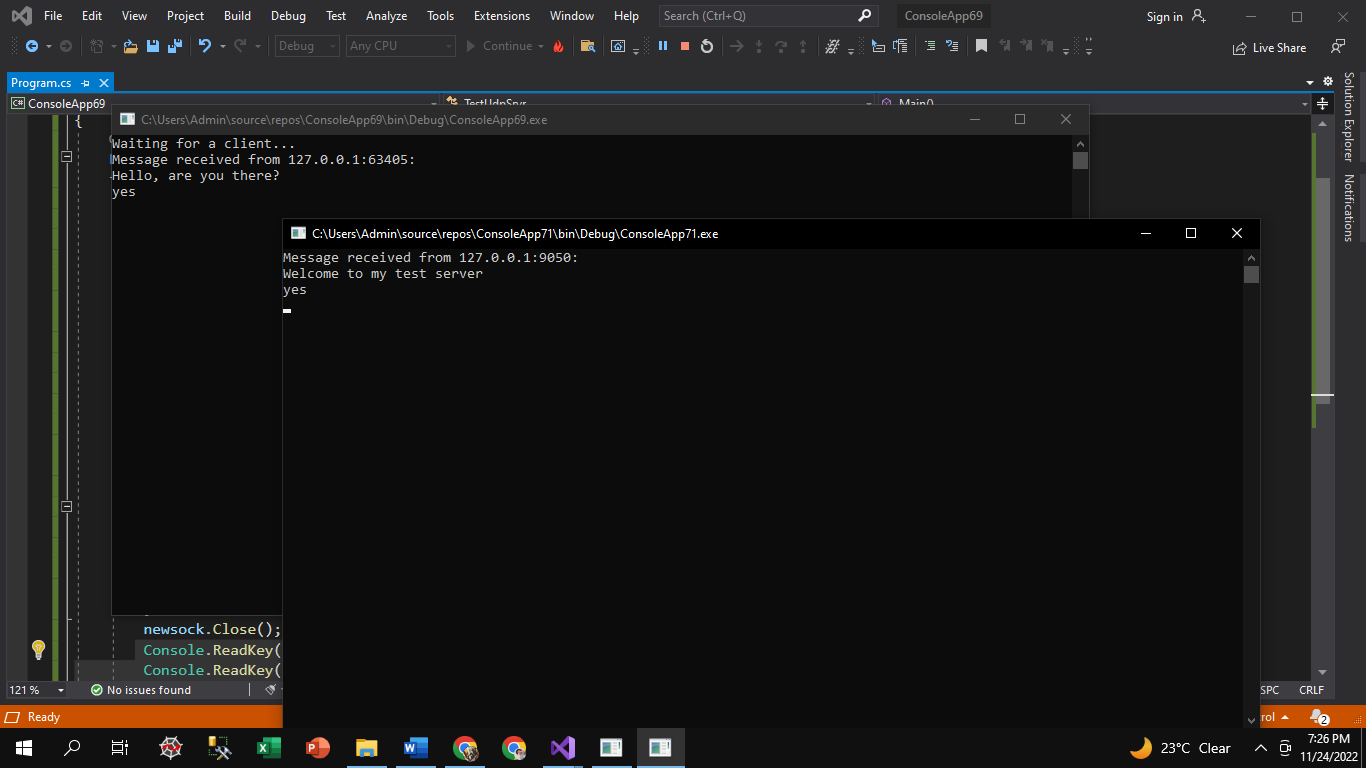
Console.WriteLine("Stopping client");

server.Close();

}

}

**Output:**



* **Listing 6.1 & 6.9: The SimpleUdpServer.cs program & The ExceptionUdpClient.cs program**

**Server code (6.1):**

using System;

using System.Net;

using System.Net.Sockets;

using System.Text;

class SimpleUdpSrvr

{

public static void Main()

{

int recv;

byte[] data = new byte[1024];

IPEndPoint ipep = new IPEndPoint(IPAddress.Any, 9050);

Socket newsock = new Socket(AddressFamily.InterNetwork,

SocketType.Dgram, ProtocolType.Udp);

newsock.Bind(ipep);

Console.WriteLine("Waiting for a client...");

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint Remote = (EndPoint)(sender);

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine("Message received from {0}:", Remote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

string welcome = "Welcome to my test server";

data = Encoding.ASCII.GetBytes(welcome);

newsock.SendTo(data, data.Length, SocketFlags.None, Remote);

while (true)

{

data = new byte[1024];

recv = newsock.ReceiveFrom(data, ref Remote);

