Name Surname : Arda Dumanoğlu Sign:
Number : 190315072

190315072 ARDA DUMANOĞLU HOMEWORK REPORT

First, as our Lecturer said we can only use System libraries not any other additional libraries on both client and server codes.

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Linq;
using System.Net;
using System.Threading;
using System.Net.Sockets;
using System.Text;
using System.Windows.Forms;
```

First I created the required variables. We define 2 sockets, this one is a server and the other one is defined as a client in our client code. It opens 1 socket on localhost, server Client connects to it. The problem is we try to communicate them as Asynchronous by using the AsyncCallBack function.

```
namespace server
{
    4 başvuru
    public partial class Form1 : Form
    {
        public  Socket server;
        private byte[] data = new byte[10];
        private static int squareRoot = 15;// flag'ların etrafında 15 x 15 'lik bir alan belirliyor
        private int areaSize = squareRoot * 2;

        private int fCounter = 0;
        private Boolean attackTurn = false;
        private string enemyAttackResult;
        private int destroyedFlags = 0;
        private int maxAreaCount = 5; //başlangıçta maksimum seçilebilecek alan sayısı
        private Boolean areaChooseMode = true;
        private List<int> coordinates = new List<int>();
        private List<List<string>> AREAS = new List<List<string>>();
```

SOURCES

I took advantage of Microsoft's C# related Form posts and StackOverflow for errors and problems. I watched the lecture recordings.

Name Surname : Arda Dumanoğlu Sign: 

Number : 190315072

Also, I use some parts from our chapter 8 and 9 codes. Especially these parts in below are very useful and required for communication so I used it. They're coming from Chapter 9 Codes in TcpChat.cs file.

```
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 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);
}
```

SOLUTION DEVELOPMENT PROCESS

1-) First we took the pixels of the map as coordinates, in the flag sewing code, we scanned the 15x15 area around the flag with nested for loops and put it on the list, then we put those flag lists into the player list.

2-) We sent coordinates for the attack to the other side / received the coordinates of the attack from the opposite side. We checked with a for loop if the received coordinate is in any of our flag lists, if it does, we send the center coordinate of the flag it passed back to the opponent, if not, we send a 'miss' response.

3-) Then We evaluated the response, if not, we increased the flag counter we hit by one. In addition, in the evaluation part of the response, we added the text "Lose" next to the flag that was shot if we were hit, to show that there is a lost area.

Name Surname : Arda Dumanoğlu Sign:
Number : 190315072

4-) Lastly, If either of the two players conquered 5 lands if the received land counter was 5, we sent the winning message to the other side by turning off our coordinate sending mechanism.

CODE AND PROBLEM EXPLANATION

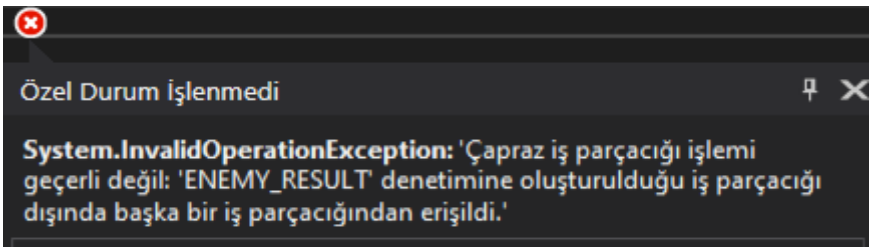
```
1 başvuru
public Form1()
{
    InitializeComponent();
    this.Text = "Server - Player 2";
    Control.CheckForIllegalCrossThreadCalls = false;

    attackButton.Click += new EventHandler(attackButton_click);
    attackCoords.Click += new EventHandler(coor_In);

    attackButton.Enabled = false;
    attackCoords.Enabled = false;

}

2 başvuru
```

The first code above it is like the main method in Java. It creates a listener for the Attack button and for the box where you write the attack coordinates. They're disabled until we select 5 coordinates on the map. Also after the game ends again it's disabled again.

