LAB#01 (OOP PILLARS)

CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using System.Threading;

namespace Hotelmanagementsystem
{
    static class Program
    {
        /// <summary>
        /// The main entry point for the application.
        /// </summary>
        [STAThread]
        static void Main()  //----------------------------Entry-Point---------
--------------------------------//
        {
            SingleRoom SR = new SingleRoom();
            SR.Price();
            DoubleRoom DR = new DoubleRoom();
            DR.Price();
            FamilyRoom FR = new FamilyRoom();
            FR.Price();
            DecideMealPlan DMP = new DecideMealPlan("AmericanPlan");
            DecideMealPlan DMP1 = new DecideMealPlan("BednBreakfast", "Europea
nPlan", "_BednBreakfast");
            Economy E = new Economy();
            E.Bill();
            Business B = new Business();
            B.Bill();
            VIP V = new VIP();
            V.Bill();

        }



    }
    //-------------------------------------------Encapsulation---------------
-------------------------------------------------------//
    class DecideMealPlan
    {
        private string _AmericanPlan;
```

```
        private string _BednBreakfast;
        private string _ContinentalPlan;
        private string _EuropeanPlan;

        public DecideMealPlan(string AmericanPlan)
        {
            this._AmericanPlan = AmericanPlan;
            if(AmericanPlan==_AmericanPlan)
            {
                Console.WriteLine("The cost charged is 5000");
            }

        }
        public DecideMealPlan(string BednBreakfast,string ContinentalPlan,stri
ng EuropeanPlan)
        {
            this._BednBreakfast = BednBreakfast;
            this._ContinentalPlan = ContinentalPlan;
            this._EuropeanPlan = EuropeanPlan;
            if(_BednBreakfast==BednBreakfast)
            {
                Console.WriteLine("The cost charged is 5000");
            }
            if(ContinentalPlan==_ContinentalPlan)
            {
                Console.WriteLine("The cost charged is 8000");
            }
            if(EuropeanPlan==_EuropeanPlan)
            {
                Console.WriteLine("The cost charged is 10000");
            }

        }

    }
    //-----------------------------------------Abstraction & Inheritance---
-------------------------------------------------------------------//
    public abstract class Room
    {
        public abstract double Price();
        protected double SingleRoomCharges = 8000;


    }
    public class SingleRoom : Room
    {

        public override double Price()
        {
```

```
            return SingleRoomCharges * 1;

        }
    }
        public class DoubleRoom : Room
        {
        public override double Price()
            {
            return SingleRoomCharges * 2;
            }

        }
        public class FamilyRoom : Room
        {
        public override double Price()
        {
            return SingleRoomCharges * 3;

        }

    }
    //-------------------------------------------Polymorphism----------------
-------------------------------------------------------//
    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:

```
Microsoft Visual Studio Debug Console
The cost charged is 5000
The cost charged is 5000
The cost charged is 8000
The cost charged is 10000

C:\Users\Omar\source\repos\ConsoleApp19\ConsoleApp19\bin\Debug\netcoreapp3.1\ConsoleApp19.exe (proce
h code 0.
Press any key to close this window . . .
```

LAB#02 (CLIENT SERVER CONNECTION)

CLIENT SIDE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace ConsoleApplication112
/// <summary>
/// ///////////client side code
/// </summary>
{
    class Program
    {
        static void Main(string[] args)
        {
            IPAddress ip = IPAddress.Parse("192.168.0.107");
            IPEndPoint ep = new IPEndPoint(ip, 2000);
            Socket sp = new Socket(ip.AddressFamily, SocketType.Stream, Protoc
olType.Tcp);
            sp.Connect(ep);
            Console.WriteLine("CONNECTED WITH THE SERVER");
            Console.ReadKey();




