

# Chatting Server

IOCP를 활용한 채팅 서버

# 목차

- 개요
- Design
  1. 요구사항
  2. 설계 세부 계획
- Class Diagram
- Sequence Diagram
- 파일 구조
- 실행화면
- 코드 샘플
- 블로그 & Github 주소

# 개요

- 구현 목표

1. 채팅방 별로 참가자들끼리 채팅이 가능한 채팅 서버 구현.
2. Database를 통해 유저의 로그인 정보를 저장하고 접속 유저를 인증
3. 멀티스레드를 활용하되 패킷 송수신 시에는 IOCP를 활용하고, 패킷 처리 시에는 단일 워커 스레드를 사용해 스레드 간의 동기화 비용 줄임.

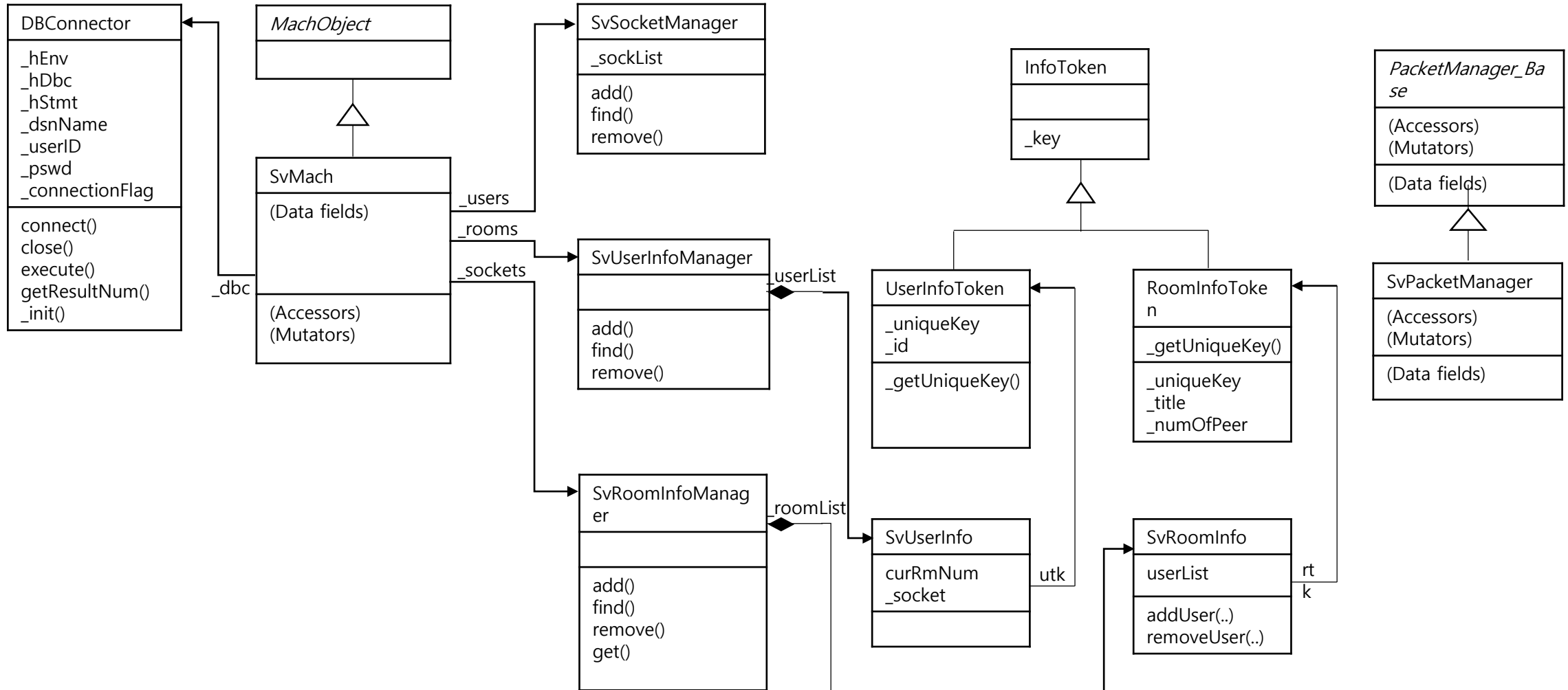
# 디자인 1. 설계 요구 사항

- 서버와 유저의 로그인 정보를 확인하는 Database를 연결
- 유저의 정보를 기록하고 관리
- 채팅방의 정보를 기록하고 관리
- 서버에서 관리하는 정보 중 전송용 정보는 처리 효율성과 보안성을 위해 다른 정보와 구분
- 별도의 패킷 매니저를 통해 패킷 송수신을 처리
- 서버에 연결된 클라이언트의 소켓을 관리