```
if (receivedInfo == "win")
{
    attackTurn = false;

    attackButton.Enabled = false;

    attackCoords.Enabled = false;

    PLAYER_RESULT.Text = " Player 2 Lose !";

    ENEMY_RESULT.Text = "Player 1 Win !";
}

else if (attackTurn)
```

Name Surname : Arda Dumanoğlu Sign:
Number : 190315072

I got this error which is called ‘Cross-thread exception’.



So after I researched I found a solution in StackOverflow which is the solution to setting this highlighted boolean values as false.

```
public Form1()
{
    InitializeComponent();
    this.Text = "Server - Player 2";
    Control.CheckForIllegalCrossThreadCalls = false;

    attackButton.Click += new EventHandler(attackButton_click);
    attackCoords.Click += new EventHandler(coor_In);

    attackButton.Enabled = false;
    attackCoords.Enabled = false;

}
```

NOTE: All the remaining parts where is not mentioned here are

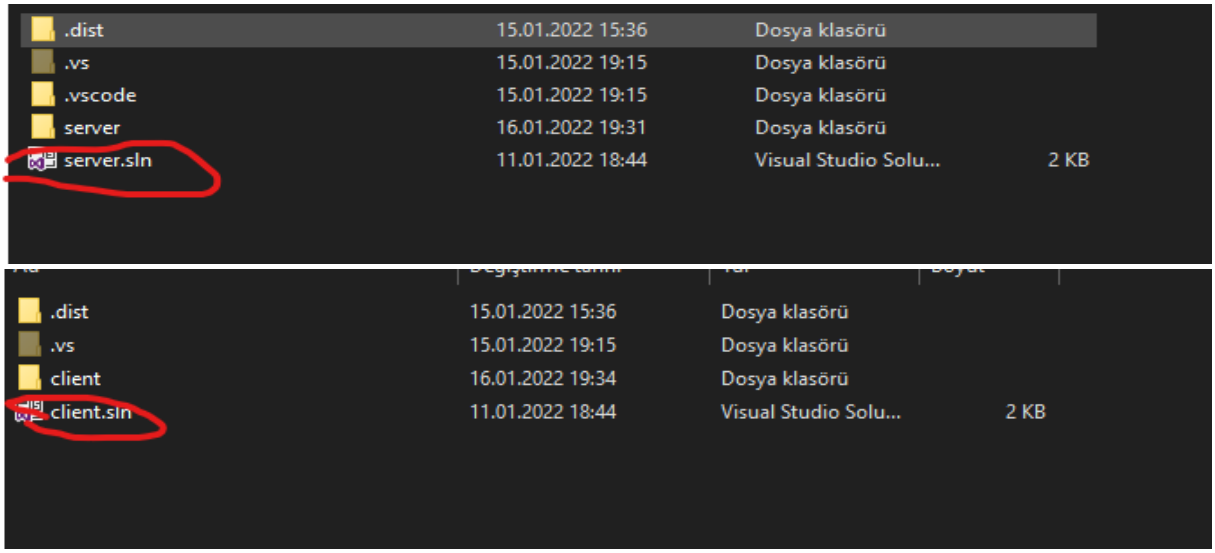
