

# 컴퓨터공학 All in One

---

C/C++ 문법, 자료구조 및 심화 프로젝트 (나동빈)  
제 79강 - 오목 서버 프로그램 소스코드 리팩토링

# 오목 서버 프로그램 소스코드 리팩토링

## 문제점 분석 ① 프로젝트 구성

- 현재 서버 프로그램의 모든 모듈이 하나의 소스코드(Server.cpp)에 작성되어 있습니다.
- 확장성 및 재사용성의 증대를 위하여 프로그램을 모듈화 할 필요가 있습니다.

# 오목 서버 프로그램 소스코드 리팩토링

## 문제점 분석 ① 프로젝트 구성

다음과 같이 각 기능을 모듈별로 분해하여 프로그램을 작성합니다.



# 오목 서버 프로그램 소스코드 리팩토링

## 문제점 분석 ② 불필요한 서버 자원 낭비

- 현재 접속한 플레이어(Client)를 저장하는 connections 리스트에서 포인터 원소를 사용하지 않고 있습니다.
- 한 명의 플레이어(Client) 정보는 한 번 선언되면 반복적으로 선언하지 않도록 해야 합니다.

# 오목 서버 프로그램 소스코드 리팩토링

## Client.h

```
#ifndef GOMOKU_CLIENT_H
#define GOMOKU_CLIENT_H
#include <Winsock.h>
class Client {
private:
    int clientID;
    int roomID;
    SOCKET clientSocket;
public:
    Client(int clientID, SOCKET clientSocket);
    int getClientID();
    int getRoomID();
    void setRoomID(int roomID);
    SOCKET getClientSocket();
};
#endif
```

# 오목 서버 프로그램 소스코드 리팩토링

## Client.cpp

```
#include "Client.h"

Client::Client(int clientID, SOCKET clientSocket) {
    this->clientID = clientID;
    this->roomId = -1;
    this->clientSocket = clientSocket;
}

int Client::getClientID() {
    return clientID;
}

int Client::getRoomID() {
    return roomId;
}

void Client::setRoomID(int roomId) {
    this->roomId = roomId;
}

SOCKET Client::getClientSocket() {
    return clientSocket;
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Util.h

```
#ifndef GOMOKU_UTIL_H
#define GOMOKU_UTIL_H
using namespace std;
#include <vector>
#include <sstream>
class Util {
public:
    vector<string> getTokens(string input, char delimiter);
};
#endif
```

# 오목 서버 프로그램 소스코드 리팩토링

## Util.cpp

```
#include "Util.h"

vector<string> Util::getTokens(string input, char delimiter) {
    vector<string> tokens;
    istringstream f(input);
    string s;
    while (getline(f, s, delimiter)) {
        tokens.push_back(s);
    }
    return tokens;
}
```



# 오목 서버 프로그램 소스코드 리팩토링

## Server.h ①

```
#ifndef GOMOKU_SERVER_H
#define GOMOKU_SERVER_H
#define _CRT_SECURE_NO_WARNINGS
#pragma comment(lib, "ws2_32.lib")
#include <iostream>
using namespace std;
#include <Winsock.h>
#include <vector>
#include "Util.h"
#include "Client.h"
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.h ②

```
static class Server {  
private:  
    static SOCKET serverSocket;  
    static WSADATA wsaData;  
    static SOCKADDR_IN serverAddress;  
    static int nextID;  
    static vector<Client*> connections;  
    static Util util;  
public:  
    static void start();  
    static int clientCountInRoom(int roomID);  
    static void playClient(int roomID);  
    static void exitClient(int roomID);  
    static void putClient(int roomID, int x, int y);  
    static void fullClient(Client *client);  
    static void enterClient(Client *client);  
    static void ServerThread(Client *client);  
};  
#endif
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ①

```
#include "Server.h"

SOCKET Server::serverSocket;
WSADATA Server::wsaData;
SOCKADDR_IN Server::serverAddress;
int Server::nextID;
vector<Client*> Server::connections;
Util Server::util;
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ②

```
void Server::enterClient(Client *client) {  
    char *sent = new char[256];  
    ZeroMemory(sent, 256);  
    sprintf(sent, "%s", "[Enter]");  
    send(client->getClientSocket(), sent, 256, 0);  
}  
  
void Server::fullClient(Client *client) {  
    char *sent = new char[256];  
    ZeroMemory(sent, 256);  
    sprintf(sent, "%s", "[Full]");  
    send(client->getClientSocket(), sent, 256, 0);  
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ③

```
void Server::playClient(int roomID) {  
    char *sent = new char[256];  
    bool black = true;  
    for (int i = 0; i < connections.size(); i++) {  
        if (connections[i]->getRoomID() == roomID) {  
            ZeroMemory(sent, 256);  
            if (black) {  
                sprintf(sent, "%s", "[Play]Black");  
                black = false;  
            }  
            else {  
                sprintf(sent, "%s", "[Play]White");  
            }  
            send(connections[i]->getClientSocket(), sent, 256, 0);  
        }  
    }  
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ④

```
void Server::exitClient(int roomId) {
    char *sent = new char[256];
    for (int i = 0; i < connections.size(); i++) {
        if (connections[i]->getRoomID() == roomId) {
            ZeroMemory(sent, 256);
            sprintf(sent, "%s", "[Exit]");
            send(connections[i]->getClientSocket(), sent, 256, 0);
        }
    }
}

void Server::putClient(int roomId, int x, int y) {
    char *sent = new char[256];
    for (int i = 0; i < connections.size(); i++) {
        if (connections[i]->getRoomID() == roomId) {
            ZeroMemory(sent, 256);
            string data = "[Put]" + to_string(x) + "," + to_string(y);
            sprintf(sent, "%s", data.c_str());
            send(connections[i]->getClientSocket(), sent, 256, 0);
        }
    }
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ⑤