        }
    }
}
```

SERVER SIDE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace ConsoleApplication112
```

```
{

    /// <summary>
    /// ///////////server side code
    /// </summary>
    class Program
    {
        static void Main(string[] args)
        {
            IPAddress ip = IPAddress.Parse("192.168.0.107");
            IPEndPoint ep = new IPEndPoint(ip, 2000);
            Socket sp = new Socket(ip.AddressFamily, SocketType.Stream, Protoc
olType.Tcp);
            sp.Bind(ep);
            sp.Listen(1);
            Console.WriteLine("waiting for the client");
            Socket c1 = sp.Accept();
            Console.WriteLine("connected with the server");



        }
    }
}
```

OUTPUT:

```
C:\Users\Omar\source\repos\ConsoleApp19\ConsoleApp19\bin\Debug\netcoreapp3.1\ConsoleApp19.exe
CONNECTED WITH THE SERVER
```

LAB#03 (CHAPTER ONE LISTINGS)

(1.4-1.8)

```
/// <summary>
/// ///////////server side code
/// </summary>
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        string test1 = "This is a test string";
        string test2, test3;
        test2 = test1.Insert(15, "application ");
        test3 = test1.ToUpper();
        Console.WriteLine("test1: '{0}'", test1);
        Console.WriteLine("test2: '{0}'", test2);
        Console.WriteLine("test3: '{0}'", test3);
        if (test1 == test3)
            Console.WriteLine("test1 is equal to test3");
        else
            Console.WriteLine("test1 is not equal to test3");
        test2 = test1.Replace("test", "sample");
        Console.WriteLine("the new test2: '{0}'", test2);
    }
```

No issues found

```
C:\Windows\system32\cmd.exe
test1: 'This is a test string'
test2: 'This is a test application string'
test3: 'THIS IS A TEST STRING'
test1 is not equal to test3
the new test2: 'This is a sample string'
Press any key to continue . . .
```

App21 | ConsoleApplication112.Program | Main(string[] args)

```
/// ///////////server side code
/// </summary>
0 references
class Program
{
    0 references
    static void Main(string[] args)
    {
        StringBuilder sb = new StringBuilder("test string");
        int length = 0;
        length = sb.Length;
        Console.WriteLine("The result is: '{0}'", sb);
        Console.WriteLine("The length is: {0}", length);
        sb.Length = 4;
        length = sb.Length;
        Console.WriteLine("The result is: '{0}'", sb);
        Console.WriteLine("The length is: {0}", length);
        sb.Length = 20;
        length = sb.Length;
        Console.WriteLine("The result is: '{0}'", sb);
        Console.WriteLine("The length is: {0}", length);
    }

}
}
```

No issues found

```
C:\Windows\system32\cmd.exe
The result is: 'test string'
The length is: 11
The result is: 'test'
The length is: 4
The result is: 'test                '
The length is: 20
Press any key to continue . . .
```

MUHAMMAD UMAR KHAN 10619
NUMERICAL COMPUTING LAB MANUAL 108157

```
/// <summary>
/// ////////////server side code
/// </summary>
class Program
{
    static void Main(string[] args)
    {
        string logFile = "LOGFILE.TXT";
        FileStream fs = new FileStream(logFile,
        FileMode.OpenOrCreate, FileAccess.Write);
        StreamWriter sw = new StreamWriter(fs);
        StreamReader sr = new StreamReader(fs);
        sw.WriteLine("First log entry");
        sw.WriteLine("Second log entry");
        while (sr.Peek() > -1)
        {
            Console.WriteLine(sr.ReadLine());
        }
        sw.Close();
        sr.Close();
        fs.Close();
    }
}
```

```
C:\Windows\system32\cmd.exe
Unhandled Exception: System.ArgumentException: Stream was not readable.
   at System.IO.StreamReader..ctor(Stream stream, Encoding encoding, Boolean detectEncodingFromByteOrderMarks, Int32 buf
ferSize, Boolean leaveOpen)
   at System.IO.StreamReader..ctor(Stream stream)
   at ConsoleApplication112.Program.Main(String[] args) in C:\Users\Omar\source\repos\ConsoleApp21\ConsoleApp21\Program.
cs:line 24
Press any key to continue . . .
```

```
/// </summary>
class Program
{
    static void Main(string[] args)
    {
        int var1 = 1000, var2 = 0, var3;
        var3 = var1 / var2;
        Console.WriteLine("The result is: {0}", var3);

    }
}
}
```

```
C:\Windows\system32\cmd.exe
Unhandled Exception: System.DivideByZeroException: Attempted to divide by zero.
   at ConsoleApplication112.Program.Main(String[] args) in C:\Users\Omar\source\repos\ConsoleApp21\ConsoleApp21\Program.
cs:line 21
Press any key to continue . . .
```

```
class Program
{
    static void Main(string[] args)
    {
        int var1 = 1000, var2 = 0, var3;
        try
        {
            var3 = var1 / var2;
        }
        catch (ArithmeticException e)
        {
            Console.WriteLine("Exception: {0}",
            e.ToString());
            var3 = -1;
        }
        catch (Exception e)
        {
            Console.WriteLine("Exception: {0}",
            e.ToString());
            var3 = -2;
        }
        Console.WriteLine("The result is: {0}",

    }
```

```
C:\Windows\system32\cmd.exe
Exception: System.DivideByZeroException: Attempted to divide by zero.
   at ConsoleApplication112.Program.Main(String[] args) in C:\Users\Omar\source\r
cs:line 23
The result is: -1
Press any key to continue . . .
```

