**Đề thi CTK44 (2022-2023)**

Cho đối tượng thiết bị gồm những thuộc tính được mô hả như sau:

+ Mã thiết bị (kiểu số nguyên không dấu)

+ Tên thiết bị (kiểu chuỗi)

+ Năm sản xuất (kiểu số nguyên không dấu)

+ Giá thiết bị (kiểu số thực)

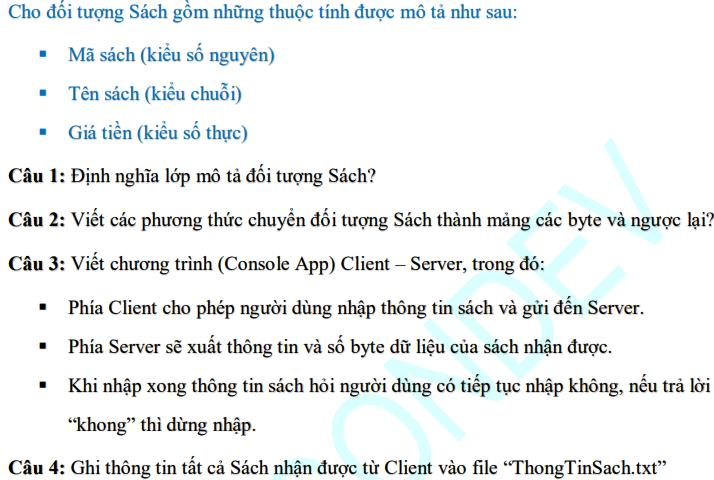
Câu 1: Định nghĩ lớp mô tả đối tượng Thiết Bị?

Câu 2: Viết các phương thức chuyển đổi đối tượng Thiết Bị thành mảng các byte và ngược lại?

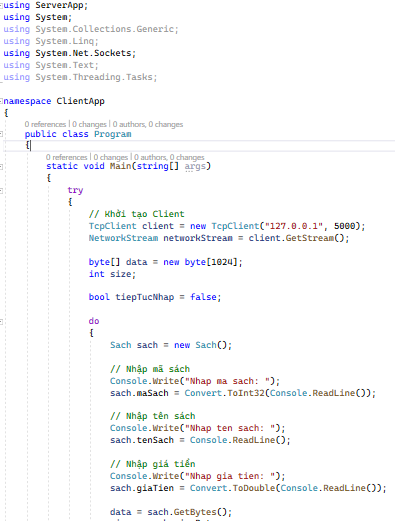
Câu 3: Viết chương trình Client-Server trong đó: phía Client cho phép nhập vào thông tin thiết bị và gửi đến Server, phía Server cho phép hiển thị thông tin và số byte dữ liệu nhận được

