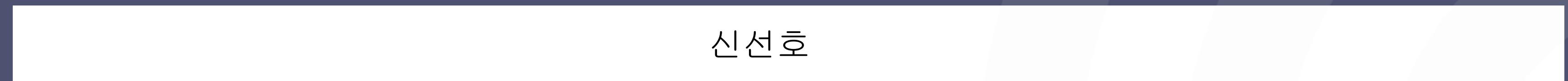




TCP/IP Socket 기반 서버-클라이언트 통신 프로그램

신선호



목차

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

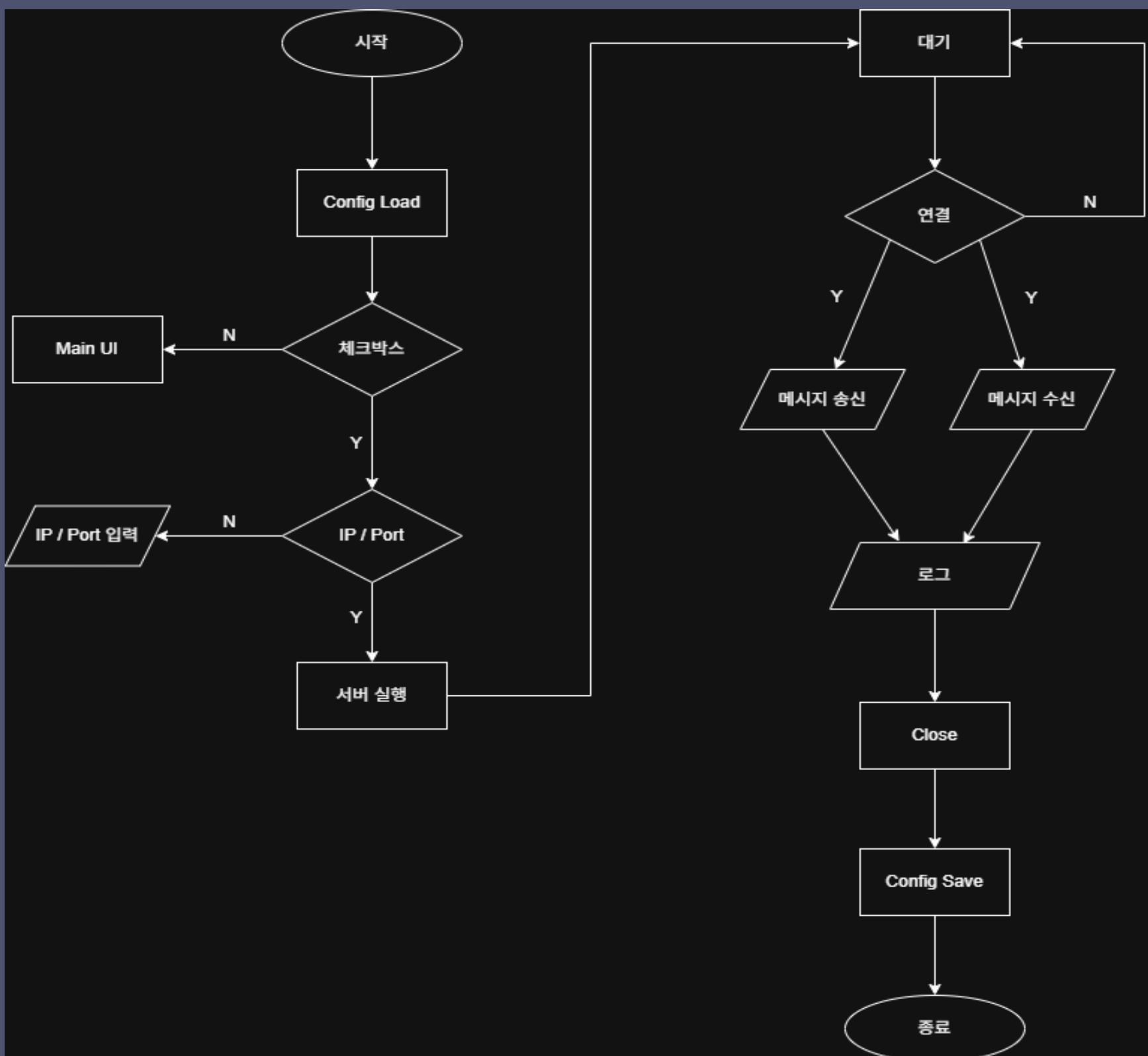
- 본 프로젝트는 C# TCP/IP Socket통신의 기본 구조를 정리하기 위해, 재사용 가능한 서버/클라이언트 베이스 프로젝트입니다.
- TCP/IP 기반 서버-클라이언트 통신 구조를 구현하였고, UI와 네트워크 로직을 분리한 데스크톱 애플리케이션을 구현하였습니다.
- 추후 다중 클라이언트 관리, 프로토콜 정의, 비동기 통신으로 확장이 가능합니다.

