References

1- Rob Miles, "C# Development", Department of Computer Science University of Hull, 2008.

2- Fiach Reid, "Network Programming in .NET With C# and Visual Basic .NET", Elsevier Digital Press, 2004.

3- Richard Blum, "C# Network Programming", Sybex, 2003.

4- Tariq Latif Kranthi and Kumar Malkajgiri, "Adoption of VoIP", Master thesis, Lulea University of Technology, 2007.

5- David Alonso Tlahuetl Herrera, "Analysis and Implementation of TCP Friendly Rate Control in the Context of VoIP", Master thesis, Royal Institute of Technology, 2005.

6- Sharam Hekmat, "Communication Networks", PragSoft Corporation, 2000

www.pragsoft.com

6- By John Q. Walker, Jeffrey T. Hicks (2004), "Taking Charge of Your VoIP Project".

7- Jonathan Davidson, James Peter, “Voice Over IP Fundamentals”, Cisco Press, 2000.

8- "A number of VoIP-related white papers"

http://www.voip-news.com/wp/wphome.html **access 1/2009**.

9- Valdes, R. “How VoIP Works”

http://electronics.howstuffworks.com/ip-telephony.htm **access 7/2008**

10- BHUMIP KHASNABISH, "IMPLEMENTING VOICE OVER IP", A JOHN WILEY & SONS, 2003.

11- "Microsoft VoIP Introduction", Microsoft, 2008. http://download.microsoft.com/download/b/0/6/b06e9c6e-cf9c-481f-a6ed-c674e82ed1d1/VOIP.doc access 10/2008

13- Jim Doherty, Neil Anderson, "Internet Phone Services Simplified", Cisco press, 2006.