**- > Tương tự Lab 4**

**Đề thi khóa trước**

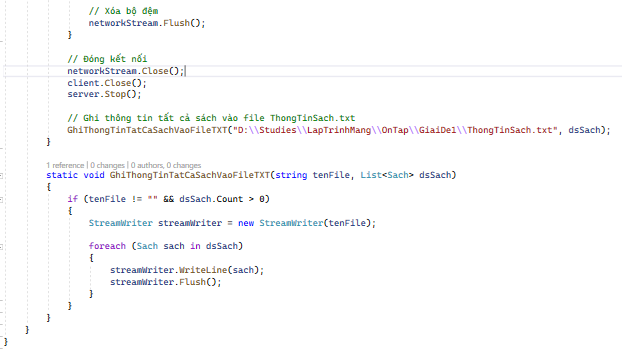


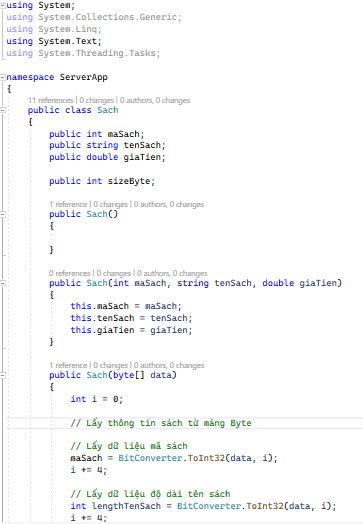
**Client**

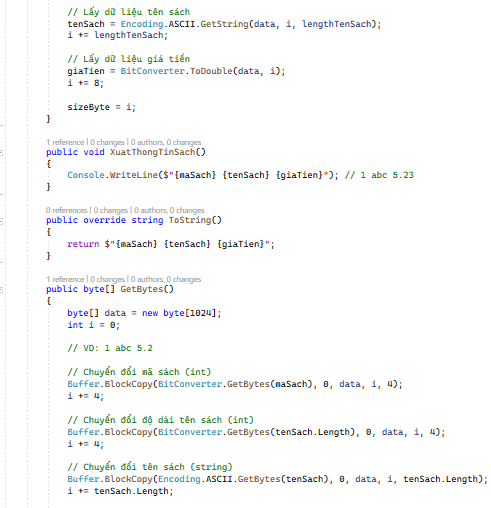


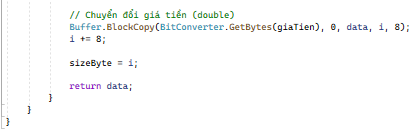




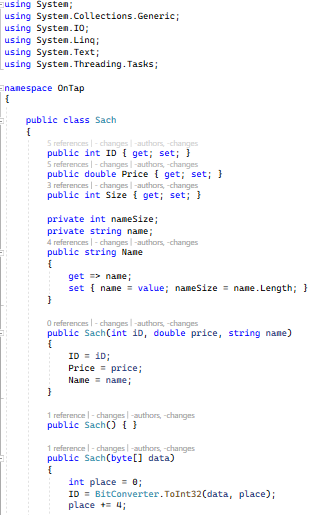


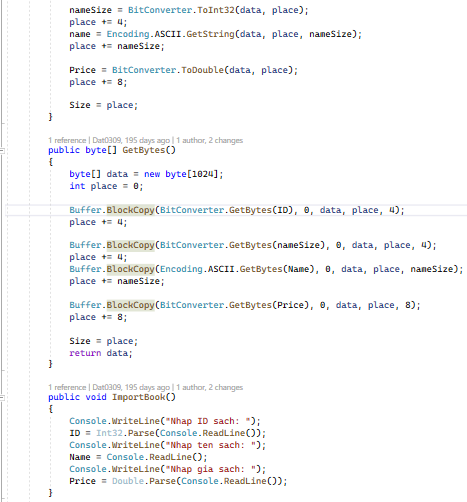


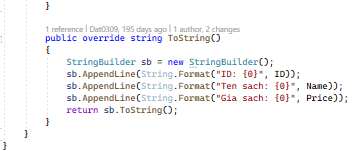




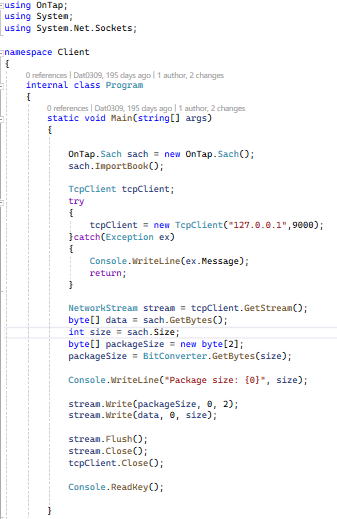
**Lab04 Gửi thông tin Object**



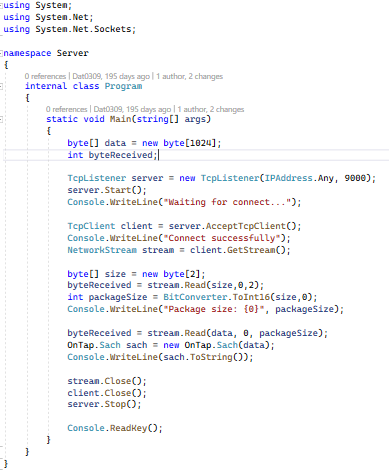




Client



Server



Lab06 chat client – server

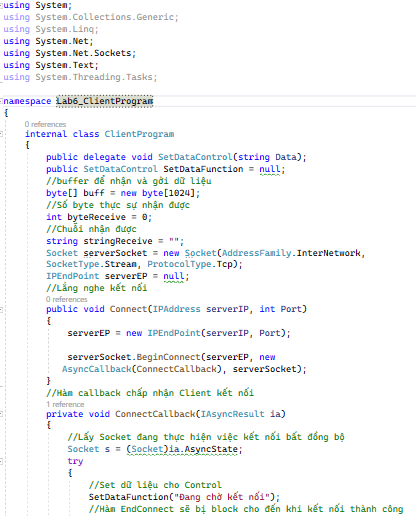
Form Client

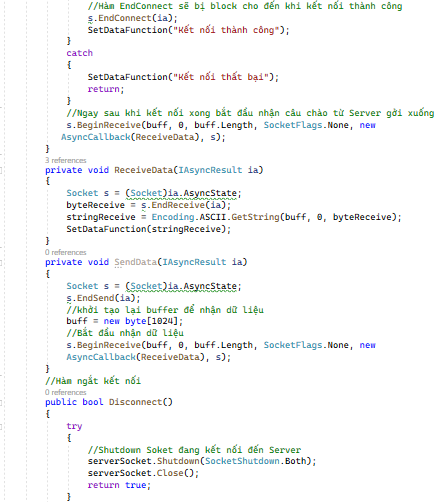