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과



1. 프로젝트 개요

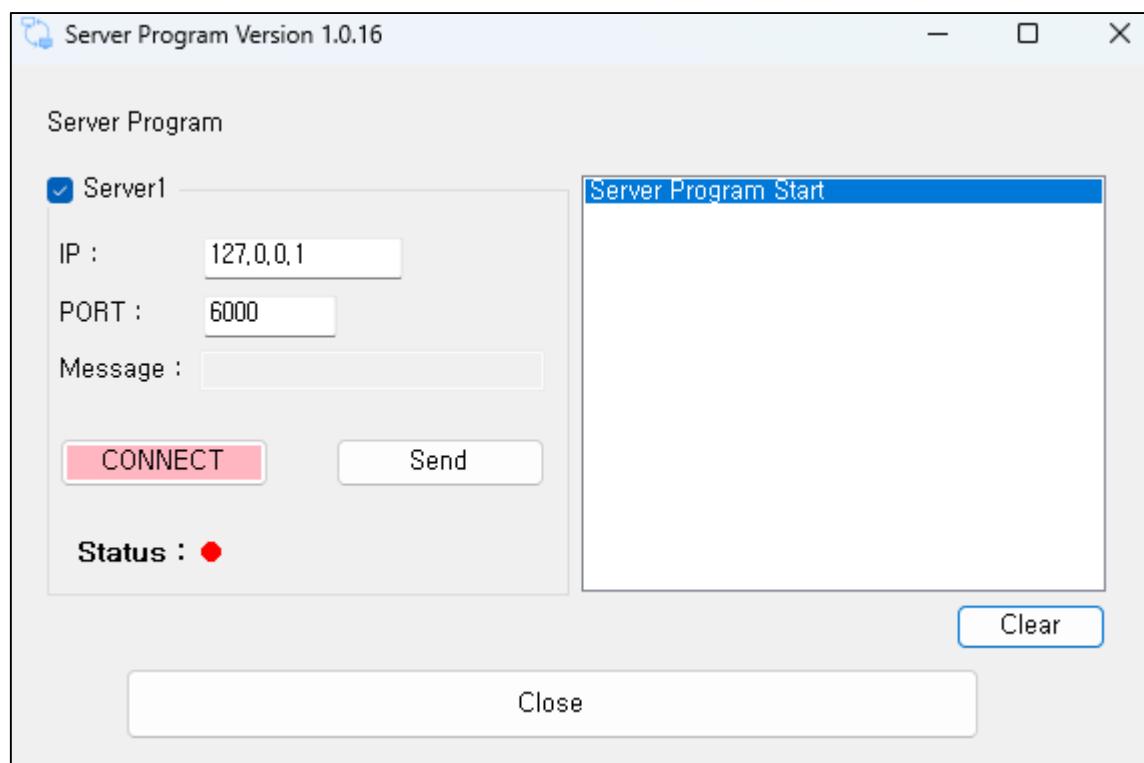
2. Flowchart

3. UI 및
주요코드

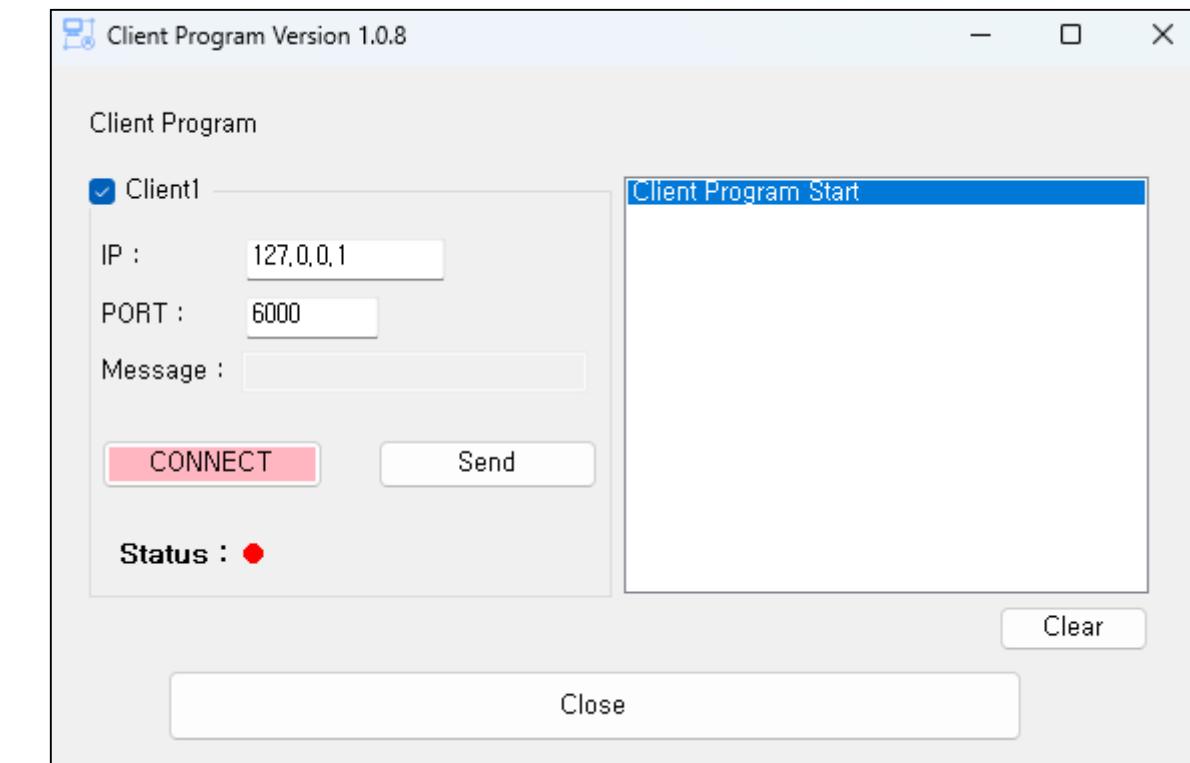
4. 실행결과

3-1. UI

Server



Client



1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

3-2. 주요코드 [ServerDrive]

```
참조 3개
public class ServerDrive
{
    private Socket listener;
    private Thread listenThread;

    private readonly List<Socket> clients = new List<Socket>();
    private readonly object lockObj = new object();

    public event Action<string> OnLog;
    public event Action<int> OnClientCountChanged;

    참조 13개
    public bool IsRunning { get; private set; }

    참조 7개
    public int ConnectedClientCount
    {
        get { lock (lockObj) return clients.Count; }
    }

    참조 1개
    public void Connect(string ip, int port)
    {
        if (IsRunning)
        {
            s_Log("이미 서버가 실행 중입니다.");
            return;
        }

        try
        {
            listener = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            listener.Bind(new IPEndPoint(IPAddress.Parse(ip), port));
            listener.Listen(10);

            IsRunning = true;

            listenThread = new Thread(ListenLoop);
            listenThread.IsBackground = true;
            listenThread.Start();

            s_Log($"서버 시작: {ip}: {port}");
        }
        catch (Exception ex)
        {
            s_Log($"서버 시작 실패: {ex.Message}");
        }
    }
}
```

```
참조 2개
public void Disconnect()
{
    if (!IsRunning) return;

    IsRunning = false;

    try { listener.Close(); } catch { }

    lock (lockObj)
    {
        foreach (var c in clients) try { c.Close(); } catch { }
        clients.Clear();
    }

    s_Log("서버 종료");
}

참조 1개
private void ListenLoop()
{
    while (IsRunning)
    {
        try
        {
            Socket client = listener.Accept();

            lock (lockObj)
            {
                clients.Add(client);
            }

            OnClientCountChanged?.Invoke(ConnectedClientCount);

            s_Log($"클라이언트 접속: {client.RemoteEndPoint}");

            Thread recvThread = new Thread(() => ReceiveLoop(client));
            recvThread.IsBackground = true;
            recvThread.Start();
        }
        catch
        {
            if (!IsRunning)
            {
                return;
            }
        }
    }
}
```

