BT1- ThreadedTcp Server:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
class ThreadedTcpSrvr
{
  private TcpListener client;
  public ThreadedTcpSrvr()
  {
   client = new TcpListener(9050);
   client.Start();
   Console.WriteLine("Waiting for clients...");
   while(true)
   {
     while (!client.Pending())
     {
      Thread.Sleep(1000);
     }
     ConnectionThread newconnection = new ConnectionThread();
     newconnection.threadListener = this.client;
     Thread newthread = new Thread(new
          ThreadStart(newconnection.HandleConnection));
     newthread.Start();
   }
  }
  public static void Main()
  {
   ThreadedTcpSrvr server = new ThreadedTcpSrvr();
  }
}
class ConnectionThread
{
  public TcpListener threadListener;
  private static int connections = 0;
  public void HandleConnection()
  {
   int recv;
   byte[] data = new byte[1024];
   TcpClient client = threadListener.AcceptTcpClient();
   NetworkStream ns = client.GetStream();
   connections++;
   Console.WriteLine("New client accepted: {0} active connections",
            connections);
   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;
```

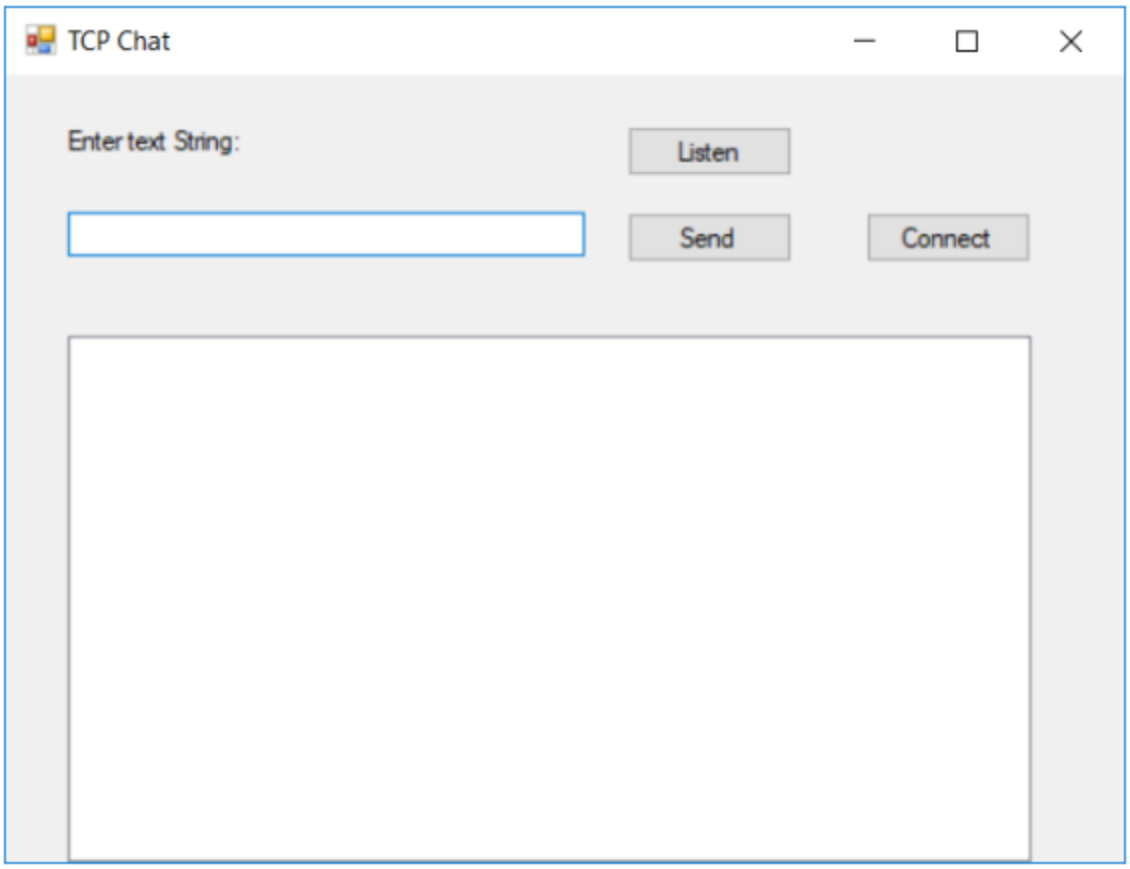
```
      ns.Write(data, 0, recv);
    }
    ns.Close();
    client.Close();
    connections—;
    Console.WriteLine("Client disconnected: {0} active connections",
              connections);
  }
}
```

BT2 - TCPChat:

```
using System;
using System.Drawing;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.Windows.Forms;
class TcpChatForm
{
  private static Socket client;
  private static byte[] data = new byte[1024];
  public TcpChatForm ()
  {
   InitializeComponent();
  }
  void ButtonListenOnClick(object obj, EventArgs ea)
  {
   results.Items.Add("Listening for a client...");
   Socket newsock = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
    ProtocolType.Tcp);
   IPEndPoint iep = new IPEndPoint(IPAddress.Any, 9050);
   newsock.Bind(iep);
   newsock.Listen(5);
   newsock.BeginAccept(new AsyncCallback(AcceptConn), newsock);
  }
  void ButtonConnectOnClick(object obj, EventArgs ea)
  {
   results.Items.Add("Connecting...");
   client = new Socket(AddressFamily.InterNetwork, SocketType.Stream,
    ProtocolType.Tcp);
   IPEndPoint iep = new IPEndPoint(IPAddress.Parse("127.0.0.1"), 9050);
   client.BeginConnect(iep, new AsyncCallback(Connected), client);
  }
  void ButtonSendOnClick(object obj, EventArgs ea)
  {
   byte[] message = Encoding.ASCII.GetBytes(newText.Text);
   newText.Clear();
   client.BeginSend(message, 0, message.Length, 0,
    new AsyncCallback(SendData), client);
  }
  void AcceptConn(IAsyncResult iar)
```

```
  {
   Socket oldserver = (Socket)iar.AsyncState;
   client = oldserver.EndAccept(iar);
   results.Items.Add("Connection from: " + client.RemoteEndPoint.ToString());
   Thread receiver = new Thread(new ThreadStart(ReceiveData));
   receiver.Start();
  }
  void Connected(IAsyncResult iar)
  {
   try
   {
     client.EndConnect(iar);
     results.Items.Add("Connected to: " + client.RemoteEndPoint.ToString());
     Thread receiver = new Thread(new ThreadStart(ReceiveData));
     receiver.Start();
   } catch (SocketException)
   {
     results.Items.Add("Error connecting");
   }
  }
  void SendData(IAsyncResult iar)
  {
   Socket remote = (Socket)iar.AsyncState;
   int sent = remote.EndSend(iar);
  }
  void ReceiveData()
  {
   int recv;
   string stringData;
   while (true)
   {
     recv = client.Receive(data);
     stringData = Encoding.ASCII.GetString(data, 0, recv);
     if (stringData == "bye")
      break;
     results.Items.Add(stringData);
   }
   stringData = "bye";
   byte[] message = Encoding.ASCII.GetBytes(stringData);
   client.Send(message);
   client.Close();
   results.Items.Add("Connection stopped");
   return;
  }

}
```



BT3 - ThreadPoolSample:

```
using System;
using System.Threading;
class ThreadPoolSample
{
  public static void Main()
  {
   ThreadPoolSample tps = new ThreadPoolSample();
  }
  public ThreadPoolSample()
  {
   int i;
   ThreadPool.QueueUserWorkItem(new WaitCallback(Counter));
   ThreadPool.QueueUserWorkItem(new WaitCallback(Counter2));
   for(i = 0; i < 10; i++)
   {
     Console.WriteLine("main: {0}", i);
     Thread.Sleep(1000);
   }
  }
  void Counter(object state)
  {
   int i;
   for (i = 0; i < 10; i++)
   {
     Console.WriteLine(" thread: {0}", i);
```

```
      Thread.Sleep(2000);
    }
  }
  void Counter2(object state)
  {
   int i;
   for (i = 0; i < 10; i++)
   {
     Console.WriteLine("  thread2: {0}", i);
     Thread.Sleep(3000);
   }
  }
}
```

ThreadPoolTcpSrvr:

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
class ThreadPoolTcpSrvr
{
  private TcpListener client;
  public ThreadPoolTcpSrvr()
  {
   client = new TcpListener(9050);
   client.Start();
   Console.WriteLine("Waiting for clients...");
   while(true)
   {
     while (!client.Pending())
     {
      Thread.Sleep(1000);
     }
     ConnectionThread newconnection = new ConnectionThread();
     newconnection.threadListener = this.client;
     ThreadPool.QueueUserWorkItem(new
          WaitCallback(newconnection.HandleConnection));
   }
  }
  public static void Main()
  {
   ThreadPoolTcpSrvr tpts = new ThreadPoolTcpSrvr();
  }
}
class ConnectionThread
{
  public TcpListener threadListener;
  private static int connections = 0;
  public void HandleConnection(object state)
  {
   int recv;
```

```
    byte[] data = new byte[1024];
    TcpClient client = threadListener.AcceptTcpClient();
    NetworkStream ns = client.GetStream();
    connections++;
    Console.WriteLine("New client accepted: {0} active connections",
              connections);
    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;

      ns.Write(data, 0, recv);
    }
    ns.Close();
    client.Close();
    connections—;
    Console.WriteLine("Client disconnected: {0} active connections",
              connections);
  }
}
```