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv));

newsock.SendTo(data, recv, SocketFlags.None, Remote);

}

}

}

**Client code (6.9):**

class ExceptionUdpClient

{

public static void Main()

{

byte[] data = new byte[1024];

string input, stringData;

int recv;

IPEndPoint ipep = new IPEndPoint(

IPAddress.Parse("127.0.0.1"), 9050);

Socket server = new Socket(AddressFamily.InterNetwork, SocketType.Dgram, ProtocolType.Udp);

int sockopt = (int)server.GetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout);

Console.WriteLine("Default timeout: {0}", sockopt);

server.SetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout, 3000);

sockopt = (int)server.GetSocketOption(SocketOptionLevel.Socket, SocketOptionName.ReceiveTimeout);

Console.WriteLine("New timeout: {0}", sockopt);

string welcome = "Hello, are you there?";

data = Encoding.ASCII.GetBytes(welcome);

server.SendTo(data, data.Length, SocketFlags.None, ipep);

IPEndPoint sender = new IPEndPoint(IPAddress.Any, 0);

EndPoint Remote = (EndPoint)sender;

data = new byte[1024];

try

{ recv = server.ReceiveFrom(data, ref Remote);

Console.WriteLine("Message received from {0}:", Remote.ToString());

Console.WriteLine(Encoding.ASCII.GetString(data, 0, recv)); }

catch (SocketException)

{ Console.WriteLine("Problem communicating with remote server");

return; }

while (true)

{ input = Console.ReadLine();

if (input == "exit")

break;

server.SendTo(Encoding.ASCII.GetBytes(input), ipep);

data = new byte[1024];

try

{ recv = server.ReceiveFrom(data, ref Remote);

stringData = Encoding.ASCII.GetString(data, 0, recv);

Console.WriteLine(stringData); }

catch (SocketException)

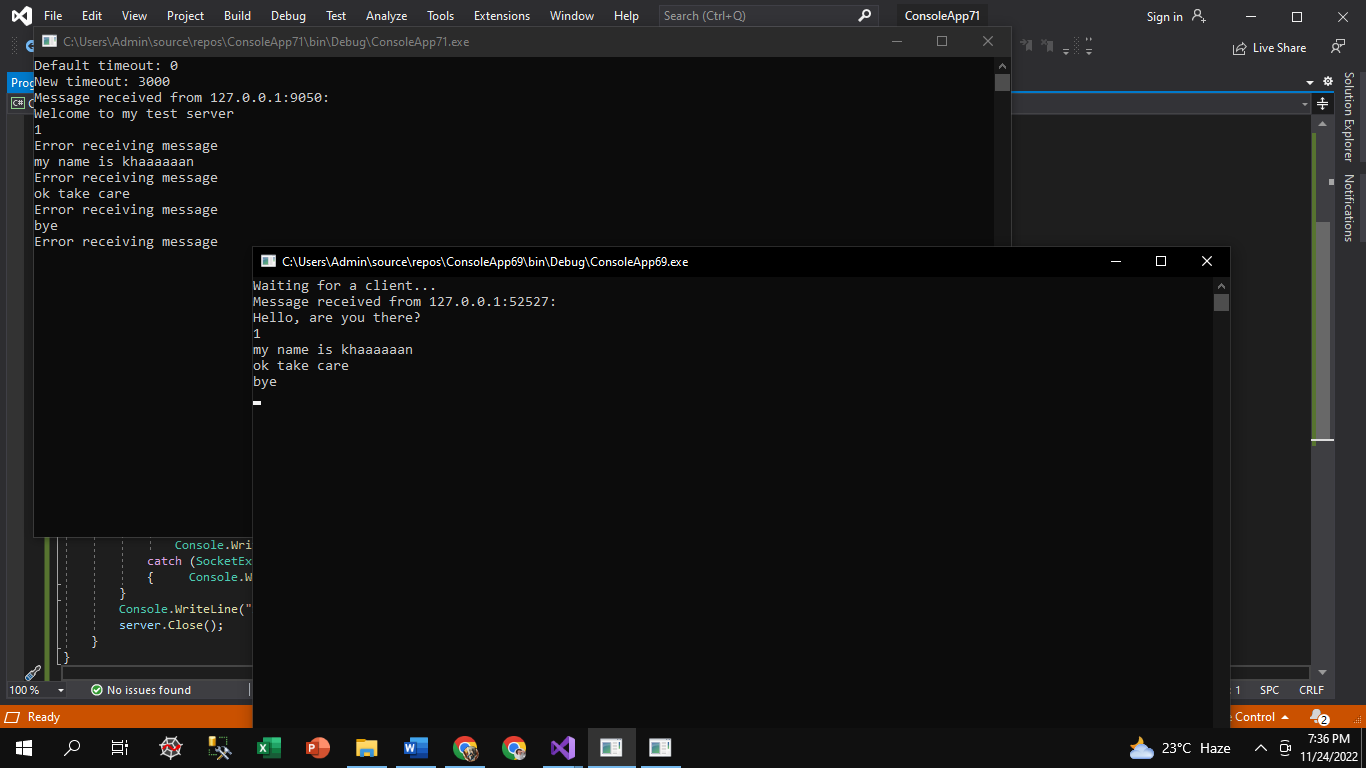
{ Console.WriteLine("Error receiving message"); }