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

3-2. 주요코드 [ServerDrive]

```
참조 1개
private void ReceiveLoop(Socket client)
{
    try
    {
        byte[] buffer = new byte[4096];

        while (IsRunning && client.Connected)
        {
            int n = client.Receive(buffer);

            if (n <= 0)
            {
                break;
            }

            string msg = Encoding.UTF8.GetString(buffer, 0, n);

            s_Log($"[수신]: {msg}");
        }
    catch ( )
    finally
    {
        lock (lockObj)
        {
            clients.Remove(client);
        }

        if (IsRunning)
        {
            OnClientCountChanged?.Invoke(ConnectedClientCount);
            s_Log("클라이언트 연결 종료");
        }
    }
    client.Close();
}
```

```
참조 1개
public void SendMessage(string msg)
{
    if (!IsRunning)
    {
        return;
    }

    if (ConnectedClientCount == 0)
    {
        return;
    }

    byte[] data = Encoding.UTF8.GetBytes(msg);

    lock (lockObj)
    {
        foreach (var cli in clients.ToArray())
        {
            try
            {
                cli.Send(data);
            }
            catch
            {
                clients.Remove(cli);
            }
        }

        s_Log($"[송신]: {msg}");
    }
}

참조 8개
private void s_Log(string msg)
{
    OnLog?.Invoke(msg);
}
```

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

3-2. 주요코드 [ClientDrive]

```
// 클라이언트 드라이브 클래스  
  
참조 22개  
public enum ClientState // 내부 상태 점검용  
{  
    Ready,  
    Connecting,  
    Connected,  
    Disconnecting,  
    Disconnected,  
    Finished  
}  
  
참조 14개  
public enum ConnectionStatus // UI 변경용  
{  
    Connecting,  
    Connected,  
    Disconnecting,  
    Disconnected,  
    Error  
}
```

```
참조 2개  
public class ClientDrive  
{  
    private Socket socket;  
    private Thread workerThread;  
    private Thread receiveThread;  
    private volatile ClientState state = ClientState.Ready;  
  
    private readonly object sendLock = new object();  
  
    public event Action<ConnectionStatus> OnStatusChanged;  
    public event Action<string> OnMessageReceived;  
    public event Action<string> OnLog;  
  
    참조 4개  
    public string IP { get; private set; }  
    참조 4개  
    public int Port { get; private set; }  
  
    참조 10개  
    public bool IsConnected => state == ClientState.Connected;  
  
    참조 1개  
    public void Connect(string host, int port)  
    {  
        if (state == ClientState.Connected || state == ClientState.Connecting) { return; }  
  
        IP = host;  
        Port = port;  
  
        state = ClientState.Connecting;  
  
        if (workerThread == null || !workerThread.IsAlive)  
        {  
            workerThread = new Thread(RunWorker);  
            workerThread.IsBackground = true;  
            workerThread.Start();  
        }  
    }
```

```
참조 1개  
private void RunWorker()  
{  
    while (state != ClientState.Finished)  
    {  
        Thread.Sleep(10);  
  
        switch (state)  
        {  
            case ClientState.Connecting:  
                TryConnect();  
                break;  
  
            case ClientState.Connected:  
                if (socket == null || !socket.Connected)  
                {  
                    state = ClientState.Disconnecting;  
                }  
                break;  
  
            case ClientState.Disconnecting:  
                TryDisconnect();  
                break;  
  
            case ClientState.Disconnected:  
                state = ClientState.Finished;  
                break;  
  
            case ClientState.Finished:  
                return;  
        }  
    }  
}
```

