LABORATORIUM SIECI KOMPUTEROWYCH

Data wykonania ćwiczenia:	**19.05.2020**
Rok studiów:	**2**
Semestr:	*4*
Grupa studencka:	**1**
Grupa laboratoryjna:	**B**

Ćwiczenie nr	*9*

Temat: **Aplikacje klient-serwer.**

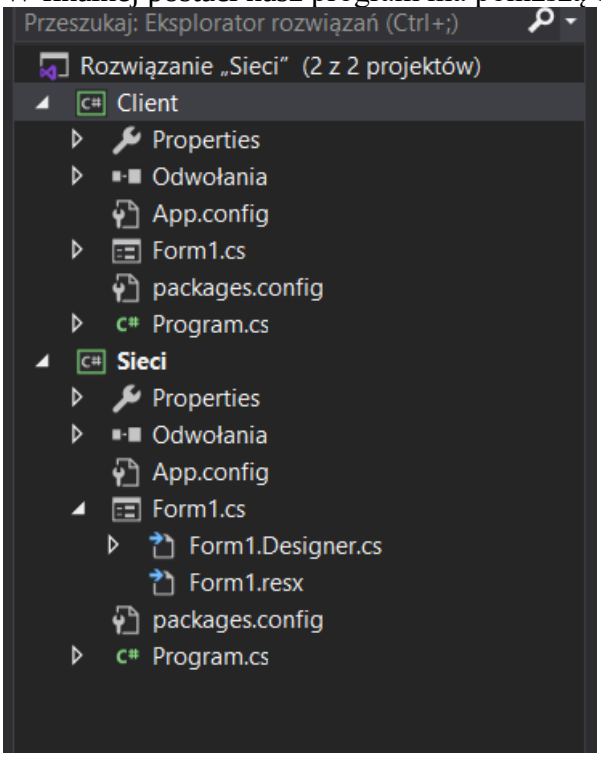
Osoby wykonujące ćwiczenia:

1. Marek Żyła
2. Michał Krzyżowski

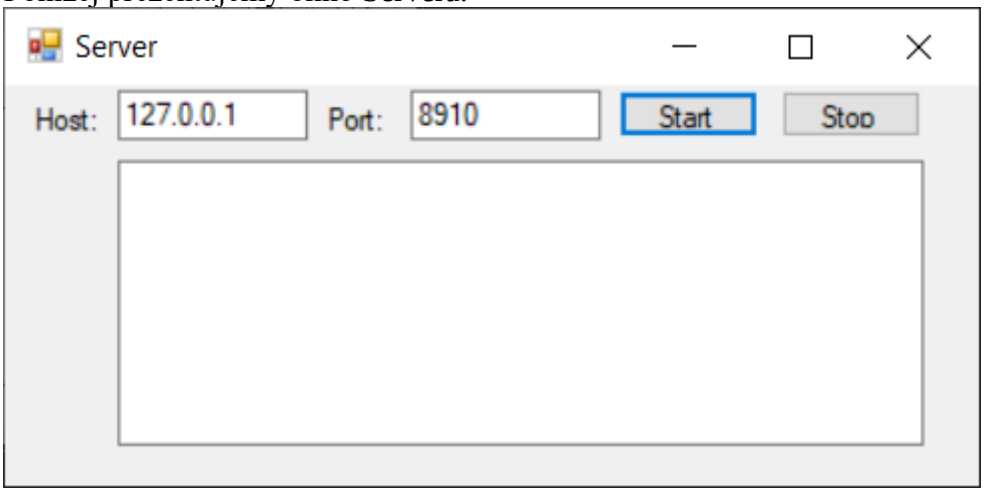
Katedra Informatyki i Automatyki

Na obecnych laboratoriach należało wykonać aplikację klient-server. Aplikacja miała być wykonana w dowolnym języku programowania – my napisaliśmy ją w c#. Celem aplikacji była komunikacja między dwoma użytkownikami, gdzie dodatkowym wymogiem było by aplikacja składała się z 2 programów – żądającego usług i tego który przetwarza.