14- http://www.packetizer.com/ipmc/papers/understanding_voip access 5/2008

`

Appendix

A- Program codes

I- Main program (voip) code

```
using System;
using System.Windows.Forms;
using System.Net;
using System.Net.Sockets;
using System.Threading;
using Voice;
using System.IO;
using System.Text;
using System.Diagnostics;
using System.Collections.Generic;
using System.ComponentModel;

namespace VOIP
{

      public class Form1 : System.Windows.Forms.Form
      {

            #region variables
            private Socket r;
            private Thread t;
            private bool connected = false;
        private bool receive = false;
            private System.Windows.Forms.TextBox textBox1;
            private System.Windows.Forms.Label label1;
            private Button Call;
            private Button Exit;
            private System.ComponentModel.Container components =
null;
        private int ReceivePort = 6000;
        private Button On;
```

```
        private int SendPort = 5000;
        private PictureBox pictureBox1;
        private Button button1;
        private bool run = false;
            #endregion

            #region Form1()
            public Form1()
            {
            InitializeComponent();
            r = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
            t = new Thread(new ThreadStart(Voice_In));
            }
            #endregion

            #region For Desginer
            /// <summary>
            /// Clean up any resources being used.
            /// </summary>
            protected override void Dispose(bool disposing)
            {
                  if (disposing)
                  {
                        if (components != null)
                        {
                              components.Dispose();
                        }
                  }
                  base.Dispose(disposing);
            }

            #region Windows Form Designer generated code
            /// <summary>
            /// Required method for Designer support - do not modify
            /// the contents of this method with the code editor.
            /// </summary>
            private void InitializeComponent()
            {
```

```
            System.ComponentModel.ComponentResourceManager resources
= new System.ComponentModel.ComponentResourceManager(typeof(Form1));
            this.textBox1 = new System.Windows.Forms.TextBox();
            this.label1 = new System.Windows.Forms.Label();
            this.Call = new System.Windows.Forms.Button();
            this.On = new System.Windows.Forms.Button();
            this.button1 = new System.Windows.Forms.Button();
            this.pictureBox1 = new System.Windows.Forms.PictureBox();
            this.Exit = new System.Windows.Forms.Button();

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).BeginI
nit();
            this.SuspendLayout();
            //
            // textBox1
            //
            this.textBox1.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(255)))),
((int)(((byte)(255)))), ((int)(((byte)(192)))));
            this.textBox1.BorderStyle =
System.Windows.Forms.BorderStyle.FixedSingle;
            this.textBox1.Font = new System.Drawing.Font("Tahoma",
9.75F, System.Drawing.FontStyle.Regular,
System.Drawing.GraphicsUnit.Point, ((byte)(0)));
            this.textBox1.Location = new System.Drawing.Point(114,
214);
            this.textBox1.Name = "textBox1";
            this.textBox1.Size = new System.Drawing.Size(136, 23);
            this.textBox1.TabIndex = 4;
            //
            // label1
            //
            this.label1.Location = new System.Drawing.Point(9, 219);
            this.label1.Name = "label1";
            this.label1.Size = new System.Drawing.Size(77, 16);
            this.label1.TabIndex = 5;
            this.label1.Text = "Destination IP";
            //
            // Call
```

```
            //
            this.Call.BackColor = System.Drawing.Color.White;
            this.Call.Enabled = false;
            this.Call.Font = new System.Drawing.Font("Tahoma", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.Call.ForeColor = System.Drawing.Color.Lime;
            this.Call.Location = new System.Drawing.Point(55, 249);
            this.Call.Name = "Call";
            this.Call.Size = new System.Drawing.Size(99, 56);
            this.Call.TabIndex = 10;
            this.Call.Text = "Call";
            this.Call.UseVisualStyleBackColor = false;
            this.Call.Click += new
System.EventHandler(this.Call_Click);
            //
            // On
            //
            this.On.BackColor = System.Drawing.Color.White;
            this.On.Location = new System.Drawing.Point(285, 25);
            this.On.Name = "On";
            this.On.Size = new System.Drawing.Size(93, 41);
            this.On.TabIndex = 12;
            this.On.Text = "Power ON";
            this.On.UseVisualStyleBackColor = false;
            this.On.Click += new System.EventHandler(this.On_Click);
            //
            // button1
            //
            this.button1.BackColor = System.Drawing.Color.White;
            this.button1.DialogResult =
System.Windows.Forms.DialogResult.Cancel;
            this.button1.Location = new System.Drawing.Point(285,
93);
            this.button1.Name = "button1";
            this.button1.Size = new System.Drawing.Size(93, 41);
            this.button1.TabIndex = 15;
            this.button1.Text = "Exit";
            this.button1.UseVisualStyleBackColor = false;
```

```
            this.button1.Click += new
System.EventHandler(this.button1_Click);
            //
            // pictureBox1
            //
            this.pictureBox1.Image =
global::cswavrec.Properties.Resources.background;
            this.pictureBox1.Location = new System.Drawing.Point(-3,
-2);
            this.pictureBox1.Name = "pictureBox1";
            this.pictureBox1.Size = new System.Drawing.Size(282,
210);
            this.pictureBox1.TabIndex = 13;
            this.pictureBox1.TabStop = false;
            //
            // Exit
            //