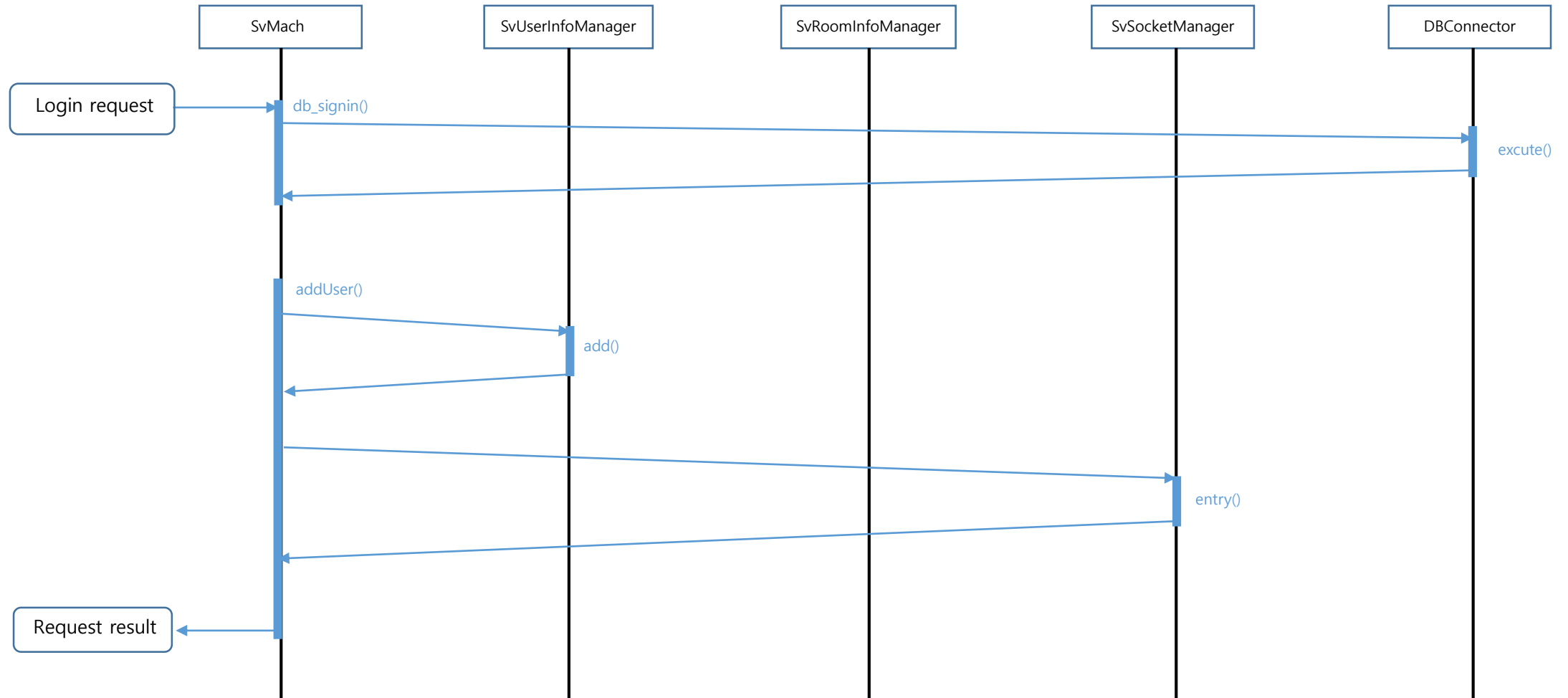
## 디자인 2. 설계 세부 사항

1. DBConnection : DB와 연결 및 쿼리 처리를 담당하는 클래스
2. UserInfoToken : 유저의 정보 중 클라이언트와 서버 간 전송되는 정보를 저장하는 클래스
3. SvUserInfo : 유저의 정보 중 전송될 필요가 없는 정보를 저장하는 클래스
4. SvUserManager : 유저 정보 관련 객체의 할당/해제와 유저 정보의 데이터구조를 관리하는 클래스
5. RoomInfoToken : 채팅방의 정보 중 클라이언트와 서버 간 전송되는 정보를 저장하는 클래스
6. SvRoomInfo : 채팅방의 정보 중 전송될 필요가 없는 정보를 저장하는 클래스
7. SvRoomInfoManager : 채팅방 정보 관련 객체의 할당/해제와 유저 정보의 데이터구조를 관리하는 클래스
8. SvSocketManager : 소켓과 유저 정보를 매핑하여 저장하고, 그 데이터구조를 관리하는 클래스
9. SvPacketManager : 서버의 송수신 패킷을 관리하는 클래스