Form Server



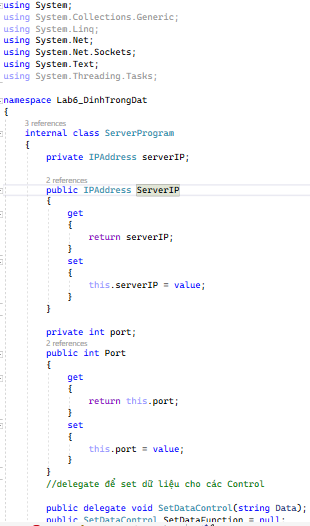
Client Program

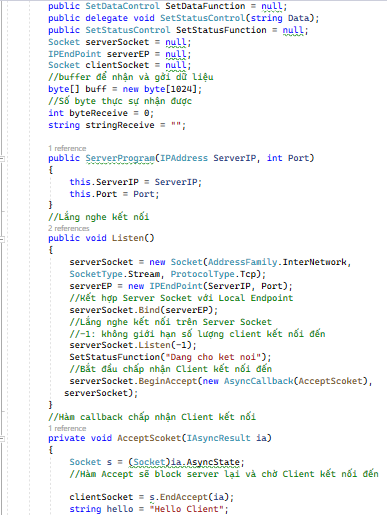


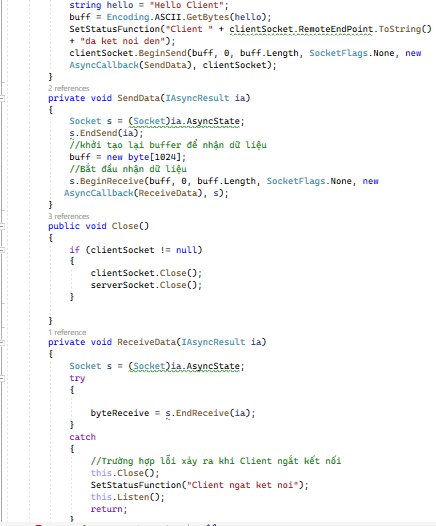


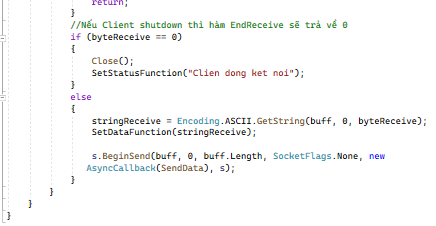


Server Program









Gửi Object bằng cả TCP và UDP

ClientClass

Text

Description automatically generated

Text

Description automatically generated

Sinhvien.cs

Text

Description automatically generated

Text

Description automatically generated

ServerClass

Text

Description automatically generated

Text

Description automatically generated

SimpleClientTCP

Text