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

3-2. 주요코드 [ClientDrive]

```
private void TryConnect()
{
    try
    {
        OnStatusChanged?.Invoke(ConnectionStatus.Connecting);
        c_Log($"서버({IP}:{Port}) 연결 시도 중...");

        if (socket != null)
        {
            try { socket.Close(); } catch { }

            socket = new Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);
            var ep = new IPEndPoint(IPAddress.Parse(IP), Port);

            var result = socket.BeginConnect(ep, null, null);
            bool success = result.AsyncWaitHandle.WaitOne(2000, false);

            if (!success || !socket.Connected) { throw new Exception("연결 시간 초과 또는 서버 응답 없음"); }

            socket.EndConnect(result);
            state = ClientState.Connected;

            c_Log($"서버({IP}:{Port}) 연결됨");
            OnStatusChanged?.Invoke(ConnectionStatus.Connected);

            receiveThread = new Thread(ReceiveLoop);
            receiveThread.IsBackground = true;
            receiveThread.Start();
        }
        catch (Exception ex)
        {
            c_Log($"연결 실패: {ex.Message}");
            OnStatusChanged?.Invoke(ConnectionStatus.Error);

            state = ClientState.Disconnected;
            state = ClientState.Ready;
        }
    }
}
```

```
참조 3개
public void Disconnect()
{
    if (state == ClientState.Connected || state == ClientState.Connecting)
    {
        state = ClientState.Disconnecting;
    }
}

참조 1개
private void TryDisconnect()
{
    try
    {
        OnStatusChanged?.Invoke(ConnectionStatus.Disconnecting);

        if (socket != null)
        {
            if (socket.Connected)
            {
                socket.Shutdown(SocketShutdown.Both);
            }

            socket.Close();
            socket = null;
        }
    }
    catch { }
    finally
    {
        state = ClientState.Disconnected;

        c_Log("서버 연결 종료");
        OnStatusChanged?.Invoke(ConnectionStatus.Disconnected);

        state = ClientState.Ready;
    }
}
```

1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

3-2. 주요코드 [ClientDrive]

```
참조 2개
public void SendMessage(string msg)
{
    if (!IsConnected)
    {
        c_Log("서버에 연결되어 있지 않습니다.");
        return;
    }

    lock (sendLock)
    {
        try
        {
            byte[] data = Encoding.UTF8.GetBytes(msg);
            socket.Send(data);
        }
        catch (Exception ex)
        {
            c_Log($"송신 오류: {ex.Message}");
        }
    }
}
```

```
참조 1개
private void ReceiveLoop()
{
    try
    {
        byte[] buffer = new byte[4096];

        while (IsConnected)
        {
            int n = socket.Receive(buffer);

            if (n <= 0)
            {
                break;
            }

            string msg = Encoding.UTF8.GetString(buffer, 0, n);

            OnMessageReceived?.Invoke(msg);
        }
    }
    catch (Exception)
    {
        if (IsConnected)
        {
            c_Log("서버 연결 끊김");
        }
    }
    finally
    {
        if (IsConnected)
        {
            Disconnect();
        }
    }
}
```

```
참조 8개
private void c_Log(string msg)
{
    OnLog?.Invoke(msg);
}
```