}

Console.WriteLine("Stopping client");

server.Close();

**Output:**



**Lab # 9: Asynchronous**

**Server Code:**

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

TcpClient client = (TcpClient)lstClients[listBox1.SelectedItem.ToString()]; NetworkStream ns = client.GetStream();

StreamWriter sw = new StreamWriter(ns);

string textToSend = "server says:" + textBox3.Text; sw.WriteLine(textToSend);

textBox1.Text += textToSend + Environment.NewLine;sw.Flush();

}

private void Form1\_Load(object sender, EventArgs e)

{

CheckForIllegalCrossThreadCalls = false;

TcpListener listner = new TcpListener(IPAddress.Loopback, 11000);listner.Start();

listner.BeginAcceptTcpClient(new AsyncCallback(ClientConnect), listner);

}

Dictionary<string, TcpClient> lstClients = new Dictionary<string, TcpClient>(); byte[] b = new byte[1024];

private void ClientConnect(IAsyncResult ar)

{

TcpListener listner = (TcpListener)ar.AsyncState;TcpClient client = listner.EndAcceptTcpClient(ar);NetworkStream ns = client.GetStream();

object[] a = new object[2];a[0] = ns; a[1] = client;

ns.BeginRead(b, 0, b.Length, new AsyncCallback(ReadMsg), a); listner.BeginAcceptTcpClient(new AsyncCallback(ClientConnect), listner);

}

private void ReadMsg(IAsyncResult ar)

{

object[] a = (object[])ar.AsyncState; NetworkStream ns = (NetworkStream)a[0];TcpClient client = (TcpClient)a[1]; int count = ns.EndRead(ar);

string msg = ASCIIEncoding.ASCII.GetString(b,0,count); if(msg.Contains("@name@"))

{

string name = msg.Replace("@name@",""); lstClients.Add(name,client); listBox1.Items.Add(name);

}

else

{

}

textBox1.Text +=msg +Environment.NewLine;

ns.BeginRead(b, 0, b.Length, new AsyncCallback(ReadMsg), a);

}

private void button2\_Click(object sender, EventArgs e)

{

foreach (object item in listBox1.Items)

{

var a = (item.ToString()); TcpClient client =

(TcpClient)lstClients[a];NetworkStream ns

= client.GetStream(); StreamWriter sw = new StreamWriter(ns);

string textToSend = "server says:" + textBox3.Text;

sw.WriteLine(textToSend);

textBox1.Text += textToSend + Environment.NewLine; sw.Flush();

}

**Client Code:**

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

private void Form1\_Load(object sender, EventArgs e)

{

}

byte[] b = new byte[1024]; TcpClient client = new TcpClient();

private void button1\_Click(object sender, EventArgs e)

{

CheckForIllegalCrossThreadCalls = false; client.Connect(IPAddress.Loopback, 11000);NetworkStream ns = client.GetStream(); StreamWriter sw

= new StreamWriter(ns); sw.WriteLine("@name@"+\_nametxt.Text)

; sw.Flush(); ns.BeginRead(b,0,b.Length,ReadMsg,ns

);

}

private void ReadMsg(IAsyncResult ar)

{

NetworkStream ns = (NetworkStream)ar.AsyncState;int count = ns.EndRead(ar);

txtDisplay.Text += ASCIIEncoding.ASCII.GetString(b, 0, count); ns.BeginRead(b, 0, b.Length, ReadMsg, ns);

}

private void button2\_Click(object sender, EventArgs e)

{

NetworkStream ns = client.GetStream(); StreamWriter sw = new StreamWriter(ns); sw.WriteLine(\_nametxt.Text + " says: " + textBox1);sw.Flush();