LAB#04(MULTITHREADS LISTING)

SERVER SIDE CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;
using System.Threading;

namespace Server
{
    class Program                                           //SERVER SIDE
    {
        static void Main(string[] args)
        {
            Thread t = new Thread(delegate ()
            {
                Server myserver = new Server("192.168.0.107", 2100);
            });
            t.Start();
        }
    }
    class Server
    {
        TcpListener server = null;
        public Server(string ip, int port)
        {
            IPAddress localAddr = IPAddress.Parse(ip);
            server = new TcpListener(localAddr, port);
            server.Start();
            StartListener();
        }
        public void StartListener()
        {
            try
            {
                while (true)
                {
                    Console.WriteLine("Waiting for a connection...");
                    TcpClient client = server.AcceptTcpClient();
                    Console.WriteLine("Connected!");
                    Thread t = new Thread(new ParameterizedThreadStart(HandleD
evice));
                    t.Start(client);
                }
```

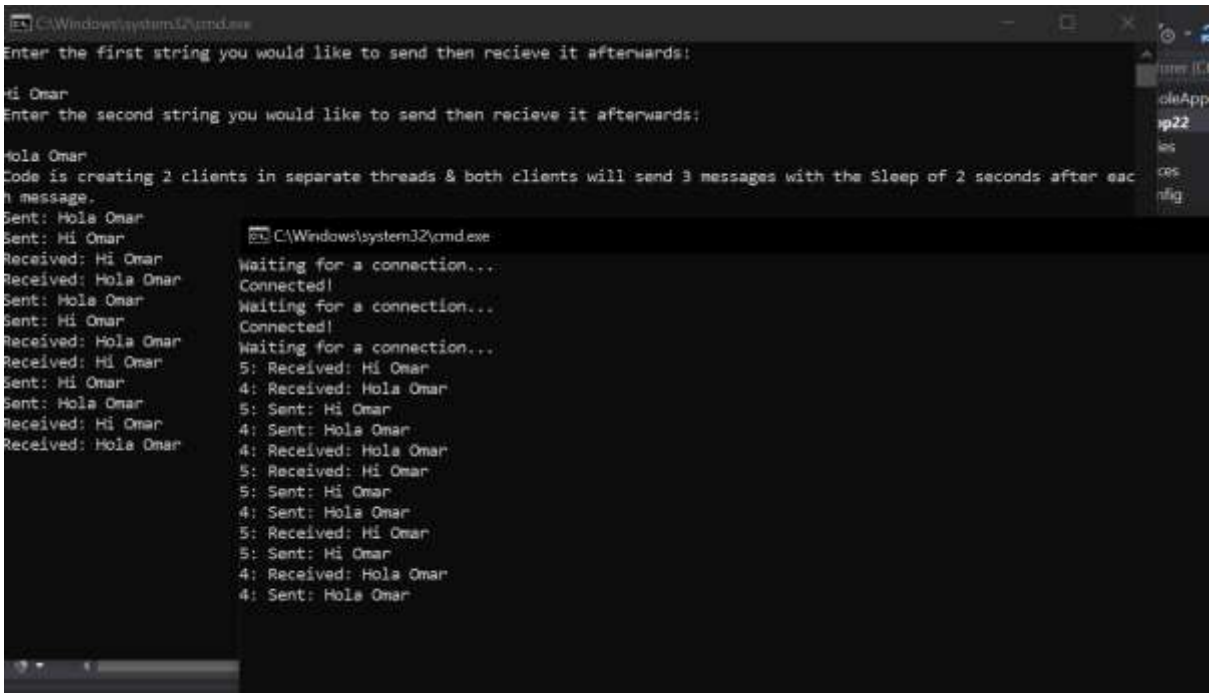
```
            }
            catch (SocketException e)
            {
                Console.WriteLine("Socket Exception : {0}", e);
            }
        }
        public void HandleDevice(Object obj)
        {
            TcpClient client = (TcpClient)obj;
            var stream = client.GetStream();
            string imei = string.Empty;
            string data = null;
            Byte[] bytes = new Byte[1024];
            int i;
            try
            {
                while ((i = stream.Read(bytes, 0, bytes.Length)) != 0)
                {
                    string hex = BitConverter.ToString(bytes);
                    data = Encoding.ASCII.GetString(bytes, 0, i);
                    Console.WriteLine("{1}: Received: {0}", data, Thread.Curre
ntThread.ManagedThreadId);
                    string str = "hello";
                    Byte[] reply = Encoding.ASCII.GetBytes(data);
                    stream.Write(reply, 0, reply.Length);
                    Console.WriteLine("{1}: Sent: {0}", data, Thread.CurrentTh
read.ManagedThreadId);
                }
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: {0}", e.ToString());
                client.Close();
            }
        }