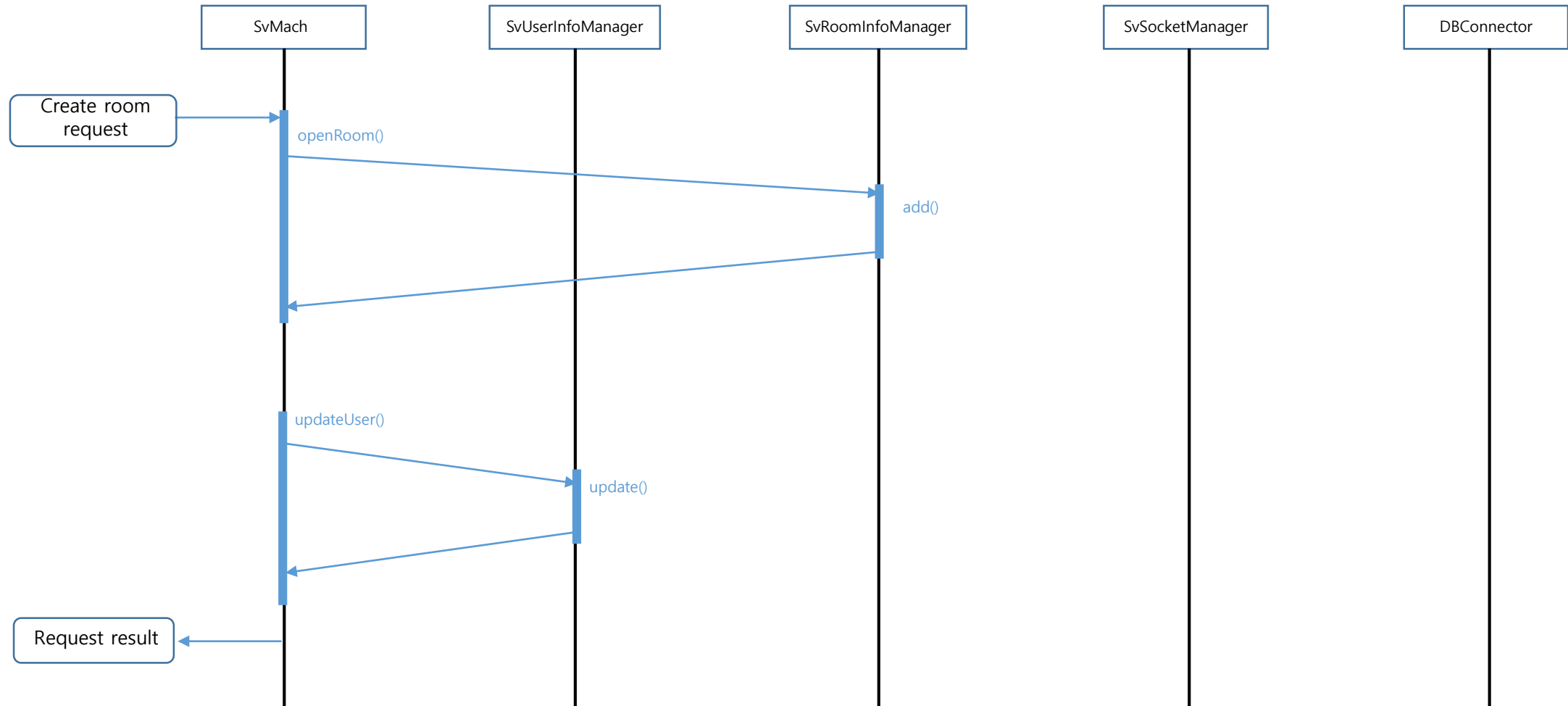
# Class Diagram



# Sequence Diagram : 로그인 시

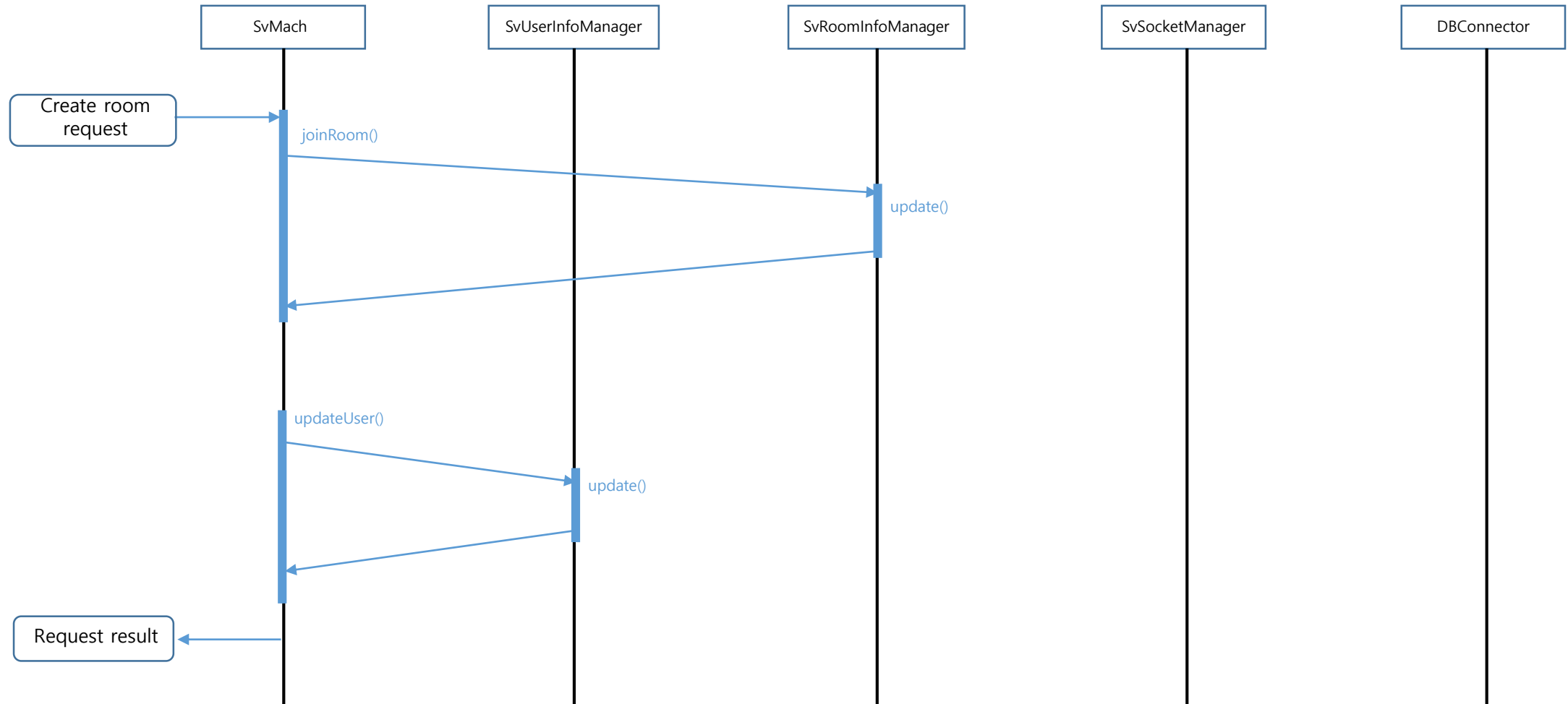


# Sequence Diagram : 새로운 채팅방 생성시



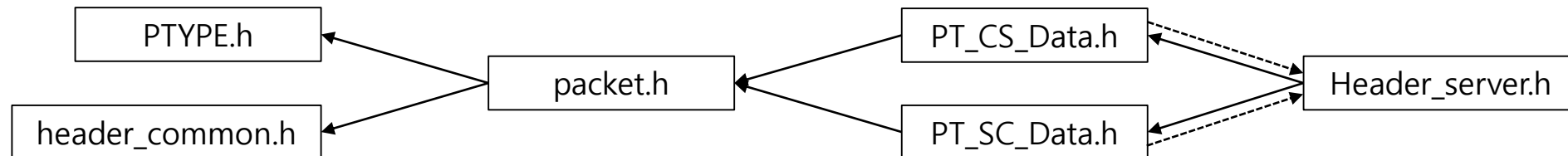