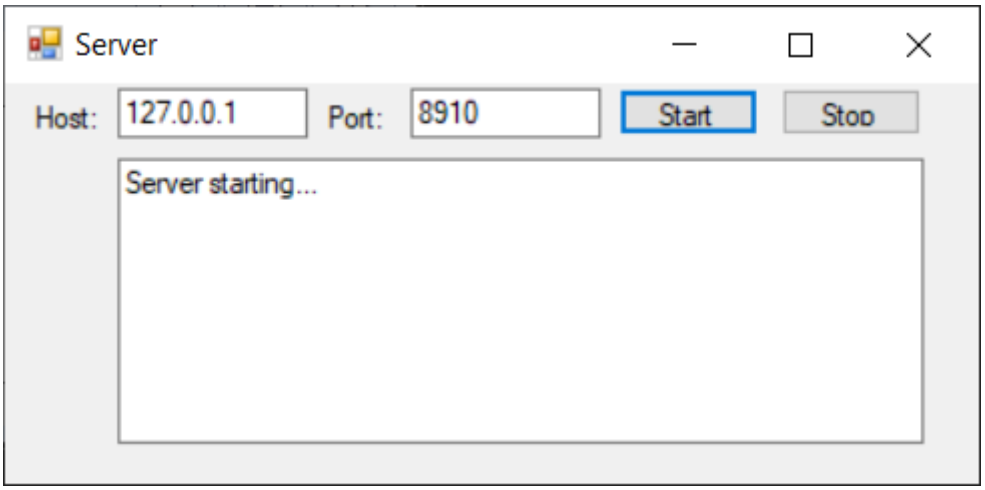
W finalnej postaci nasz program ma poniższą budowę:



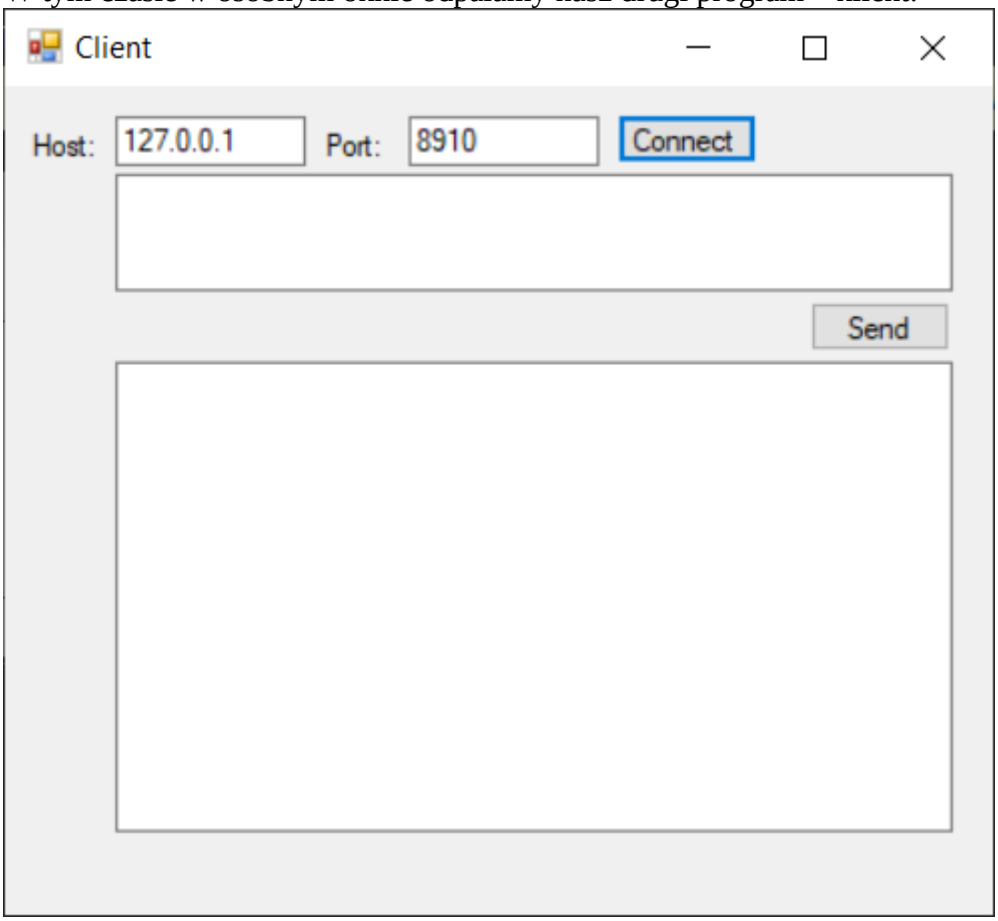
Widzimy tu dwa programy odpowiadające za Server (sieci) i Client (klient).
Poniżej prezentujemy okno Servera:



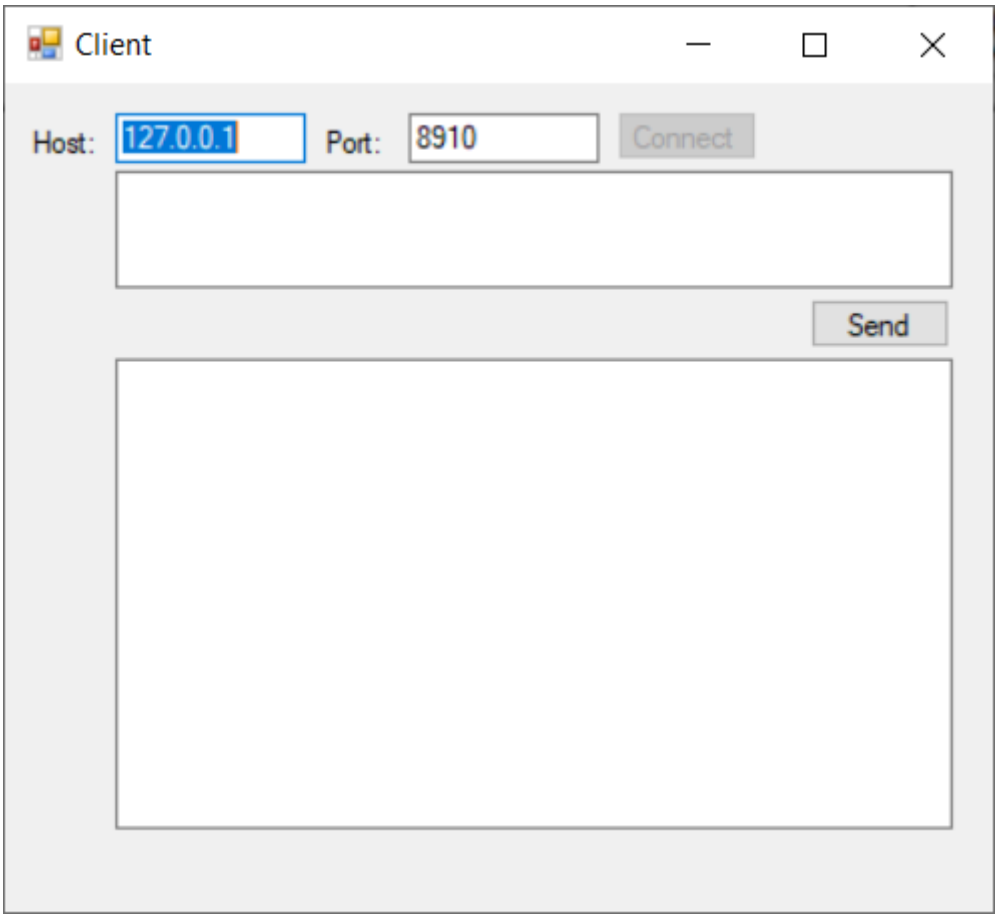
Możemy w tym oknie rozpocząć pracę serwera poprzez naciśnięcie przycisku start:



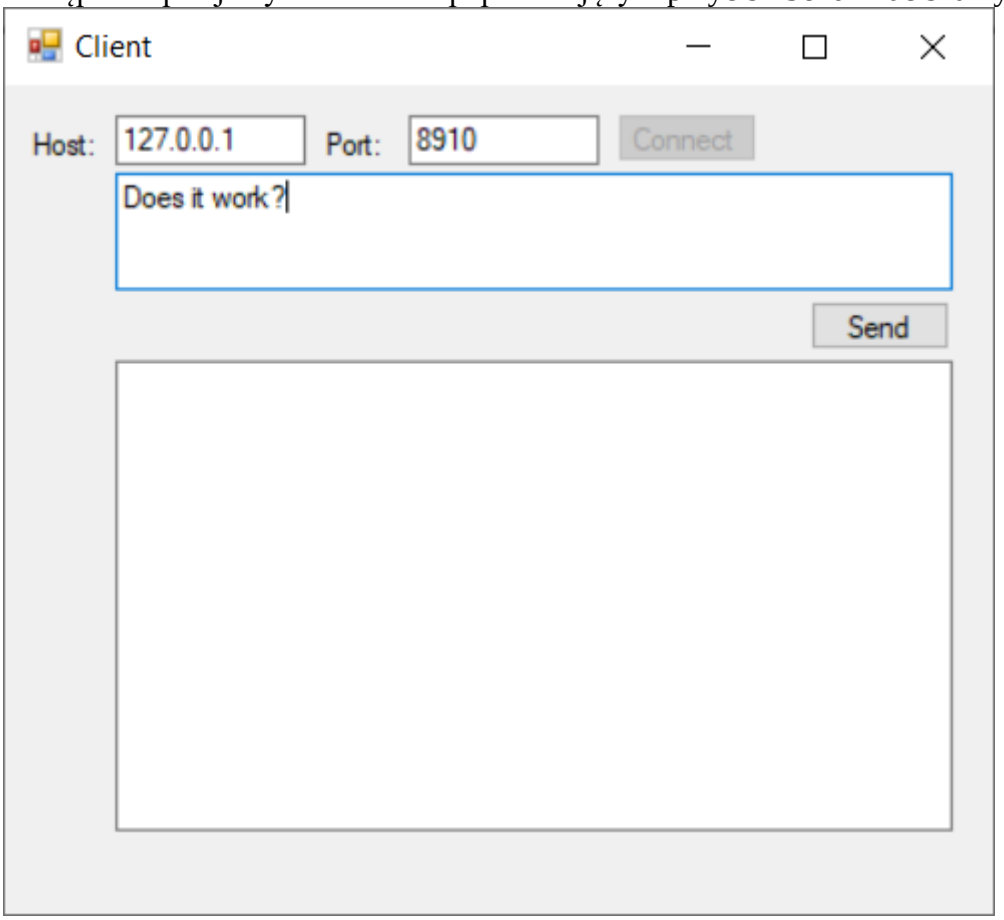
W tym czasie w osobnym oknie odpalamy nasz drugi program – klient:



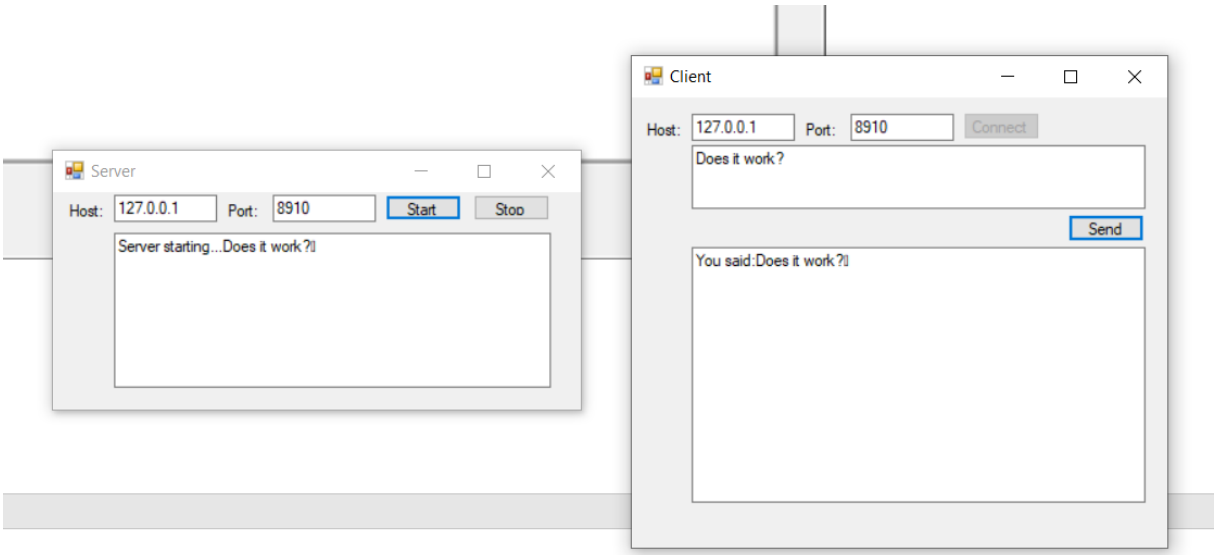
Klikamy przycisk połącz – czego efektem jest zablokowanie dostępu do przycisku.