Description automatically generated

SimpleServerTCP

Text

Description automatically generated

SimpleClientUDP

Text

Description automatically generated

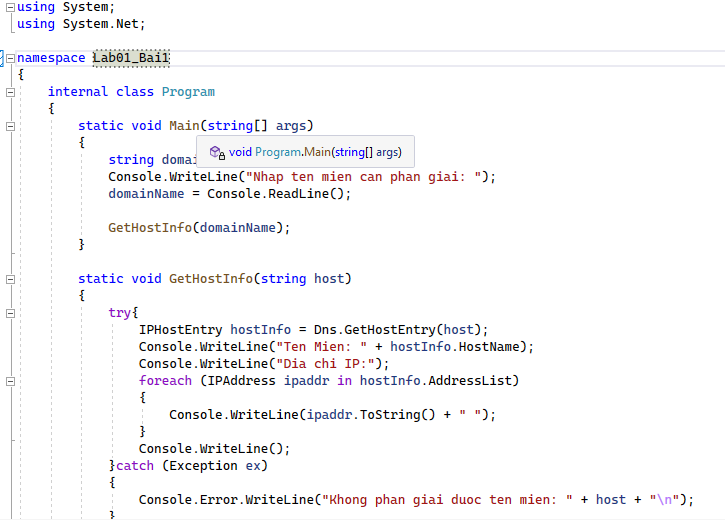
SimpleServerUDP

Text

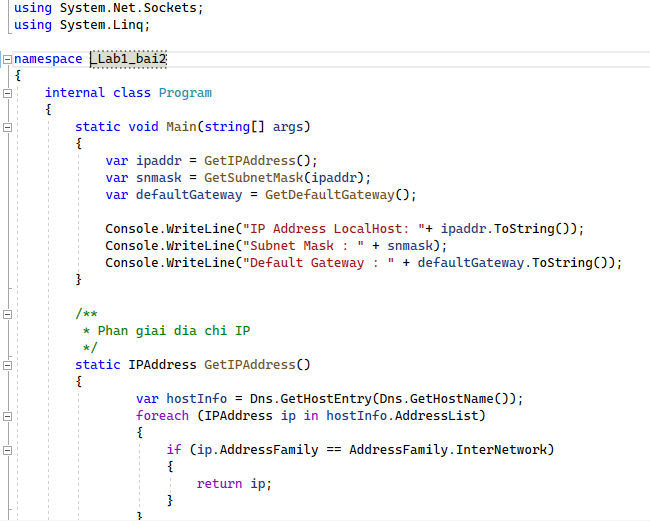
Description automatically generated

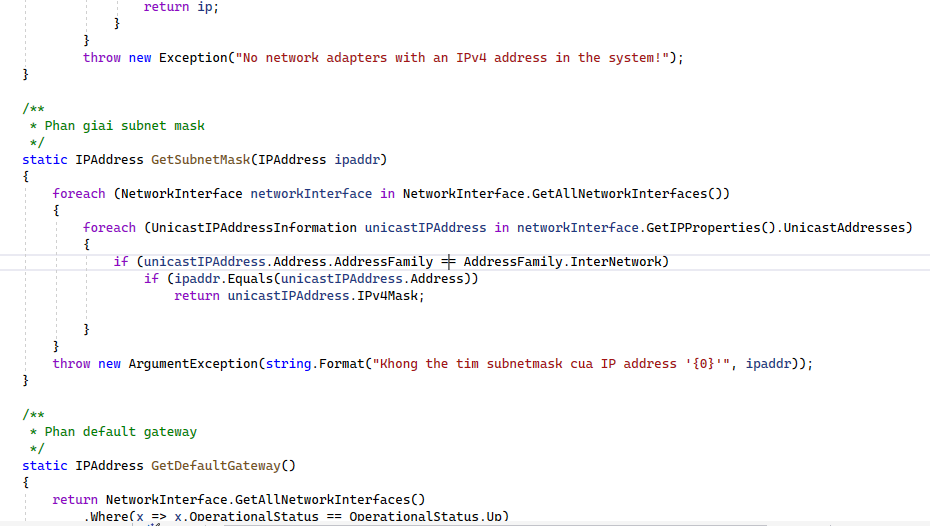
LAB 1

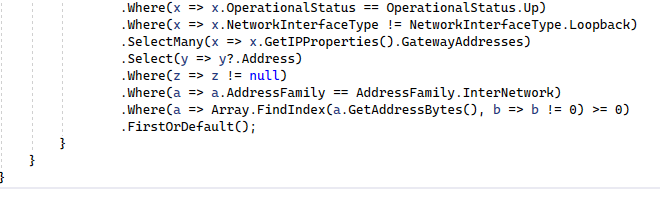
Bài 1

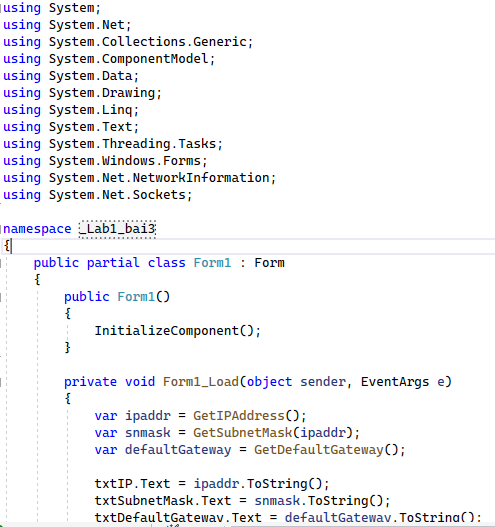


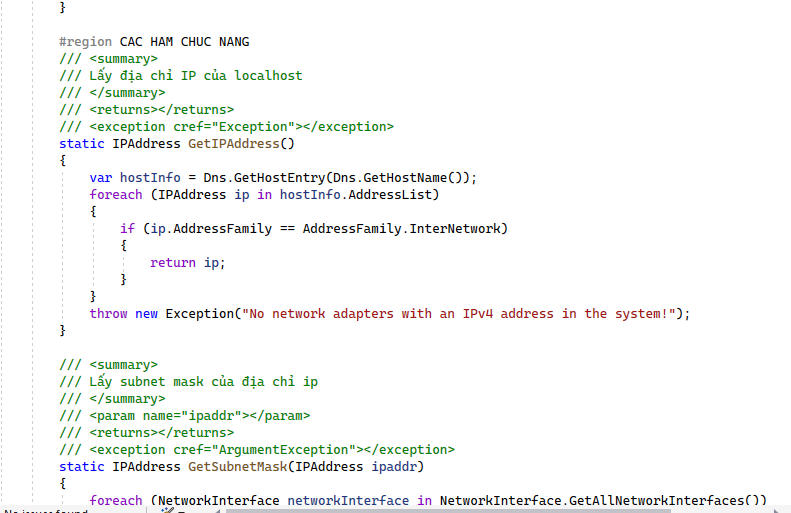
Bài 2

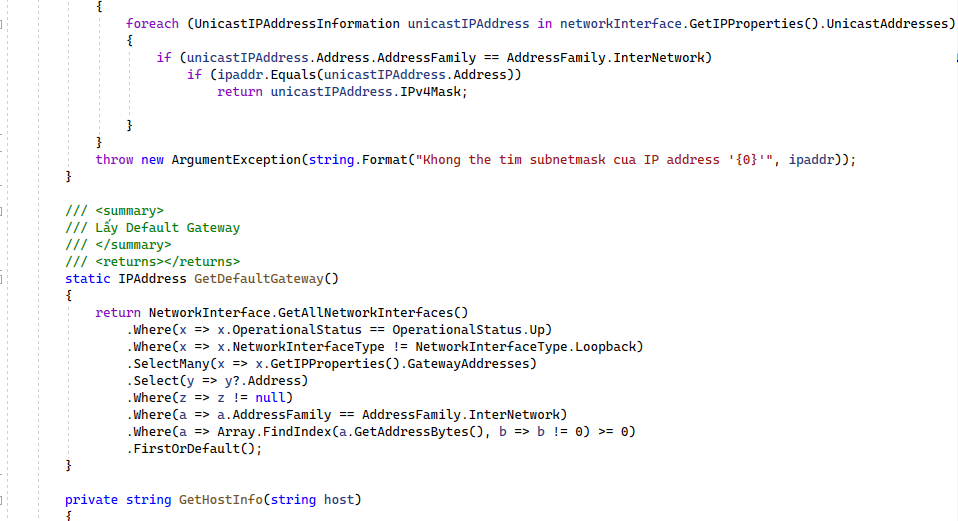


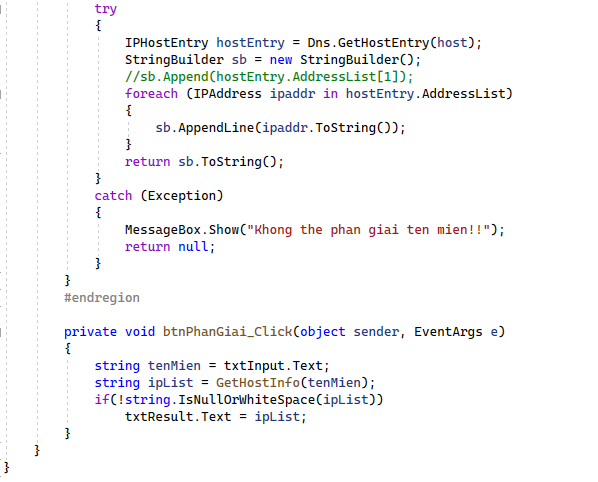


Bài 3: Form



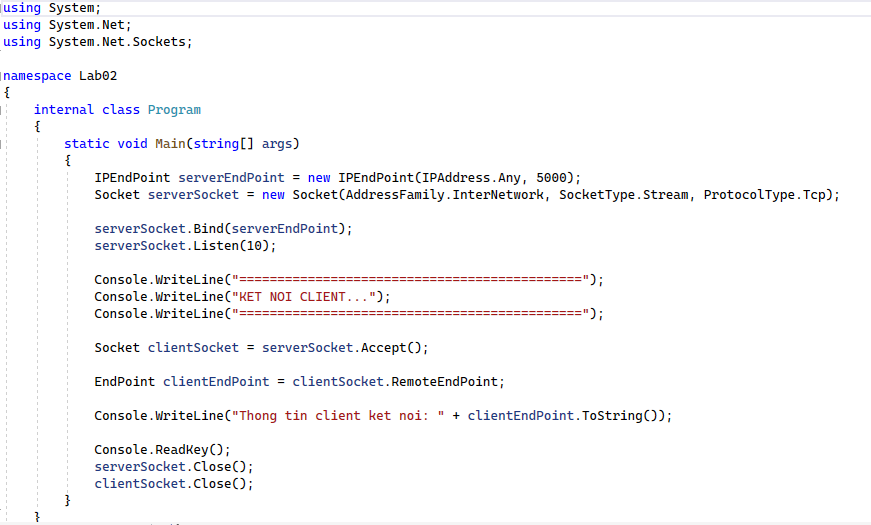