1. 프로젝트 개요

2. Flowchart

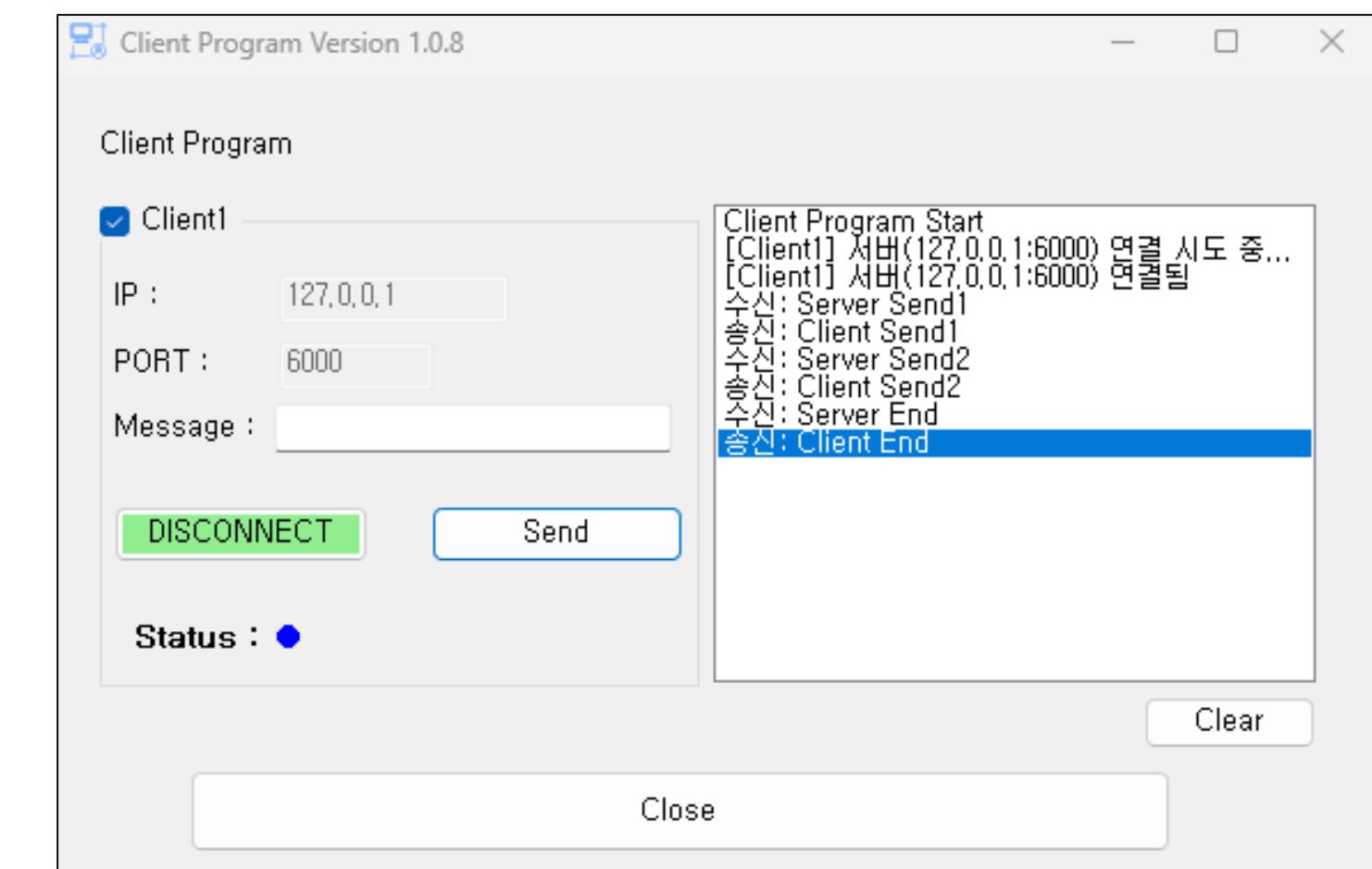
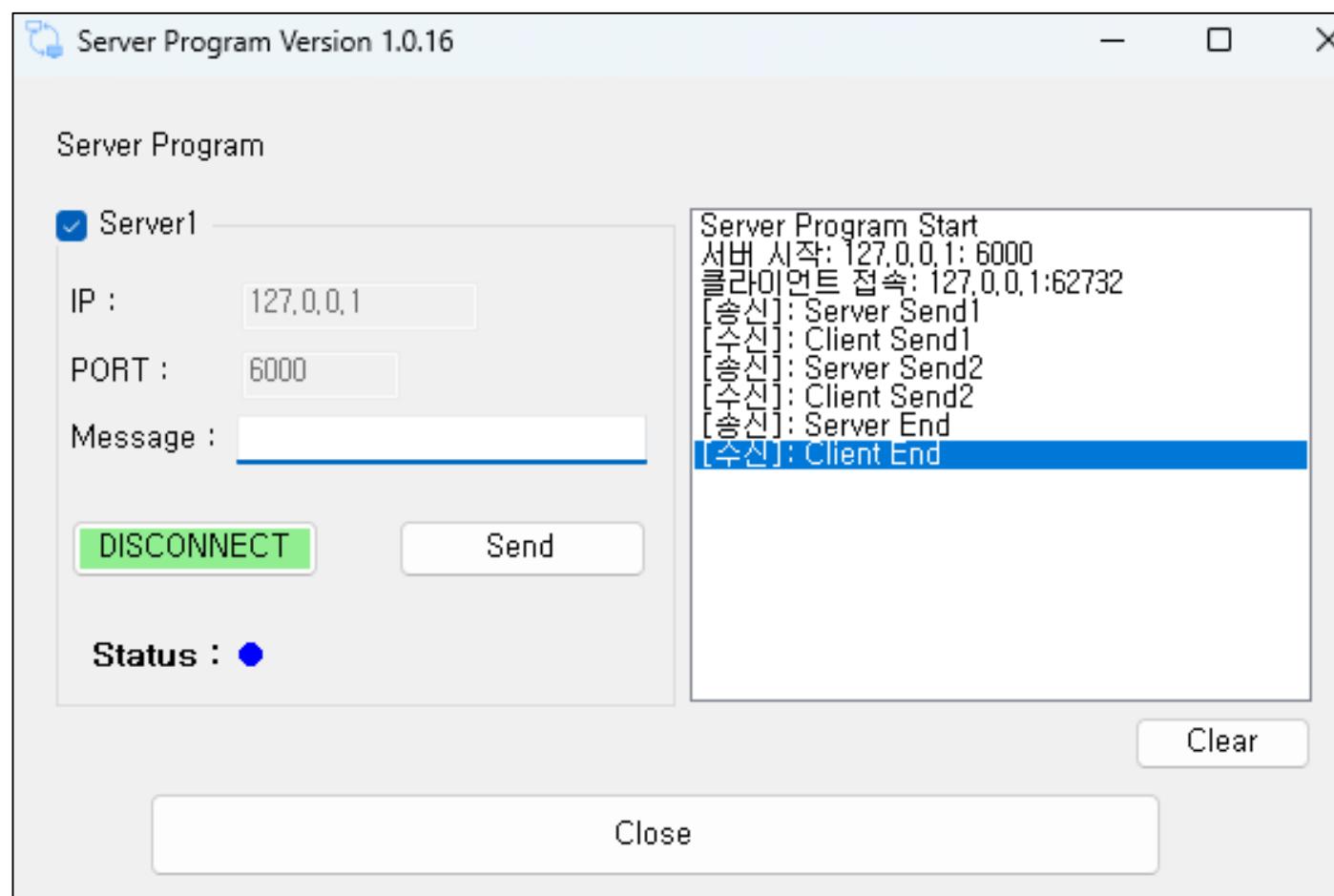
3. UI 및
주요코드