            this.Exit.BackColor = System.Drawing.Color.White;
            this.Exit.DialogResult =
System.Windows.Forms.DialogResult.Cancel;
            this.Exit.Enabled = false;
            this.Exit.Font = new System.Drawing.Font("Tahoma", 9.75F,
System.Drawing.FontStyle.Bold, System.Drawing.GraphicsUnit.Point,
((byte)(0)));
            this.Exit.ForeColor = System.Drawing.Color.Red;
            this.Exit.Location = new System.Drawing.Point(194, 249);
            this.Exit.Name = "Exit";
            this.Exit.Size = new System.Drawing.Size(99, 56);
            this.Exit.TabIndex = 11;
            this.Exit.Text = "Close";
            this.Exit.UseVisualStyleBackColor = false;
            this.Exit.Click += new
System.EventHandler(this.Exit_Click);
            //
            // Form1
            //
            this.AutoScaleBaseSize = new System.Drawing.Size(5, 13);
            this.AutoScroll = true;
```

```
            this.BackColor =
System.Drawing.Color.FromArgb(((int)(((byte)(192)))),
((int)(((byte)(255)))), ((int)(((byte)(192)))));
            this.ClientSize = new System.Drawing.Size(384, 310);
            this.Controls.Add(this.button1);
            this.Controls.Add(this.pictureBox1);
            this.Controls.Add(this.On);
            this.Controls.Add(this.Exit);
            this.Controls.Add(this.Call);
            this.Controls.Add(this.textBox1);
            this.Controls.Add(this.label1);
            this.FormBorderStyle =
System.Windows.Forms.FormBorderStyle.FixedSingle;
            this.Icon =
((System.Drawing.Icon)(resources.GetObject("$this.Icon")));
            this.MaximizeBox = false;
            this.Name = "Form1";
            this.StartPosition =
System.Windows.Forms.FormStartPosition.CenterScreen;
            this.Text = "VoIP";
            this.Load += new System.EventHandler(this.Form1_Load_1);

((System.ComponentModel.ISupportInitialize)(this.pictureBox1)).EndIni
t();
            this.ResumeLayout(false);
            this.PerformLayout();

            }
            #endregion

            /// <summary>
            /// The main entry point for the application.
            /// </summary>
            ///

            [STAThread]
            static void Main()
            {
                  Application.Run(new Form1());
```

```
            }
            #endregion

            #region Voice_In()
            private void Voice_In()
            {
                  byte[] br;
            r.Bind(new IPEndPoint(IPAddress.Any, ReceivePort));
                  while (true)
                  {
                        br = new byte[16384];
                        r.Receive(br);
                        m_Fifo.Write(br, 0, br.Length);
                  }
            }
            #endregion
        #region Voice_Out()

        private void Voice_Out(IntPtr data, int size)
        {
            //for Recorder
            if (m_RecBuffer == null || m_RecBuffer.Length < size)
                m_RecBuffer = new byte[size];
            System.Runtime.InteropServices.Marshal.Copy(data,
m_RecBuffer, 0, size);
            //Microphone ==> data ==> m_RecBuffer ==> m_Fifo
            try
            {
                r.SendTo(m_RecBuffer, new
IPEndPoint(IPAddress.Parse(this.textBox1.Text), SendPort));
            }
            catch
            {

                //MessageBox.Show("This IP is out of local subnet
mask");
                //KillTask("listener");
                //KillTask("voip");
```

```
            }

        }

        #endregion



//*****************************************************************
*************//
        private WaveOutPlayer m_Player;
        private WaveInRecorder m_Recorder;
        private FifoStream m_Fifo = new FifoStream();
        private byte[] m_PlayBuffer;
        private byte[] m_RecBuffer;

        private void button3_Click(object sender, EventArgs e)
        {

        }

        private void Start()
        {
            try
            {
                WaveFormat fmt = new WaveFormat(44100, 16, 2);
                m_Player = new WaveOutPlayer(-1, fmt, 16384, 3, new
BufferFillEventHandler(Filler));
                m_Recorder = new WaveInRecorder(-1, fmt, 16384, 3,
new BufferDoneEventHandler(Voice_Out));
            }
            catch
            {
                Stop();
                throw;
            }
        }
```

```
        private void Stop()
        {
            if (connected == true && ReadfromFile("c:/receive.txt")
== "1")
            {
                WriteinFile("c:/reply.txt", "c:/disconnect");
                System.Diagnostics.Process.Start("c:/reply.exe");
            }
            else
                if (connected == false &&
ReadfromFile("c:/receive.txt") == "1")
                {
                    WriteinFile("c:/reply.txt", "c:/reject");
                    System.Diagnostics.Process.Start("c:/reply.exe");
                }

            WriteinFile("c:/receive.txt", "0");
            if (m_Player != null)
                try
                {
                    m_Player.Dispose();
                }
                finally
                {
                    m_Player = null;
                }
            if (m_Recorder != null)
                try
                {
                    m_Recorder.Dispose();
                }
                finally
                {
                    m_Recorder = null;
                }
            m_Fifo.Flush(); // clear all pending data
            WriteinFile("c:/status.txt", "disconnect");
            receive = false;
            connected = false;
```

```
        }

        private void Filler(IntPtr data, int size)
        {
            if (m_PlayBuffer == null || m_PlayBuffer.Length < size)
                m_PlayBuffer = new byte[size];
            if (m_Fifo.Length >= size)
                m_Fifo.Read(m_PlayBuffer, 0, size);
            else
                for (int i = 0; i < m_PlayBuffer.Length; i++)
                    m_PlayBuffer[i] = 0;
            System.Runtime.InteropServices.Marshal.Copy(m_PlayBuffer,
0, data, size);