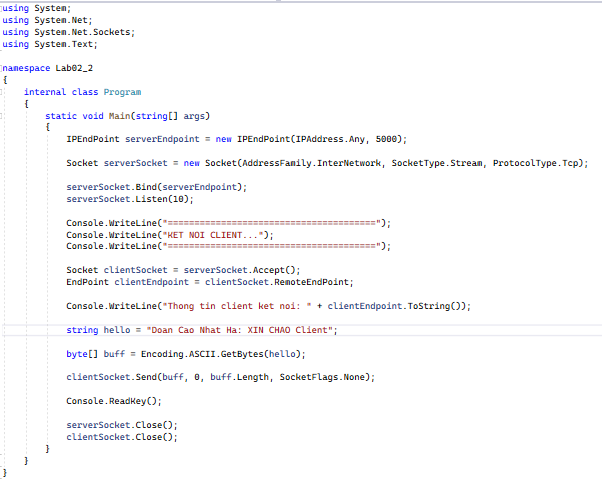


LAB 2

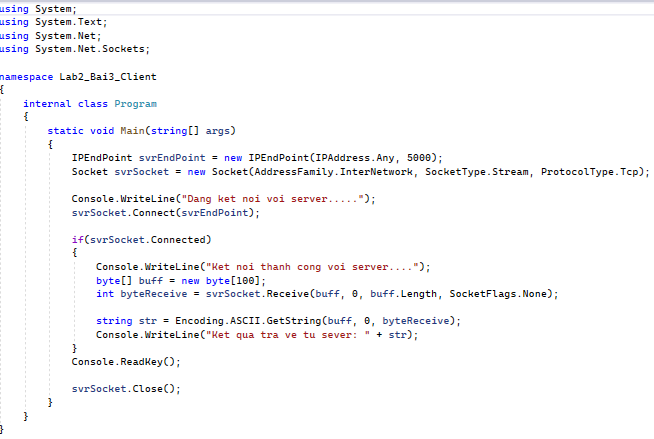
Bài 1



Bài 2



Bài 3: Client

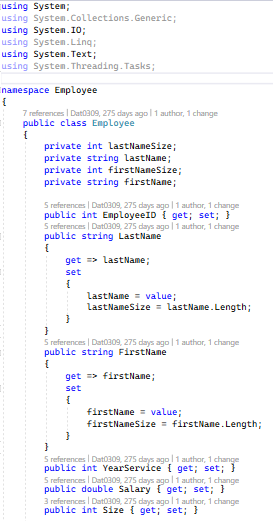


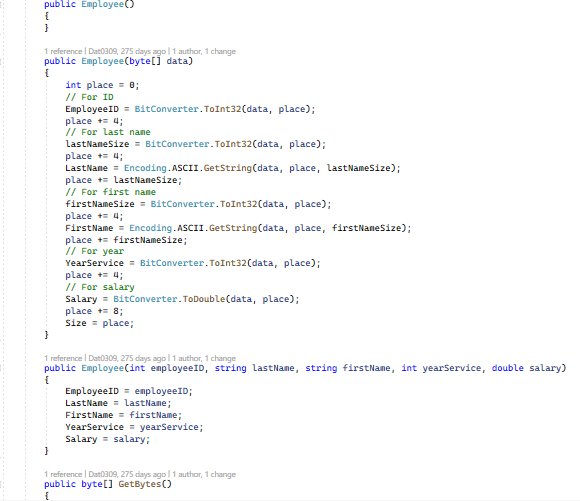
Server

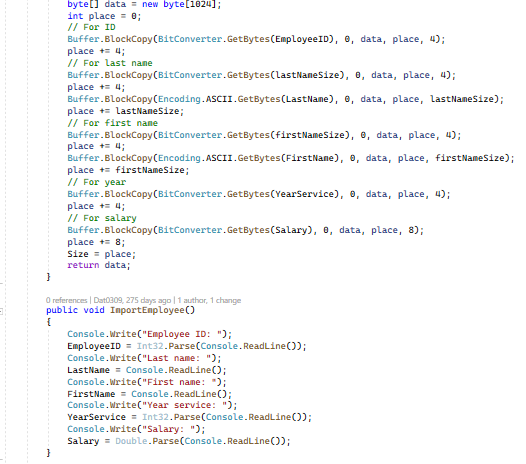


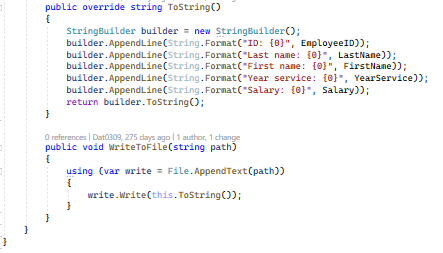
LAB 4

Employee

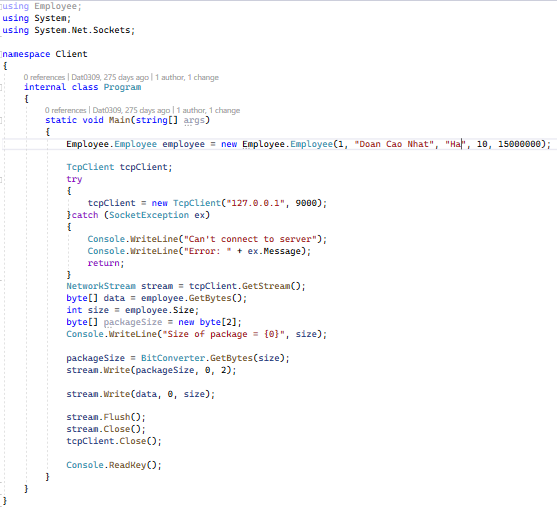








Client



Server



**Lab5**

**ConsoleLogger**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace TcpThread

{

class ConsoleLogger : ILogger

{

private static Mutex mutex = new Mutex();

public void writeEntry(ArrayList entry)