# Sequence Diagram : 기존 채팅방 참가시

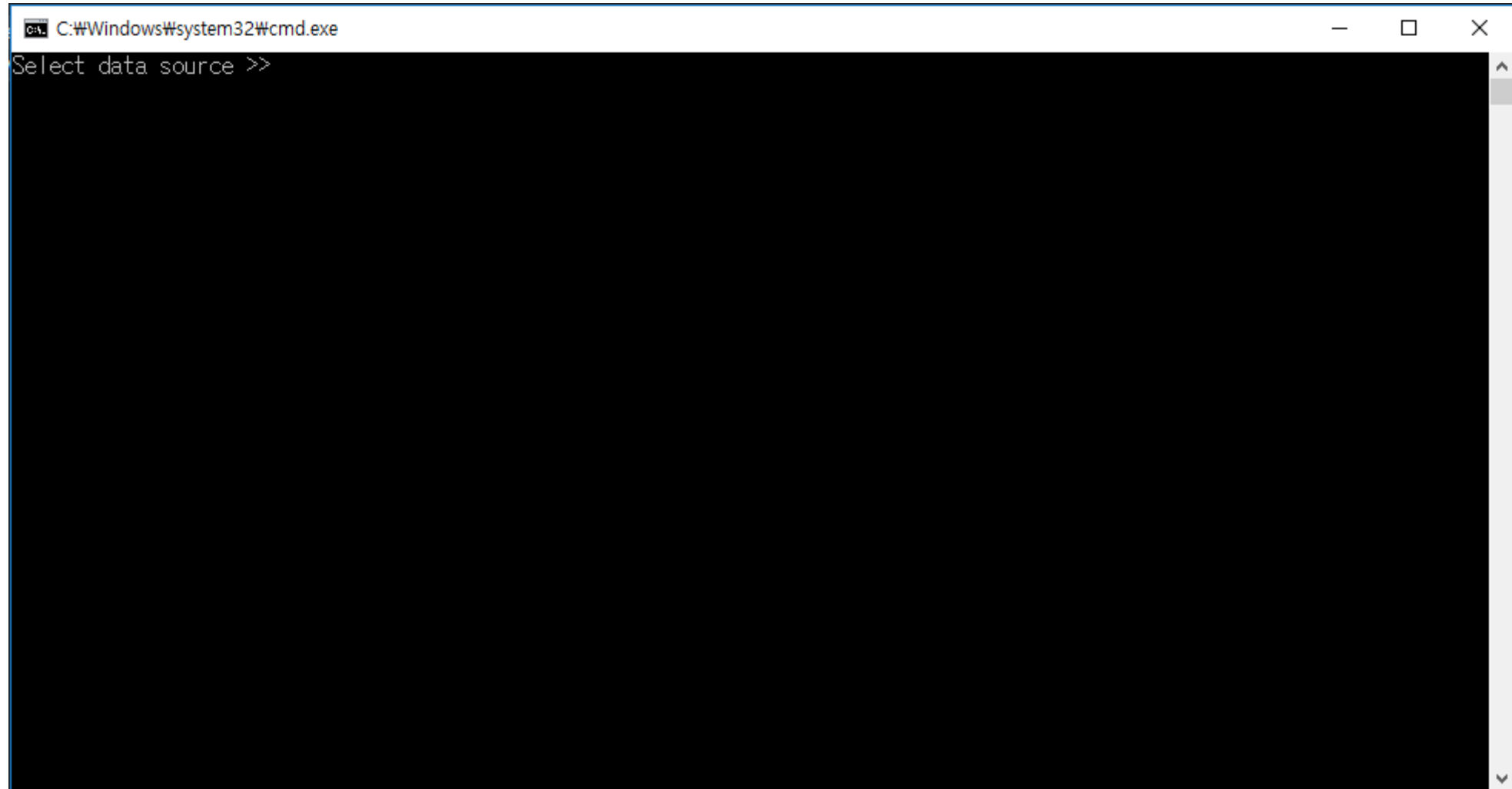


# File structure : Server

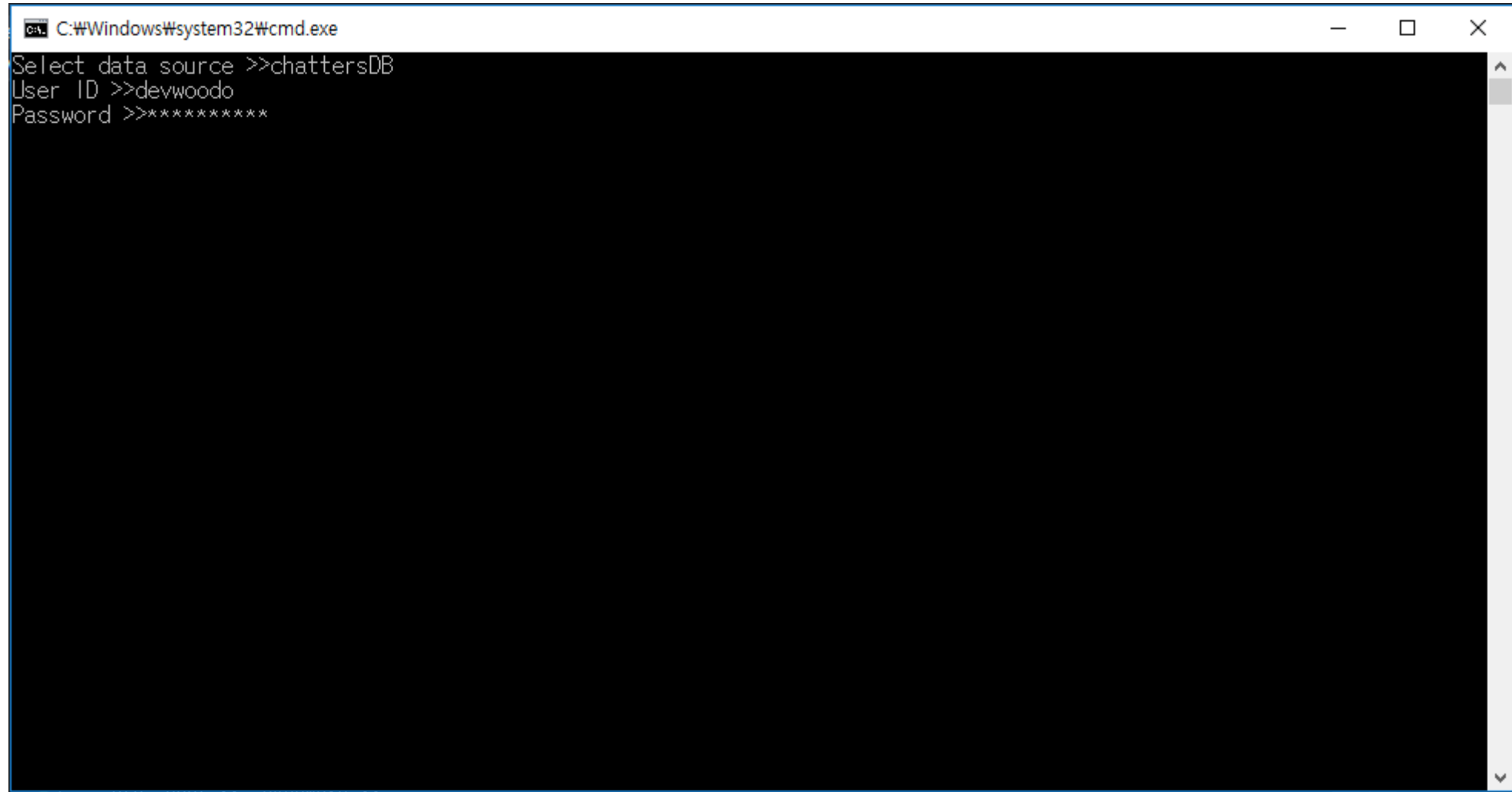
- header\_common.h : 서버와 클라이언트에 공통적으로 쓰이는 data class를 정의.
- Header\_server.h : 서버에 쓰이는 data class를 정의
- PTYPE.h : packet에 따른 고유 번호를 정의.
- packet.h : 모든 packet의 base class 정의. Packet manage의 base class 정의.
- PT\_CS\_Data.h : 클라이언트에서 서버로 가는 packet 정의.
- PT\_SC\_Data.h : 서버에서 클라이언트로 가는 packet 정의.