Written by using variables that can be understandably typed in the codes. That's why I didn't mention it here. Also, some codes are taken and edited from chapters 8 and 9 as I mentioned earlier.

INSTRUCTIONS FOR OPENING AND EXPLANATION OF RUNNING THE CODE(OUTPUT)

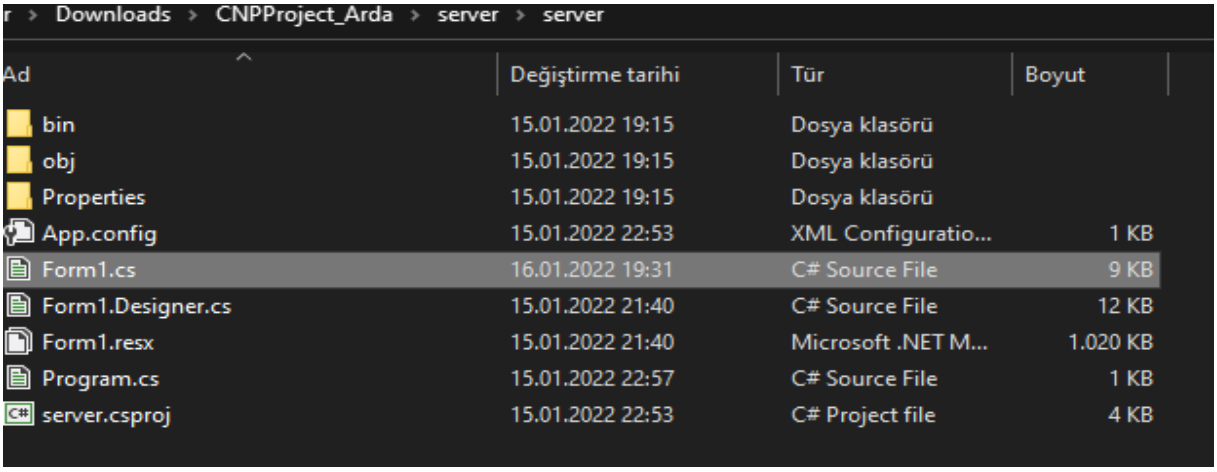
First, we must open the client and server folders and open both the **client.sln** and the **server.sln** files in them with visual studio.

Name Surname : Arda Dumanoğlu Sign:

Number : 190315072



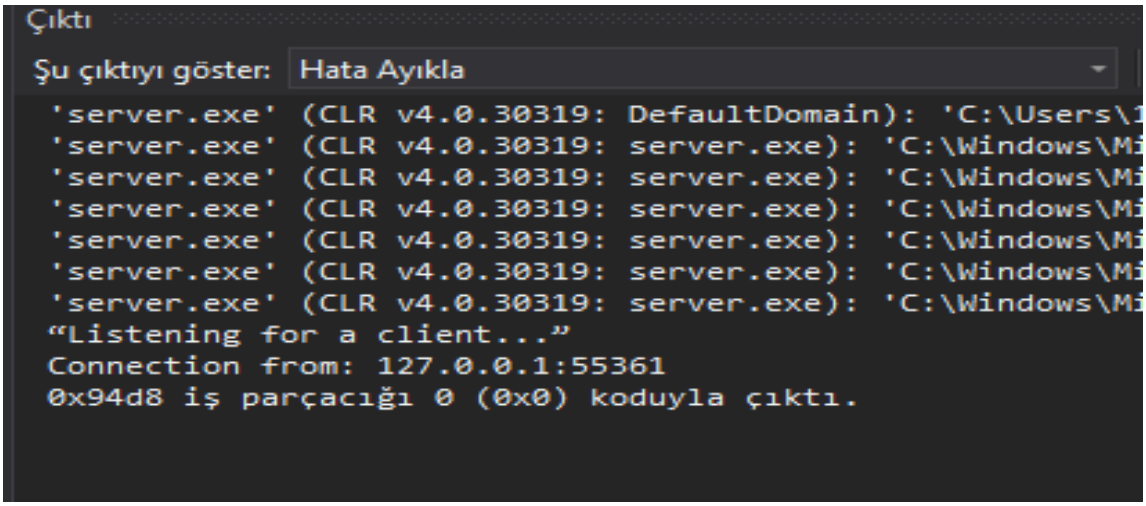
if our **form1.cs** and the other codes for both client and server did not appear on the screen. We can **drag** the **form1.cs** and the other codes from the client and server folders inside of the client and server folders into their visual studio. Now we must have 2 separate visual studios on the screen. We are ready to start.



First, we must run the **Server** code then the **client** code. When we run the server code, in the console it says “Listening for a client…”. It waits for the client’s response. Then when we run our client code it