    }

}
```

CLIENT SIDE CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net.Sockets;
using System.Threading;

namespace ConsoleApplication45
{
    class Program                   //CLIENT SIDE
    {
        static void Main(string[] args)
        {
            string a;
            string b;
            Console.WriteLine("Enter the first string you would like to send t
hen recieve it afterwards:\n");
            a = Console.ReadLine();
            Console.WriteLine("Enter the second string you would like to send
then recieve it afterwards:\n");
            b = Console.ReadLine();

            Console.WriteLine("Code is creating 2 clients in separate threads
& both clients will send 3 messages with the Sleep of 2 seconds after each mes
sage.");


            new Thread(() =>
            {


                Thread.CurrentThread.IsBackground = true;
                Connect("192.168.0.107", a);
            }).Start();
            new Thread(() =>
            {
                Thread.CurrentThread.IsBackground = true;
                Connect("192.168.0.107", b);
            }).Start();
            Console.ReadLine();
        }
        static void Connect(String server, String message)
        {
            try
            {
                Int32 port = 2100;
```

```
                TcpClient client = new TcpClient(server, port);
                NetworkStream stream = client.GetStream();
                int count = 0;
                while (count++ < 3)
                {
                    // Translate the Message into ASCII.
                    Byte[] data = System.Text.Encoding.ASCII.GetBytes(message)
;
                    // Send the message to the connected TcpServer.
                    stream.Write(data, 0, data.Length);
                    Console.WriteLine("Sent: {0}", message);
                    // Bytes Array to receive Server Response.
                    data = new Byte[256];
                    String response = String.Empty;
                    // Read the Tcp Server Response Bytes.
                    Int32 bytes = stream.Read(data, 0, data.Length);
                    response = System.Text.Encoding.ASCII.GetString(data, 0, b
ytes);
                    Console.WriteLine("Received: {0}", response);
                    Thread.Sleep(2000);
                }
                stream.Close();
                client.Close();
            }
            catch (Exception e)
            {
                Console.WriteLine("Exception: {0}", e);
            }
            Console.ReadLine();
        }
    }
}
```