            // m_Fifo ==> m_PlayBuffer==> data ==> Speakers
        }

        private void KillTask(string procname)
        {
            foreach (Process p in
System.Diagnostics.Process.GetProcessesByName(procname))
            {
                try
                {
                    p.Kill();
                    p.WaitForExit(); // possibly with a timeout
                }
                catch (Win32Exception winException)
                {
                    // process was terminating or can't be terminated
- deal with it
                }
                catch (InvalidOperationException invalidException)
                {
                    // process has already exited - might be able to
let this one go
                }
            }
        }
```

```
        private void On_Click(object sender, EventArgs e)
        {
            if (On.Text == "Power ON")
            {
                Call.Enabled = true;
                Exit.Enabled = true;
                On.Text = "Power OFF";
                System.Diagnostics.Process.Start("c:\\listener.exe");

            }
            else
            {
                Call.Enabled = false;
                Exit.Enabled = false;
                On.Text = "Power ON";
                foreach (Process p in
System.Diagnostics.Process.GetProcessesByName("listener"))
                {
                    try
                    {
                        p.Kill();
                        p.WaitForExit(); // possibly with a timeout
                    }
                    catch (Win32Exception winException)
                    {
                        // process was terminating or can't be
terminated - deal with it
                    }
                    catch (InvalidOperationException
invalidException)
                    {
                        // process has already exited - might be able
to let this one go
                    }
                }
            }
        }
```

```
        private void Call_Click(object sender, EventArgs e)
        {
            On.Enabled = false;
            run = true;
            Call.Enabled = false;
            textBox1.Enabled = false;
            WriteinFile("c:/status.txt", "busy");
            string state = ReadfromFile("c:/receive.txt");
            if (state == "0")
            {
                WriteinFile("c:/ip.txt", this.textBox1.Text);
                CreatePort();
                System.Diagnostics.Process.Start("c:/CallSetup.exe");
            }
            else
                if (state == "1")
                {
                    receive = true;
                    int i;
                    WriteinFile("c:/reply.txt", "c:/connected");
                    System.Diagnostics.Process.Start("c:/reply.exe");
                    textBox1.Text = ReadfromFile("c:/ip.txt");
                    string ports = ReadfromFile("c:/ports.txt");

                    for (i = 0; i <= 5; i++)
                    {
                        if (ports[i] == ':')
                        {
                            break;
                        }
                    }

                    SendPort = int.Parse(ports.Substring(0, i));
                    ReceivePort = int.Parse(ports.Substring(i + 1));
;

                }
            if (connected == false)
            {
```

```
                r = new Socket(AddressFamily.InterNetwork,
SocketType.Dgram, ProtocolType.Udp);
                t = new Thread(new ThreadStart(Voice_In));
                t.Start();
                connected = true;
            }
            Start();
        }

        private void CreatePort()
        {
            Random r = new Random();
            ReceivePort = r.Next(1025, 32000);
            SendPort = r.Next(1025, 32000);

            WriteinFile("c:/ports.txt", ReceivePort.ToString() + ":"
+ SendPort.ToString());
        }

        private string ReadfromFile(string FilePath)
        {
            FileStream fs = new FileStream(FilePath, FileMode.Open);
            byte[] fileContents = new byte[fs.Length];
            fs.Read(fileContents, 0, (int)fs.Length);
            string receiv =
Encoding.UTF8.GetString(fileContents).Substring(1);
            fs.Close();
            return receiv;
        }

        private void WriteinFile(string FilePath, string content)
        {
            FileStream fs = new FileStream(FilePath,
  FileMode.Create);
            BinaryWriter bw = new BinaryWriter(fs);
            bw.Write(content);
            bw.Close();
            fs.Close();
        }
```

```
        private void Exit_Click(object sender, EventArgs e)
        {
            Call.Enabled = true;
            textBox1.Enabled = true;
            Stop();
            t.Abort();
            r.Close();
            return;
        }

        private void Form1_Load_1(object sender, EventArgs e)
        {
           if(
System.Diagnostics.Process.GetProcessesByName("voip").Length>1)//here
to determining th NO of processes that running now.
            {
                try
                {
                    Application.Exit();
                }
                catch (Win32Exception winException)
                {
                    // process was terminating or can't be terminated
- deal with it
                }
                catch (InvalidOperationException invalidException)
                {
                    // process has already exited - might be able to
let this one go
                }
            }
            WriteinFile("c:/receive.txt", "0");
            WriteinFile("c:/run.txt", "1");
            WriteinFile("c:/status.txt", "disconnect");
            KillTask("listener");
        }

        private void Form1_Closing(object sender, EventArgs e)
```

```
        {
            button1_Click(sender, e);

        }

        private void button1_Click(object sender, EventArgs e)
        {
            WriteinFile("c:/run.txt", "0");
            Stop();
            t.Abort();
            r.Close();
            KillTask("listener");
            KillTask("voip");
            Application.Exit();
        }

    }
}
```