# 실행화면 1. Login server(Database) 초기화면



## 실행화면 2. Login Server Connecting

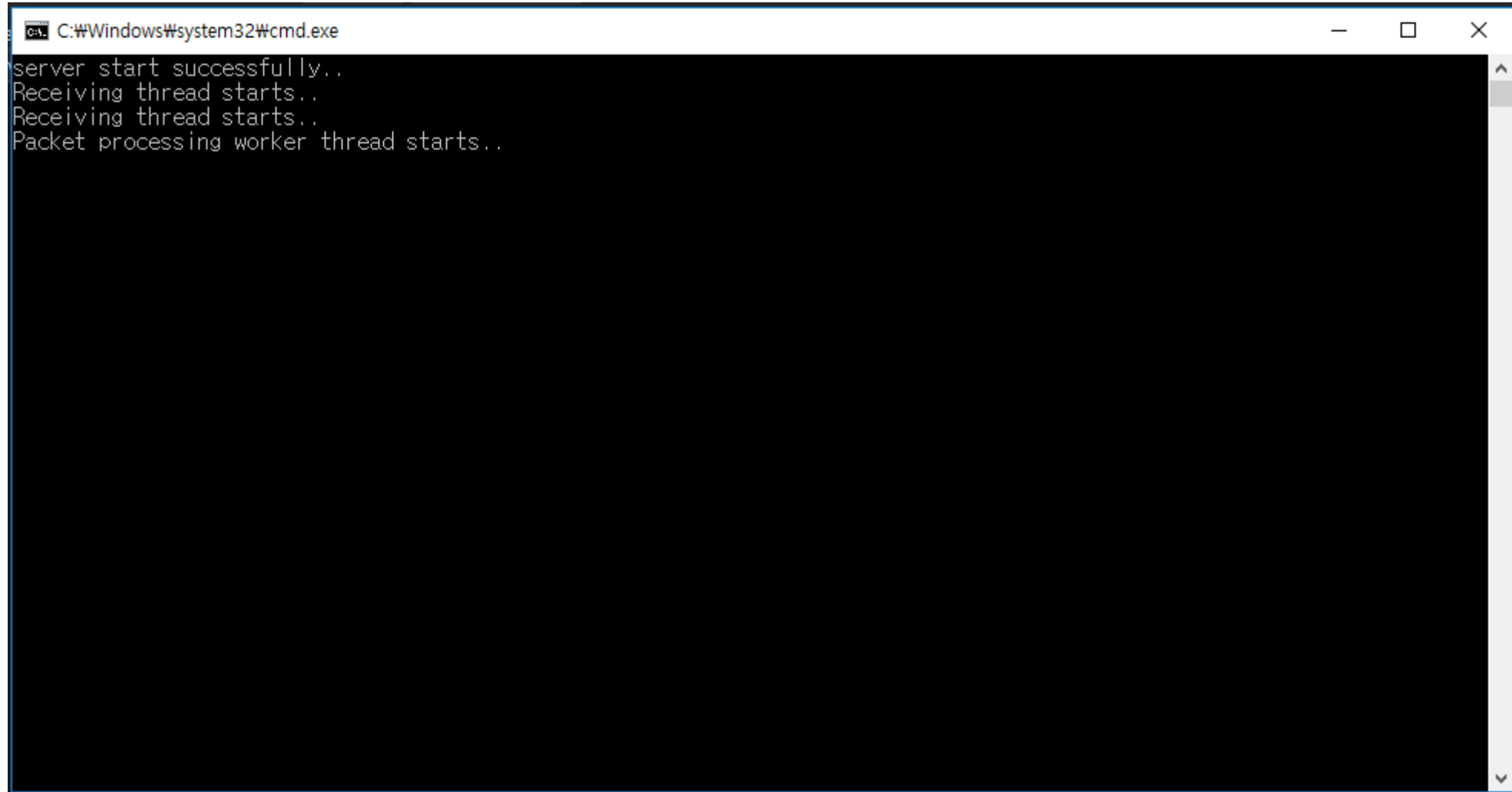


A screenshot of a Windows command prompt window. The title bar shows the path `C:\Windows\system32\cmd.exe`. The command history is as follows:

```
Select data source >>chattersDB
User ID >>devwoodo
Password >>*****
```

The rest of the window is black, indicating that the command execution has completed or is waiting for further input.