OUTPUT:



LAB#05(MULTITHREADS LISTING – CH#05 LISTINGS)

CLIENT SIDE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;
namespace ConsoleApp24
{
    class Program
    {
        static void Main(string[] args)    //CLIENT SLIDE
        {
            byte[] data = new byte[1024];
            string input, stringData;
            IPEndPoint ipep = new IPEndPoint(
            IPAddress.Parse("192.168.0.107"), 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);
            }
```

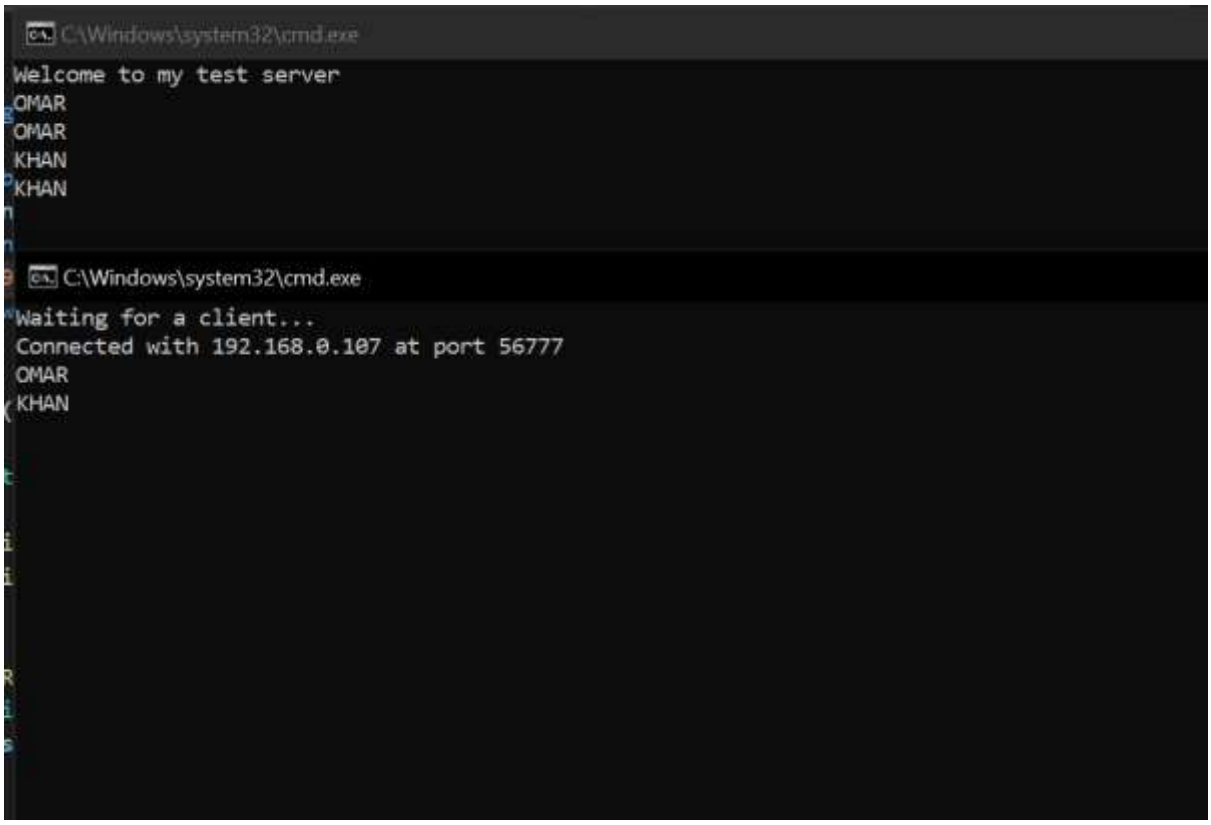
```
            Console.WriteLine("Disconnecting from server...");
            server.Shutdown(SocketShutdown.Both);
            server.Close();
        }
    }
}
```

SERVER SIDE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace ConsoleApp23
{
    class Program
    {
        static void Main(string[] args)         //SERVER SIDE
        {
            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();
        }
    }
}
```