II- Listener code

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.Threading;
using System.IO;
using System.Diagnostics;

namespace WindowsApplication2
{
    class ThreadedTcpListener
    {
        private TcpListener client;
        public ThreadedTcpListener()
        {
            client = new TcpListener(9050);
            client.Start();
```

```
            Console.WriteLine("Waiting for a call...");

            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()
        {
            ThreadedTcpListener server = new ThreadedTcpListener();
        }
    }
    class ConnectionThread
    {
        public TcpListener threadListener;
        private void WriteinFile(string FilePath, string content)
        {
            FileStream fs = new FileStream(FilePath,
  FileMode.Create);
            BinaryWriter bw = new BinaryWriter(fs);
            bw.Write(content);
            bw.Close();
            fs.Close();
        }
        public void HandleConnection()
        {
            string income;
            int recv;
            int i;
            byte[] data = new byte[1024];
```

```
            TcpClient client = threadListener.AcceptTcpClient();
            NetworkStream ns = client.GetStream();
            Console.WriteLine("New call accepted");
            string welcome = "Welcome";
            data = Encoding.ASCII.GetBytes(welcome);
            ns.Write(data, 0, data.Length);
                data = new byte[1024];
                recv = ns.Read(data, 0, data.Length);
                income = Encoding.ASCII.GetString(data, 0, recv);
                if (income == "c:/disconnect" || income ==
"c:/connected" || income == "c:/reject")
                {
                    System.Diagnostics.Process.Start(income);
                    if(income=="c:/connected")
                        WriteinFile("c:/receive.txt", "1");
                    else
                        WriteinFile("c:/receive.txt", "0");
                }
                else
                {
                    string ip =
client.Client.RemoteEndPoint.ToString();

                    for (i = 5; i <= 15; i++)
                    {
                        if (ip[i] == ':')
                        {
                            break;
                        }
                    }
                    ip = ip.Substring(0, i);

                    WriteinFile("c:/ip.txt", ip);
                    string status = ReadfromFile("c:/status.txt");
                    if (status != "busy")
                    {
                        WriteinFile("c:/ports.txt", income);
                        WriteinFile("c:/receive.txt", "1");
                        WriteinFile("c:/status.txt", "busy");
```

```
System.Diagnostics.Process.Start("c://newcall.exe");
                    }
                    else
                    {
                     Console.WriteLine("  " + ip + "  Trying to call
you");
                    }
                    data = Encoding.ASCII.GetBytes(status);
                    ns.Write(data, 0, data.Length);
                }
            ns.Close();
            client.Close();
            connections--;
        }
        private string ReadfromFile(string FilePath)
        {
            FileStream fs = new FileStream(FilePath, FileMode.Open);
            byte[] fileContents = new byte[fs.Length];
            fs.Read(fileContents, 0, (int)fs.Length);
            string receiv =
Encoding.UTF8.GetString(fileContents).Substring(1);
            fs.Close();
            return receiv;
        }