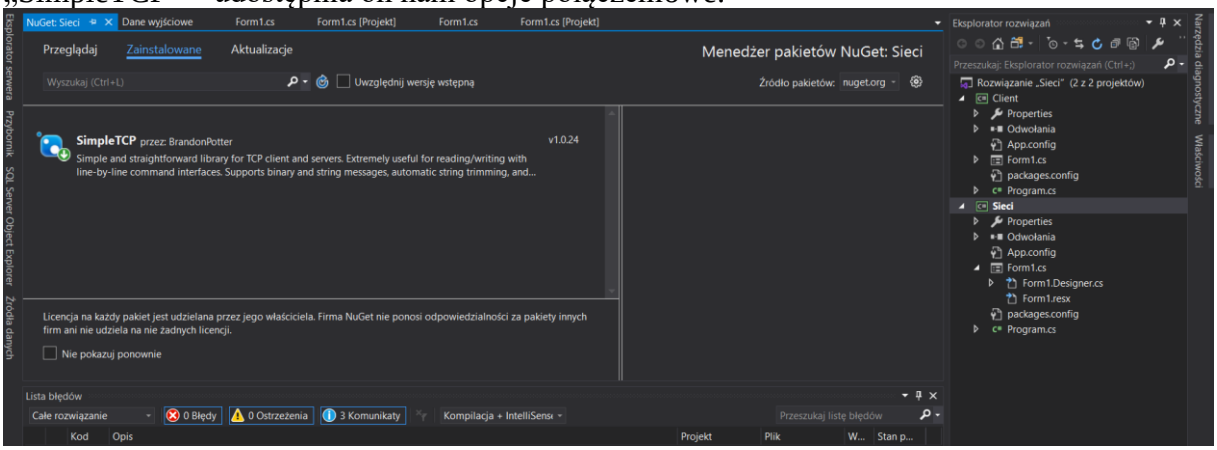
Następnie wpisujemy coś w oknie poprzedzającym przycisk Send i naciskamy na przycisk:



Efekt:



Jak widać komunikacja między serwerem i naszym klientem przebiega sprawnie.
Teraz jeżeli idzie o kod w programie, całość działa dzięki doinstalowanemu pakietkowi NuGet: „SimpleTCP” – udostępnia on nam opcje połączeniowe:



Na samym początku przy wczytywaniu naszego formularzu tworzymy zmienną typu SimpleTcpServer zmienna ta będzie reprezentowała nasz server (wprowadzamy jej podstawowe informacje jak np. kodowanie ustawione na utf-8):

```
        }

        SimpleTcpServer server;
        1 odwołanie
        private void Form1_Load(object sender, EventArgs e)
        {
            server = new SimpleTcpServer();
            server.Delimiter = 0x13;//server
            server.StringEncoder = Encoding.UTF8;
            server.DataReceived += Server_DataReceived;

        }
```

Dopisaliśmy też event o nazwie Server_DataReceived:

```
1 odwołanie
private void Server_DataReceived(object sender, SimpleTCP.Message e)
{
    txtStatus.Invoke((MethodInvoker)delegate ()
    {
        txtStatus.Text += e.MessageString;
        e.ReplyLine(string.Format("You said:{0}",e.MessageString));
    }
    );
}
```

Odpowiada on za poprawne doświetlanie informacji w textboxie.
Poniżej mamy jeszcze obsługę dwóch naszych przycisków na serwerze jeden odpowiada za rozpoczęcie pracy, a drugi za zakończenie:

```
1 odwołanie
private void BtnStart_Click(object sender, EventArgs e)
{
    txtStatus.Text += "Server starting...";
    System.Net.IPAddress ip = System.Net.IPAddress.Parse(txtHost.Text);
    server.Start(ip, Convert.ToInt32(txtPort.Text));
}

1 odwołanie
private void BtnStop_Click(object sender, EventArgs e)
{
    if (server.IsStarted)
        server.Stop();
}
}
```

Jeżeli idzie o klient jest on zbudowany analogicznie do Serwera:

```
19          }
20          SimpleTcpClient client;
21
            1 odwołanie
22          private void BtnConnect_Click(object sender, EventArgs e)
23          {
24              BtnConnect.Enabled = false;
25              client.Connect(txtHost.Text, Convert.ToInt32(txtPort.Text));
26
27
28          }
29
            1 odwołanie
30          private void Form1_Load(object sender, EventArgs e)
31          {
32              client = new SimpleTcpClient();
33              client.StringEncoder = Encoding.UTF8;
34              client.DataReceived += Client_DataReceived;
35          }
36
```

Zmienia się nasz typ którym jest SimpleTcpClient i pojawia się metoda Connect odpowiadająca za połączenie - aktywowana po naciśnięciu przycisku connect.
Jest też dopisany „wyłapywacz” danych podobnie jak na serwerze:

```
1 odwołanie
private void Client_DataReceived(object sender, SimpleTCP.Message e)
{
    txtStatus.Invoke((MethodInvoker)delegate ()
    {
        txtStatus.Text += e.MessageString;
    }
    );
}
```

No i dodatkowo okodowane działanie przycisku „send” z ustawiony opóźnieniem 3-sek w wyświetlaniu.

```
1 odwołanie
private void BtnSend_Click(object sender, EventArgs e)
{
    client.WriteLineAndGetReply(txtMessage.Text, TimeSpan.FromSeconds(3));
}
```