OUTPUT:



(5.3-5.4)

SERVER SIDE CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;
namespace ConsoleApp24
{
    class Program
    {
        static void Main(string[] args)    //SERVER SIDE
        {
            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();
        }
    }
}
```

CLIENT SIDE CODE:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Net;
using System.Net.Sockets;

namespace ConsoleApp25
{
    class Program
    {
        static void Main(string[] args)             //CLIENT SIDE
        {
            byte[] data = new byte[1024];
            string stringData;
            IPEndPoint ipep = new IPEndPoint(
            IPAddress.Parse("192.168.0.107"), 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();
        }
    }
}
```

OUTPUT:

```
C:\Windows\system32\cmd.exe
elcome to my test server
isconnecting from server...
ress any key to continue . . .
```

```
C:\Windows\system32\cmd.exe
Waiting for a client...
Connected with 192.168.0.107 at port 53293
message 1
message 2message 3message
 4message 5


Disconnecting from 192.168.0.107
Press any key to continue . . .
```

(5.5-5.6)

SERVER SIDE CODE:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class FixedTcpSrvr
{
    private static int SendData(Socket s, byte[] data)          //SERVER SIDE
    {
        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.Str
eam, 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();
    }
}
```

CLIENT SIDE CODE:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class FixedTcpClient
{
    private static int SendData(Socket s, byte[] data)          //CLIENT SIDE
    {
        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("192.168.0.107"), 905
0);
        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:

```
C:\Windows\system32\cmd.exe
Welcome to my test server
Disconnecting from server...
Press any key to continue . . .

C:\Windows\system32\cmd.exe
Waiting for a client...
Connected with 192.168.0.107 at port 53320
message 1
message 2
message 3
message 4
message 5
Disconnected from 192.168.0.107
Press any key to continue . . .
```

(CH#7 HELPER CLASSES)

(7.1-7.2)

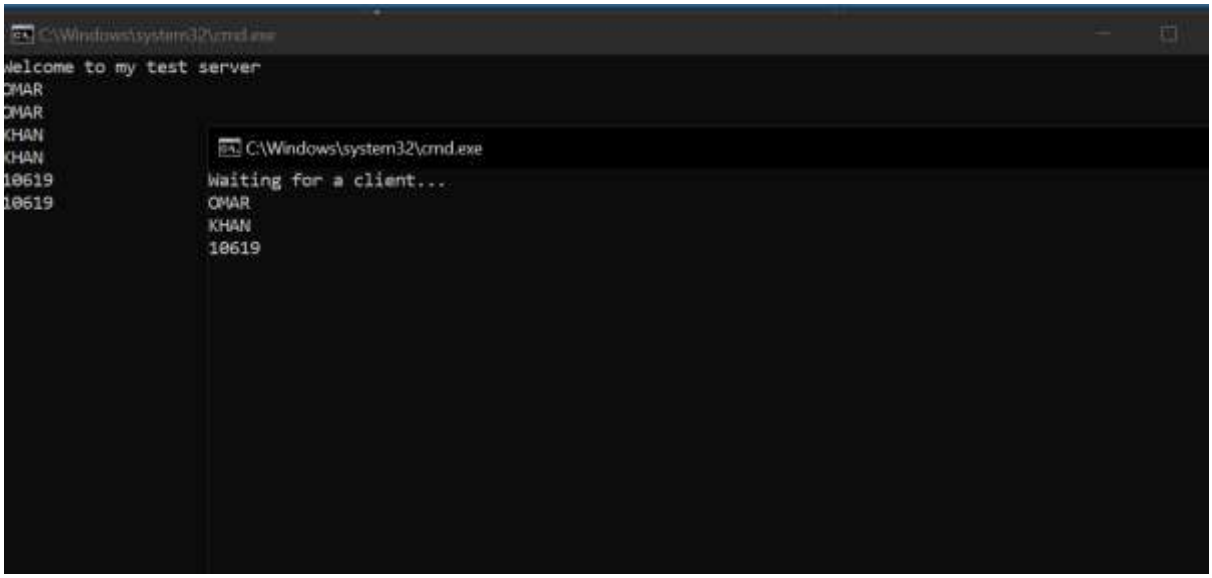
CLIENT SIDE CODE:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpClientSample
{
    public static void Main()                //CLIENT SIDE
                                               //TCP CLIENT
    {
        byte[] data = new byte[1024];
        string input, stringData;
        TcpClient server;
        try
        {
            server = new TcpClient("192.168.0.107", 9050);
        }
        catch (SocketException)
        {
            Console.WriteLine("Unable to connect to server");
            return;
        }
        NetworkStream ns = server.GetStream();
        int recv = ns.Read(data, 0, data.Length);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
        while (true)
        {
            input = Console.ReadLine();
            if (input == "exit")
                break;
            ns.Write(Encoding.ASCII.GetBytes(input), 0, input.Length);
            ns.Flush();
            data = new byte[1024];
            recv = ns.Read(data, 0, data.Length);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
        }
        Console.WriteLine("Disconnecting from server...");
        ns.Close();
        server.Close();
    }
}
```