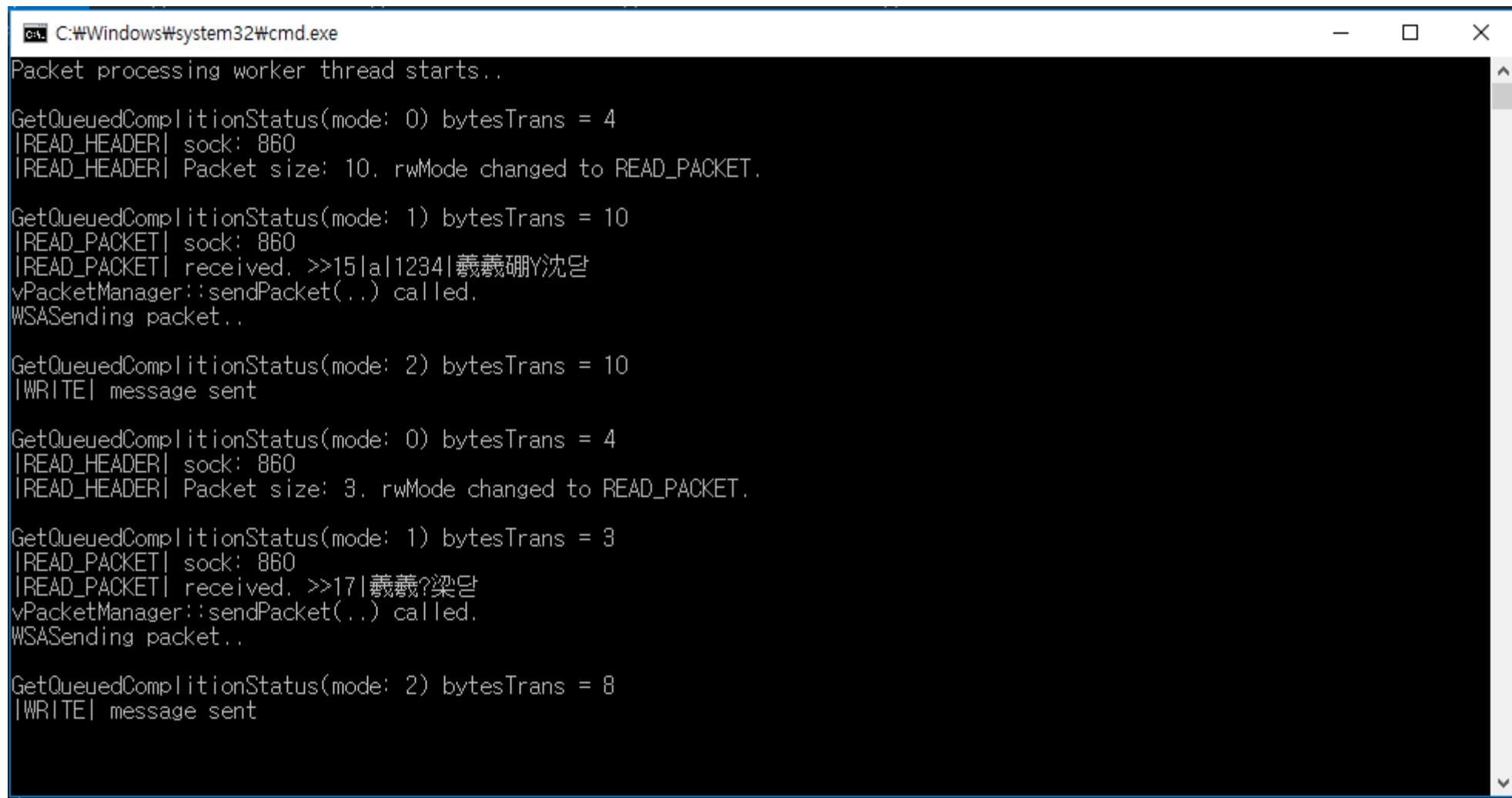
## 실행화면 3. Chatting Server Starting



```
C:\Windows\system32\cmd.exe
server start successfully..
Receiving thread starts..
Receiving thread starts..
Packet processing worker thread starts..
```

The image shows a Windows command prompt window with a black background and white text. The title bar at the top reads "C:\Windows\system32\cmd.exe". The command prompt displays four lines of output: "server start successfully..", "Receiving thread starts..", "Receiving thread starts..", and "Packet processing worker thread starts..". A vertical scrollbar is visible on the right side of the window.