{

mutex.WaitOne();

IEnumerator line = entry.GetEnumerator();

while (line.MoveNext())

Console.WriteLine(line.Current);

Console.WriteLine();

mutex.ReleaseMutex();

}

public void writeEntry(String entry)

{

mutex.WaitOne();

Console.WriteLine(entry);

Console.WriteLine();

mutex.ReleaseMutex();

}

}

}

**EchoProtoco**l

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace TcpThread

{

class EchoProtocol : IProtocol

{

public const int BUFFSIZE = 32;

private Socket clntSock;

private ILogger logger;

public EchoProtocol(Socket clntSock, ILogger logger)

{

this.clntSock = clntSock; this.logger = logger;

}

public void handleClient()

{

ArrayList entry = new ArrayList();

entry.Add("Client address and port = " + clntSock.RemoteEndPoint);

entry.Add("Thread = " + Thread.CurrentThread.GetHashCode());

try

{

int recvMsgSzie;

int totalBytesEchoed = 0;

byte[] recvBuffer = new byte[BUFFSIZE];

try

{

while ((recvMsgSzie = clntSock.Receive(recvBuffer, 0,

recvBuffer.Length, SocketFlags.None)) > 0)

{

clntSock.Send(recvBuffer, 0, recvMsgSzie, SocketFlags.None);

totalBytesEchoed += recvMsgSzie;

}

}

catch (SocketException se)

{

entry.Add(se.ErrorCode + ": " + se.Message);