SERVER SIDE CODE:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
class TcpListenerSample
{
    public static void Main()                        //SERVER SIDE
    {
        int recv;
        byte[] data = new byte[1024];
        TcpListener newsock = new TcpListener(9050);
        newsock.Start();
        Console.WriteLine("Waiting for a client...");
        TcpClient client = newsock.AcceptTcpClient();
        NetworkStream ns = client.GetStream();
        string welcome = "Welcome to my test server";
        data = Encoding.ASCII.GetBytes(welcome);
        ns.Write(data, 0, data.Length);
        while (true)
        {
            data = new byte[1024];
            recv = ns.Read(data, 0, data.Length);
            if (recv == 0)
                break;
            Console.WriteLine(
            Encoding.ASCII.GetString(data, 0, recv));
            ns.Write(data, 0, recv);
        }
        ns.Close();
        client.Close();
        newsock.Stop();
    }
}
```

OUTPUT:



CHAPTER#03 LISTINGS

(3.1-3.4)

```
public static void Main()
{
    IPAddress test1 = IPAddress.Parse("192.168.9.107");
    IPAddress test2 = IPAddress.Loopback;
    IPAddress test3 = IPAddress.Broadcast;
    IPAddress test4 = IPAddress.Any;
    IPAddress test5 = IPAddress.None;
    IPHostEntry ihe = Dns.GetHostByName(Dns.GetHostName());
    IPAddress myself = ihe.AddressList[0];
    if (IPAddress.IsLoopback(test2))
        Console.WriteLine("The Loopback address is: {0}", test2.ToString());
    else
        Console.WriteLin
    Console.WriteLine("T
    myself.ToString());
    if (myself == test2)
        Console.WriteLin
    else
        Console.WriteLin
    Console.WriteLine("T
    Console.WriteLine("
    Console.WriteLine("T
    Console.WriteLine("The NONE address is: {0}", test5.ToString());

}
```

```
C:\Windows\system32\cmd.exe
The Loopback address is: 127.0.0.1
The Local IP address is: 192.168.0.107
The loopback address is not the local address.