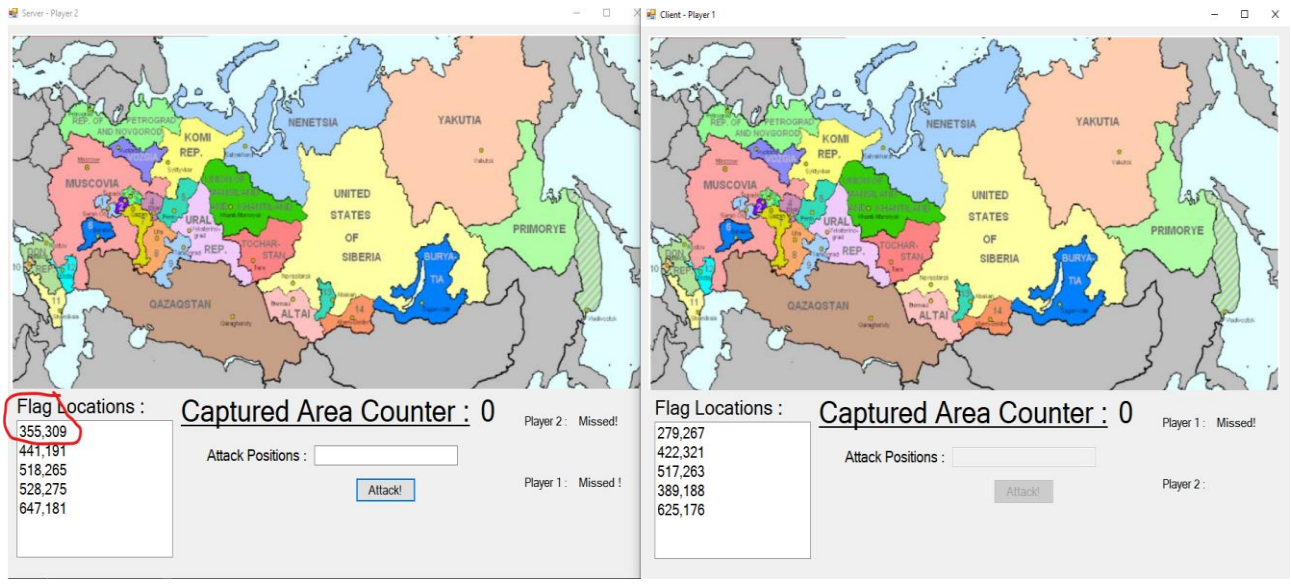
Name Surname : Arda Dumanoğlu Sign:
Number : 190315072

It connects to the server in localhost which is 127.0.0.1 and the port in this case it is 55361.



After selecting 5 flags for both client and server from the map, it keeps all possibilities in the array, up to 15 less and more of them for client and server.

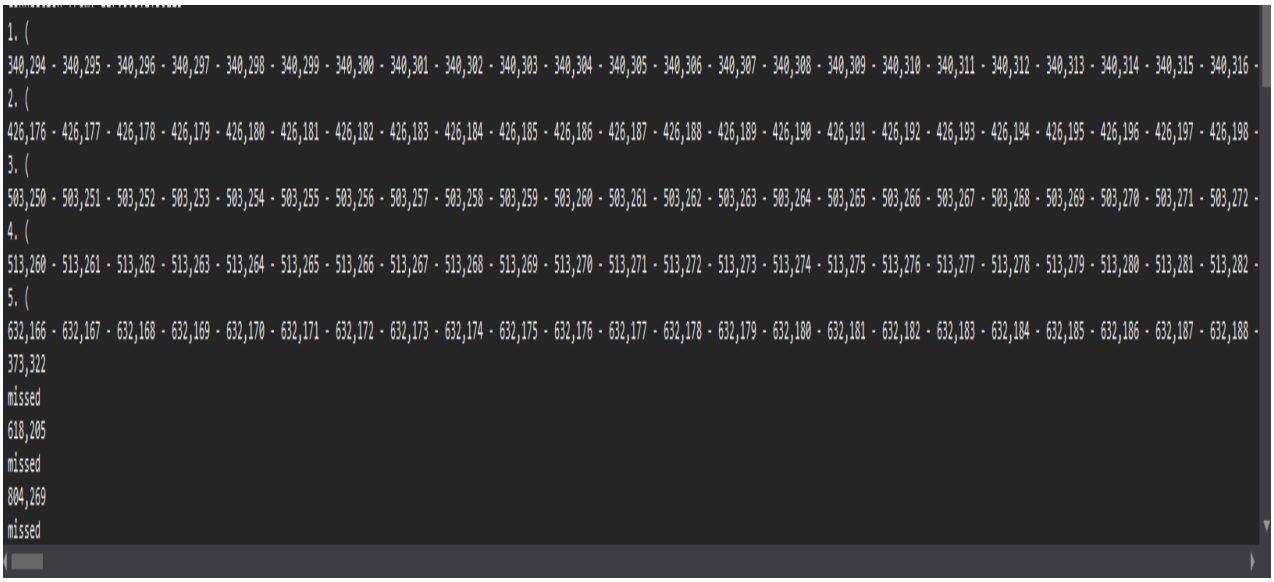
So for example, if we select 355,309 for Server's first flag.



Name Surname : Arda Dumanoğlu Sign:

Number : 190315072

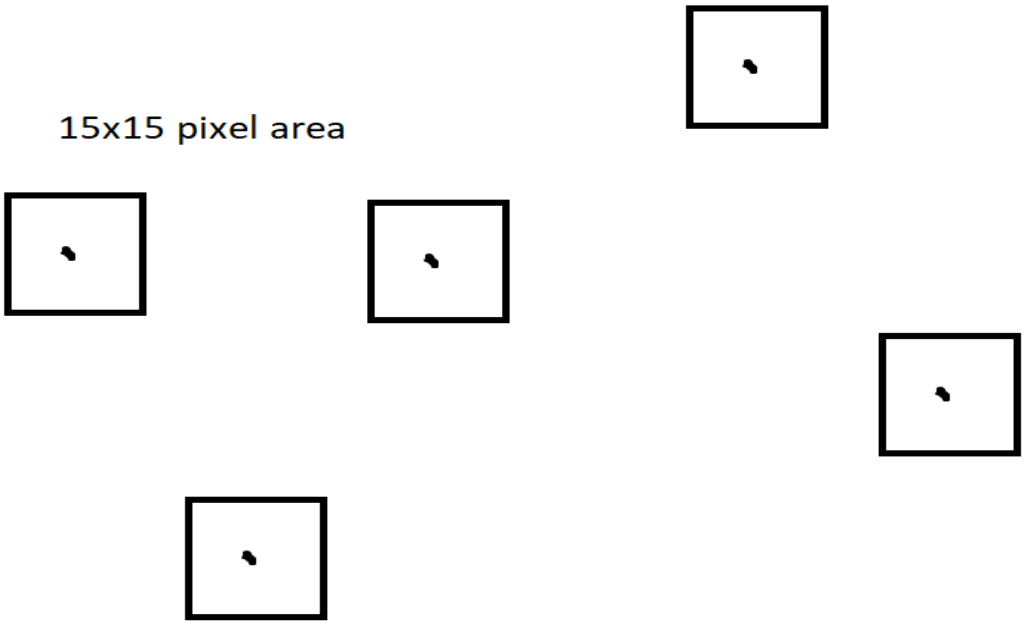
In the server's console, it appears like this.



It includes all possibilities from -15 to +15 for each flag since we create a 15x15 area in a total of 225 possibilities. In our example which is 355,309 it starts from -15 which is 340,294 up to the +15 which is 369,323. When we select any dot from the map it scans the 15-pixels for both vertical and horizontal areas around it.

Name Surname : Arda Dumanoğlu Sign:

Number : 190315072



Like this example above. It scans around all dots inside of these squares. So instead of clicking dot if the player clicks inside of square except for dot it still hit flags. This is the code for responsible printing console these selected flag coordinates from -15 to +15.

Name Surname : Arda Dumanoğlu Sign:

Number : 190315072

```
    AREAS.Add(flag);
    this.coordinates.Clear();
    if (fCounter == maxAreaCount)
    {
        areaChooseMode = false;

        attackButton.Enabled = attackTurn;

        attackCoords.Enabled = attackTurn;

        for (int i = 0; i < AREAS.Count; i++)
        {
            Console.WriteLine((i+1)+". (");
            for (int j = 0; j < areaSize*areaSize; j++) {
                Console.Write(AREAS.ElementAt(i).ElementAt(j) + " - ");
            }
            Console.WriteLine(")");
        }
    }
}
else { attackCoords.Text = MousePos; }
```

After the game ends. All buttons are set disabled as I mentioned before.