    }

}
```

III- CallSetup code

```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.IO;
class CallSetup
```

```
{

    public static void Main()
    {
        byte[] data = new byte[1024];
        string input, stringData;
        string IP = ReadfromFile("c:\\ip.txt");
        Console.WriteLine("Trying to call with " + IP);
        IPEndPoint ipep = new IPEndPoint(
                IPAddress.Parse(IP), 9050);
        Socket server = new Socket(AddressFamily.InterNetwork,
                SocketType.Stream, ProtocolType.Tcp);
        try
        {
            server.Connect(ipep);
        }
        catch (SocketException e)
        {
            Console.WriteLine("Unable to connect to target.");

System.Diagnostics.Process.Start("c:\\unabletoconnect.exe");
            return;
        }
        int recv = server.Receive(data);
        stringData = Encoding.ASCII.GetString(data, 0, recv);
        Console.WriteLine(stringData);
            input = ReadfromFile("c:\\ports.txt
            server.Send(Encoding.ASCII.GetBytes(input));
            data = new byte[1024];
            recv = server.Receive(data);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            if (stringData == "busy")
                System.Diagnostics.Process.Start("c:\\busy.exe");
        server.Shutdown(SocketShutdown.Both);
        server.Close();

    }
```

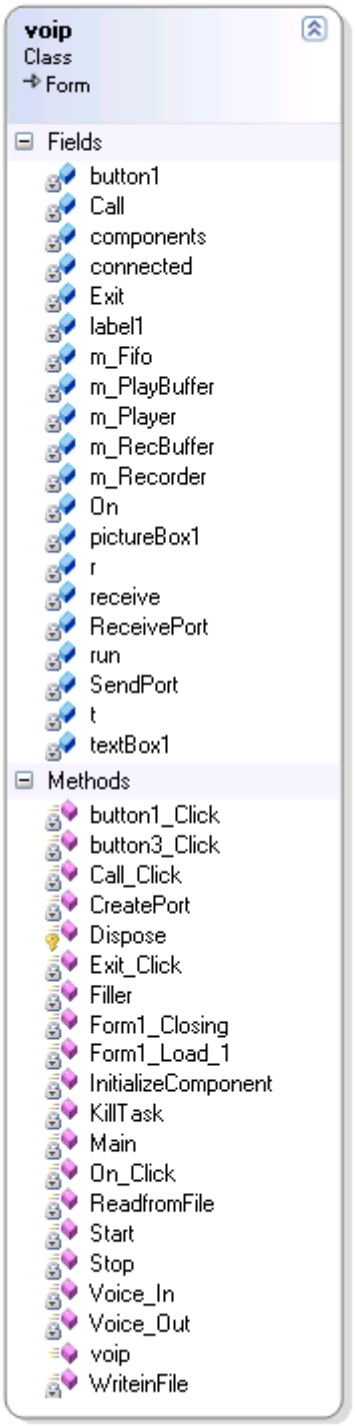
```
    private static string ReadfromFile(string FilePath)
    {
        FileStream fs = new FileStream(FilePath, FileMode.Open);
        byte[] fileContents = new byte[fs.Length];
        fs.Read(fileContents, 0, (int)fs.Length);
        string receiv =
Encoding.UTF8.GetString(fileContents).Substring(1);
        fs.Close();
        return receiv;
    }
    private static void WriteinFile(string FilePath, string content)
    {
        FileStream fs = new FileStream(FilePath,
FileMode.Create);
        BinaryWriter bw = new BinaryWriter(fs);
        bw.Write(content);
        bw.Close();
        fs.Close();
    }
}
```

IV- Reply code

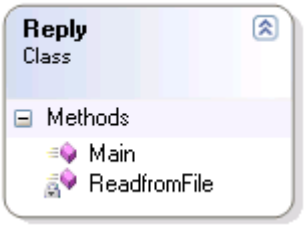
```
using System;
using System.Net;
using System.Net.Sockets;
using System.Text;
using System.IO;
class Reply
{