## 실행화면 4. Packet Sending & Receiving



```
C:\Windows\system32\cmd.exe
Packet processing worker thread starts..

GetQueuedCompletionStatus(mode: 0) bytesTrans = 4
|READ_HEADER| sock: 860
|READ_HEADER| Packet size: 10. rwMode changed to READ_PACKET.

GetQueuedCompletionStatus(mode: 1) bytesTrans = 10
|READ_PACKET| sock: 860
|READ_PACKET| received. >>15|a|1234|羲羲礪Y沈달
vPacketManager::sendPacket(..) called.
WSASending packet..

GetQueuedCompletionStatus(mode: 2) bytesTrans = 10
|WRITE| message sent

GetQueuedCompletionStatus(mode: 0) bytesTrans = 4
|READ_HEADER| sock: 860
|READ_HEADER| Packet size: 3. rwMode changed to READ_PACKET.

GetQueuedCompletionStatus(mode: 1) bytesTrans = 3
|READ_PACKET| sock: 860
|READ_PACKET| received. >>17|羲羲?梁달
vPacketManager::sendPacket(..) called.
WSASending packet..

GetQueuedCompletionStatus(mode: 2) bytesTrans = 8
|WRITE| message sent
```

# Code Sample : Server 1 / 3

```
/* *****  
 * DBConnector class  
 *  
 ***** */  
typedef struct DBConnector  
{  
public:  
    /* member method */  
    DBConnector();  
    ~DBConnector();  
  
    RETCODE connect();  
    void close();  
    RETCODE excute(const std::string & stmt);  
    //void getResultSet(); //rev  
    RETCODE getResultNum(int & number);  
  
    // accessor  
    SQLHENV hEnv() const;  
    SQLHDBC hDbc() const;  
    SQLHSTMT hStmt() const;  
public:  
    /* member field */  
private:  
    /* member method */  
    void _init();  
private:  
    /* member field */  
    SQLHENV _hEnv;  
    SQLHDBC _hDbc;  
    SQLHSTMT _hStmt;  
  
    std::wstring _dsnName;  
    std::wstring _userID;  
    std::wstring _pswd;  
  
    bool _connectionFlag;  
} DBC;
```

# Code Sample : Server 2 / 3

```
/*
 * SvUserInfo class
 * - server에 접속 중인 client의 user information을 저장하는 클래스
 */
class SvUserInfo
{
public:
    /* Member method */
    SvUserInfo(const std::string& id, SOCKET socket);
    SvUserInfo(const SvUserInfo& ui);
    SvUserInfo(SvUserInfo&& ui);
    ~SvUserInfo();

    SvUserInfo& operator= (SvUserInfo&& ui);
    SvUserInfo& operator= (const SvUserInfo& ui);

    // accessor
    SOCKET get_socket() const;

    // mutator
    void set_socket(SOCKET socket);
public:
    /* Member field */
    UserInfoToken utk; // user 정보를 담고 있는 token
    int curRmNum;      // 현재 참여 중인 채팅방, 없을 시 -1
private:
    /* Member method */
    SvUserInfo();
private:
    /* Member field */
    SOCKET _socket;
};
```

```
class SvUserInfoManager
{
public:
    /* Member method */
    SvUserInfoManager();
    ~SvUserInfoManager();

    std::shared_ptr<SvUserInfo> add(std::string id, SOCKET sock);
    std::shared_ptr<SvUserInfo> remove(UserKey uKey);
    std::shared_ptr<SvUserInfo> find(UserKey uKey);

    // accessor
    const std::unordered_map<UserKey, std::shared_ptr<SvUserInfo>> & get() const;
public:
    /* Member field */

protected:
    /* Member method */
    SvUserInfoManager(const SvUserInfoManager&);
    SvUserInfoManager(SvUserInfoManager&&);
    SvUserInfoManager& operator=(const SvUserInfoManager&);
    SvUserInfoManager& operator=(SvUserInfoManager&&);

    void clear();
protected:
    /* Member field */
    std::unordered_map<UserKey, std::shared_ptr<SvUserInfo>> _userList;
};
```



# Code Sample : Server 3 / 3

```

/*****
 * SvPacketManager class
 * - Singleton pattern.
 *
 *****/
class SvPacketManager : public PacketManager_Base
{
public:
    /* Member method */
    static SvPacketManager& Instance();

    void sendPacket(std::shared_ptr<Packet_Base> spPk); // transmit packet via network.
    std::shared_ptr<Packet_Base> recvPacket(); // get packet from incoming packet queue.
public:
    /* Member field */
protected:
    /* Member method */
    SvPacketManager();
    SvPacketManager(const SvPacketManager&);
    SvPacketManager(SvPacketManager&&);

    SvPacketManager& operator=(const SvPacketManager&);
protected:
    /* Member field */
    static SvPacketManager _instance;
    std::queue<std::shared_ptr<Packet_Base>> _msgQueue; // incoming packet queue

    friend DWORD WINAPI recvThreadMain(LPVOID);
    friend DWORD WINAPI packetProcessWorkerThreadMain(LPVOID);
};

```

```

/*****
 * etc. functions
 * - 기타 함수 선언
 *****/
DWORD WINAPI recvThreadMain(LPVOID pComPort);
DWORD WINAPI packetProcessWorkerThreadMain(LPVOID pComPort);
void ErrorHandler(char * message);

```

## 블로그 & Github 주소

- 블로그 주소 : <https://devwoodo.blogspot.kr>
- Portfolio repos. 주소 : <https://github.com/devwoodo/chatters>