}

entry.Add("Client finished; echoed " + totalBytesEchoed + " byte.");

}

catch (SocketException se)

{

entry.Add(se.ErrorCode + ": " + se.Message);

}

clntSock.Close();

logger.writeEntry(entry);

}

}

}

**FileLogge**r

using System;

using System.Collections;

using System.Collections.Generic;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace TcpThread

{

class FileLogger :ILogger

{

private static Mutex mutex = new Mutex();

private StreamWriter output;

public FileLogger(String filename)

{

output = new StreamWriter(filename, true);

} public void writeEntry(ArrayList entry)

{

mutex.WaitOne();

IEnumerator line = entry.GetEnumerator();

while (line.MoveNext())

output.WriteLine(line.Current);

output.Flush();

mutex.ReleaseMutex();

}

public void writeEntry(String entry)

{

mutex.WaitOne();

output.WriteLine(entry);

output.WriteLine();

output.Flush();

mutex.ReleaseMutex();

}

}

}

**Ilogger**

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TcpThread

{

public interface ILogger

{

void writeEntry(ArrayList entry);

void writeEntry(String entry);

}

}

**Iprotocol**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace TcpThread

{

public interface IProtocol

{

void handleClient();

}

}

**Program**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Net;

using System.Net.Sockets;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

namespace TcpThread

{

class Program

{

static void Main(string[] args)

{

if (args.Length != 1)

throw new ArgumentException("Parameter(s): <Port>");

int serverPort = Int32.Parse(args[0]);

TcpListener listener = new TcpListener(IPAddress.Any, serverPort);

ILogger logger = new ConsoleLogger();

listener.Start();

for (; ; )

{

try

{

Socket client = listener.AcceptSocket();

EchoProtocol protocol = new EchoProtocol(client, logger);

Thread thread = new Thread(new

ThreadStart(protocol.handleClient));

thread.Start();

logger.writeEntry("Create and start Thread = " +

thread.GetHashCode());

}

catch (System.IO.IOException e)

{

logger.writeEntry("Error: " + e.Message);

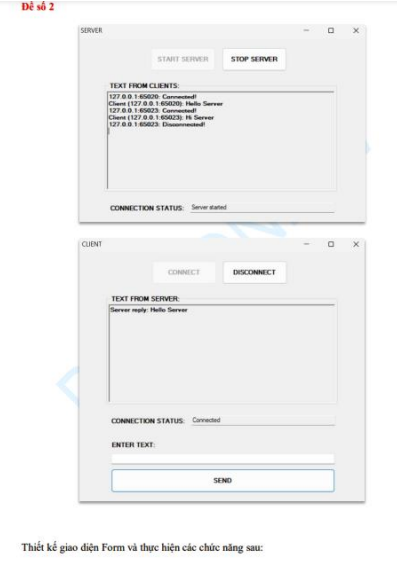
}

}

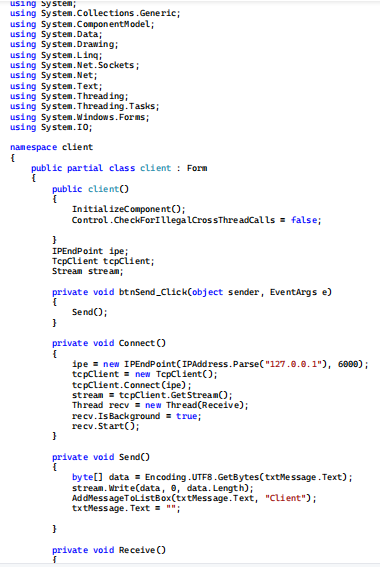
}

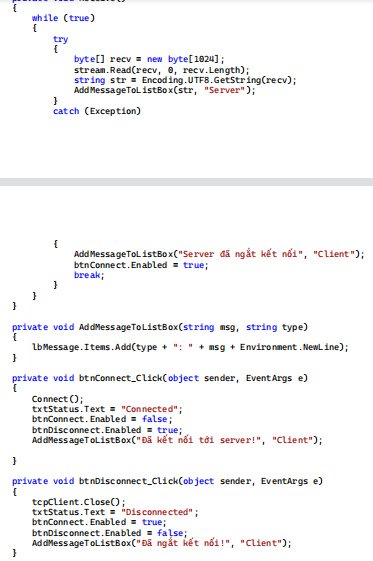
}

}



**Client**





**Server**