```
int Server::clientCountInRoom(int roomID) {  
    int count = 0;  
    for (int i = 0; i < connections.size(); i++) {  
        if (connections[i]->getRoomID() == roomID) {  
            count++;  
        }  
    }  
    return count;  
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ⑥

```
void Server::start() {
    WSStartup(MAKEWORD(2, 2), &wsaData);
    serverSocket = socket(AF_INET, SOCK_STREAM, NULL);

    serverAddress.sin_addr.s_addr = inet_addr("127.0.0.1");
    serverAddress.sin_port = htons(9876);
    serverAddress.sin_family = AF_INET;

    cout << "[ C++ 오목 게임 서버 가동 ]" << endl;
    bind(serverSocket, (SOCKADDR*)&serverAddress, sizeof(serverAddress));
    listen(serverSocket, 32);

    int addressLength = sizeof(serverAddress);
    while (true) {
        SOCKET clientSocket = socket(AF_INET, SOCK_STREAM, NULL);
        if (clientSocket = accept(serverSocket, (SOCKADDR*)&serverAddress, &addressLength)) {
            Client *client = new Client(nextID, clientSocket);
            cout << "[ 새로운 사용자 접속 ]" << endl;
            CreateThread(NULL, NULL, (LPTHREAD_START_ROUTINE)ServerThread, (LPVOID)client, NULL, NULL);
            connections.push_back(client);
            nextID++;
        }
        Sleep(100);
    }
}
```



# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ⑦

```
void Server::ServerThread(Client *client) {
    char *sent = new char[256];
    char *received = new char[256];
    int size = 0;
    while (true) {
        ZeroMemory(received, 256);
        if ((size = recv(client->getClientSocket(), received, 256, NULL)) > 0) {
            string receivedString = string(received);
            vector<string> tokens = util.getTokens(receivedString, ']');
            if (receivedString.find("[Enter]") != -1) {
                string roomID = tokens[1];
                int roomInt = atoi(roomID.c_str());
                int clientCount = clientCountInRoom(roomInt);
                /* 2명 이상이 동일한 방에 들어가 있는 경우 */
                if (clientCount >= 2) {
                    fullClient(client);
                }
            }
        }
    }
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ⑧

```
/* 접속 성공 */
client->setRoomID(roomInt);
cout << "클라이언트 [" << client->getClientID() << "]: " << client->getRoomID() << "번 방으로 접속" << endl;
/* 방에 성공적으로 접속했다고 메시지 전송 */
enterClient(client);
/* 상대방이 이미 방에 들어가 있는 경우 게임 시작 */
if (clientCount == 1) {
    playClient(roomInt);
}
}
else if (receivedString.find("[Put]") != -1) {
    /* 메시지를 보낸 클라이언트 정보 받기 */
    string data = tokens[1];
    vector<string> dataTokens = util.getTokens(data, ',');
    int roomID = atoi(dataTokens[0].c_str());
    int x = atoi(dataTokens[1].c_str());
    int y = atoi(dataTokens[2].c_str());
    /* 사용자가 놓은 돌의 위치를 전송 */
    putClient(client->getRoomID(), x, y);
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Server.cpp ⑨

```
else if (receivedString.find("[Play]") != -1) {
    string roomID = tokens[1];
    int roomInt = atoi(roomID.c_str());
    /* 사용자가 접속한 방의 번호에 시작 알림 전송 */
    playClient(client->getRoomID());
}
}
else {
    cout << "클라이언트[" << client->getClientID() << "]의 연결이 끊어졌습니다." << endl;
    /* 게임에서 나간 플레이어를 찾기 */
    for (int i = 0; i < connections.size(); i++) {
        if (connections[i]->getClientID() == client->getClientID()) {
            /* 다른 사용자와 게임 중이던 사람이 나간 경우 */
            if (connections[i]->getRoomID() != -1 &&
                clientCountInRoom(connections[i]->getRoomID()) == 2) {
                /* 남아있는 사람에게 메시지 전송 */
                exitClient(connections[i]->getRoomID());
            }
            connections.erase(connections.begin() + i);
            break;
        }
    }
    delete client;
    break;
}
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## Main.cpp

```
#include "Server.h"

int main(void) {
    Server::start();
}
```

# 오목 서버 프로그램 소스코드 리팩토링

## 리팩토링 결과 반영하기

- 깃(Git)에 리팩토링 결과를 반영합니다. 아직 테스트가 덜 되었다고 느낄 때는 새로운 브랜치에 반영합니다.
- `git branch development`
- `git checkout development`
- `git branch`
- `git add .`
- `git commit -m "Source Code Refactoring"`
- `git push`

# 오목 서버 프로그램 소스코드 리팩토링

## 온라인 게임 개발을 위한 기본기

- 온라인 게임 수준의 오목 게임을 만들고 싶다면 플레이어 오목 경기 관련 DB 구축이 필요합니다.
- 경기의 승리 판정 로직을 클라이언트가 아닌 서버에게 위임할 필요가 있습니다.
- 성능 최적화를 위하여 Boost.Asio와 같은 네트워크 통신 모듈을 사용할 수 있습니다.
- 클라이언트 개발을 위해 C#이 아닌 웹을 이용할 수 있습니다.