The test address is: 192.168.9.107
Broadcast address: 255.255.255.255
The ANY address is: 0.0.0.0
The NONE address is: 255.255.255.255
Press any key to continue . . .
```

```
using System;
using System.Net;

class IPEndPointSample
{
    public static void Main()
    {
        IPAddress test1 = IPAddress.Parse("192.168.0.107");
        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 Aport is: {1}\n", ie.Address, ie.Port);
        Console.WriteLine("The min port number is: {0}",
        IPEndPoint.MinPort);
        Console.WriteLine("The max port number is: {0}\n",
        IPEndPoint.MaxPort);
        ie.Port = 80;
        Console.WriteLine("The changed IPEndPoint value A is: {0}", ie.ToString());
        SocketAddress sa = ie.Serialize();
        Console.WriteLine("The SocketAddress is: {0}", sa.ToString());
    }
}
```

```
C:\Windows\system32\cmd.exe
The IPEndPoint is: 192.168.0.107:8000
The AddressFamily is: InterNetwork
The address is: 192.168.0.107, and the Aport is: 8000

The min port number is: 0
The max port number is: 65535

The changed IPEndPoint value A is: 192.168.0.107:80
The SocketAddress is: InterNetwork:16:{0,80,192,168,0,107,0,0,0,0,0,0,0,0}
Press any key to continue . . .
```

MUHAMMAD UMAR KHAN 10619
NUMERICAL COMPUTING LAB MANUAL 108157

```
using System;
using System.Net;
using System.Net.Sockets;
class SockProp
{
    public static void Main()
    {
        IPAddress ia = IPAddress.Parse("192.168.0.107");
        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();
    }
}
```

```
AddressFamily: InterNetwork
SocketType: Stream
ProtocolType: Tcp
Blocking: True
new Blocking: False
Connected: False
Local EndPoint: 192.168.0.107:8000
Press any key to continue . . .
```

```
IPAddress host = IPAddress.Parse("192.168.0.107");
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;
}
sock.Close();
```

```
Problem connecting to host
System.Net.Sockets.SocketException (0x80004005): No connection could be made because the tar
et machine actively refused it 192.168.0.107:8000
   at System.Net.Sockets.Socket.DoConnect(EndPoint endPointSnapshot, SocketAddress socketAdd
ess)
   at System.Net.Sockets.Socket.Connect(EndPoint remoteEP)
   at SocketExcept.Main() in C:\Users\Dmar\source\repos\ConsoleApp25\ConsoleApp25\Program.cs
line 15
Press any key to continue . . .
```

LAB#08 (ASYNCHRONOUS)

SEVER SIDE CODE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Windows.Forms;

namespace AsynServer
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {

            CheckForIllegalCrossThreadCalls = false;
            TcpListener listener = new TcpListener(IPAddress.Loopback, 11000);
            listener.Start(10);
            listener.BeginAcceptTcpClient(new AsyncCallback(ClientConnect), li
stener);
        }

        Dictionary<string, TcpClient> lstClients =new Dictionary<string, TcpCl
ient>();
        byte[] b = new byte[1024];
        private void ClientConnect(IAsyncResult ar)
        {
            TcpListener listener =(TcpListener) ar.AsyncState;
            TcpClient client= listener.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);
            listener.BeginAcceptTcpClient(new AsyncCallback(ClientConnect), li
stener);
```

```
        }

        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);
                lstbxClients.Items.Add(name);
            }
            else
            {
                txtDisplay.Text += msg + Environment.NewLine;
            }
            ns.BeginRead(b, 0, b.Length, new AsyncCallback(ReadMsg), a);
        }

        private void button2_Click(object sender, EventArgs e)
        {
            TcpClient client = (TcpClient)lstClients[lstbxClients.SelectedItem
.ToString()];
            NetworkStream ns = client.GetStream();
            StreamWriter sw = new StreamWriter(ns);
            string textToSend = "Server Says:" + txtMsg.Text;
            sw.WriteLine(textToSend);
            txtDisplay.Text += textToSend + Environment.NewLine;
            sw.Flush();
        }

        private void lstbxClients_SelectedIndexChanged(object sender, EventArg
s e)
        {

        }
    }
}
```

CLIENT SIDE CODE:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Windows.Forms;

namespace AysncClient
{
    public partial class Form1 : Form
    {
        public Form1()
        {
            InitializeComponent();
        }
        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@" + txtName.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(txtName.Text + " Says: " + txtMsg.Text);
            sw.Flush();

        }

        private void Form1_Load(object sender, EventArgs e)
        {

        }
    }
}
```