    public static void Main()
    {
        byte[] data = new byte[1024];
        string input, stringData;


        string IP = ReadfromFile("c:\\ip.txt");
```

```
            Console.WriteLine("Trying to call with " + IP);
            IPEndPoint ipep = new IPEndPoint(
                    IPAddress.Parse(IP), 9050);
            Socket server = new Socket(AddressFamily.InterNetwork,
                    SocketType.Stream, ProtocolType.Tcp);
            try
            {
                server.Connect(ipep);
            }
            catch (SocketException e)
            {
                return;
            }
            int recv = server.Receive(data);
            stringData = Encoding.ASCII.GetString(data, 0, recv);
            Console.WriteLine(stringData);
            input = ReadfromFile("c:\\reply.txt");
            server.Send(Encoding.ASCII.GetBytes(input));
            server.Shutdown(SocketShutdown.Both);
            server.Close();
        }

        private static string ReadfromFile(string FilePath)
        {
            FileStream fs = new FileStream(FilePath, FileMode.Open);
            byte[] fileContents = new byte[fs.Length];
            fs.Read(fileContents, 0, (int)fs.Length);
            string receiv =
    Encoding.UTF8.GetString(fileContents).Substring(1);
            fs.Close();
            return receiv;
        }

    }
```

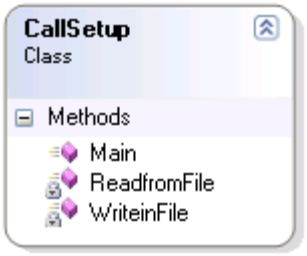
B- Classes' diagrams



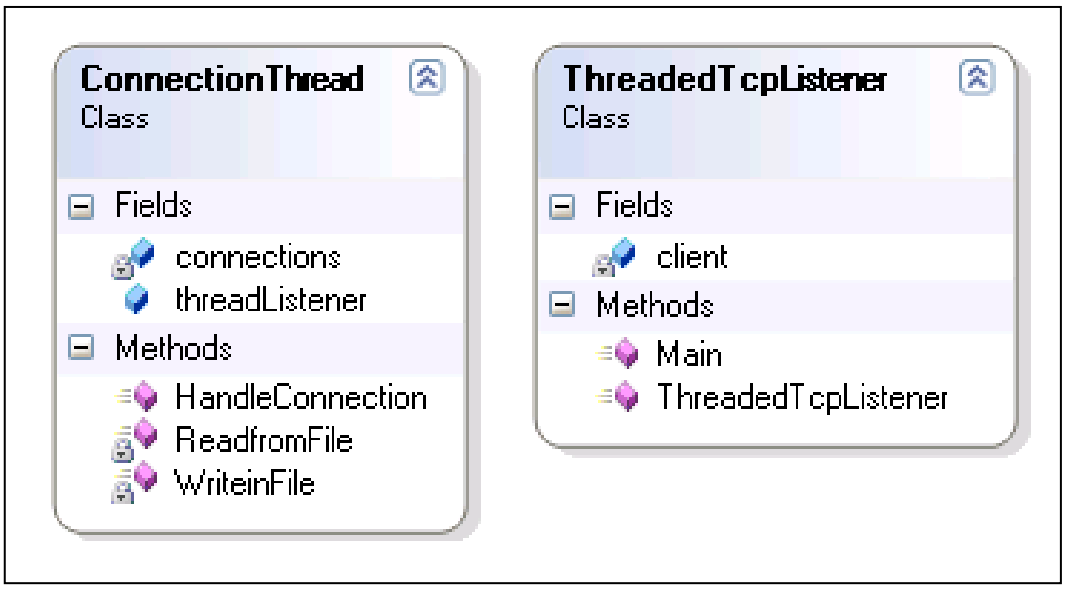
Class diagram of voip program.



Class diagram of Reply module



Class diagram of CallSetup module



Class diagram of Listener module