4. 실행결과

[Form]

Server

Client



1. 프로젝트 개요

2. Flowchart

3. UI 및 주요코드

4. 실행결과

[Config.INI]

[Log]

Server

```
[-] [CheckBox1]
  Enable=1
[-] [Server1]
  IP1=127.0.0.1
  PORT1=6000
```

Client

```
[-] [CheckBox1]
  Enable=1
[-] [Client1]
  IP1=127.0.0.1
  PORT1=6000
```

Server

```
2026-02-03 17:13:04.159  Server Program Start
2026-02-03 17:13:12.880  서버 시작: 127.0.0.1: 6000
2026-02-03 17:13:14.127  클라이언트 접속: 127.0.0.1:62732
2026-02-03 17:13:27.310  [송신]: Server Send1
2026-02-03 17:13:36.138  [수신]: Client Send1
2026-02-03 17:13:42.762  [송신]: Server Send2
2026-02-03 17:13:48.263  [수신]: Client Send2
2026-02-03 17:13:54.767  [송신]: Server End
2026-02-03 17:14:06.612  [수신]: Client End
2026-02-03 17:14:51.099  서버 종료
```

Client

```
2026-02-03 17:13:08.511  Client Program Start
2026-02-03 17:13:14.112  [Client1] 서버(127.0.0.1:6000) 연결 시도 중...
2026-02-03 17:13:14.121  [Client1] 서버(127.0.0.1:6000) 연결됨
2026-02-03 17:13:27.310  수신: Server Send1
2026-02-03 17:13:36.135  송신: Client Send1
2026-02-03 17:13:42.762  수신: Server Send2
2026-02-03 17:13:48.261  송신: Client Send2
2026-02-03 17:13:54.769  수신: Server End
2026-02-03 17:14:06.607  송신: Client End
2026-02-03 17:14:51.099  [Client1] 서버 연결 끊김
2026-02-03 17:14:51.127  [Client1] 서버 연결 종료
```

Thank You

감사합니다.