RoomChatServer v1.01

커밋된 **Git** 이름 : 로그인시 유저가 가지고있는 카드 불러오기까지
2276fe352061259fcabc11568009fc96154fb327

Card 관리

CCardManager
+ CCardManager()
+ bool ChangeUserCardAmount(const char* textName, const int& curCardAmount, const int& userPKnum, const int& cardNum)

typedef
list<shared_ptr<CChannel>> ChannelList
ChannelList::iterator ChannelListIt
list<shared_ptr<CLink>> LinkList
LinkList::iterator LinkListIt
list<shared_ptr<CRoom>> RoomList
RoomList::iterator RoomListIt
list<shared_ptr<MyCardInfo>> MyCardList
MyCardList::iterator MyCardListIt

ConstEnumInfo.h
```
const int Port = 9000;
const int BufSize = 1024;
const int ExcelBufSize = 4096;
const int NameSize = 30;
const int IdPwSize = 256;
const int IntSize = 4;
const int MakeThreadNum = 3;
const int EnterChannelNum = 1; // 처음 접속 채널
const int EnterRoomPeopleLimit = 2;
const int MaxChannelNum = 5;
const int ChannelAmount = 5;
const int NoneRoom = -1;
const int CardNumCols = 0; // 엑셀파일 카드 번호 열번호
const int CardNameCols = 1;
const int CardProbCols = 2;
const int CardStatCols = 3;
const int CardCost = 20;
const int StartMoney = 100; // 시작 머니
const int ErrorCardNum = 999;
 const string
StartCardInventory("01/00I02/00I03/00I04/00I05/00I06/00I07/00I08/00I09
const string NameMemberInfoTxt = "MemberInfo.txt";
const string NameMemberCardInfoTxt = "MemberCardInfo.txt";
const string CardEmpty = "00";
 const int CardNameBuf = 64;
const string MakeNextJoinNumberTxt
```

클래스(객체여러개)
클래스(한개)
private 멤버변수 &함수
생성자
public 멤버함수
public 멤버함수
구조체
주석처리

로그인/회원가입

CLobby
- MessageStruct MS
- int Login(SOCKET& clientSocket, CActionNetWork& actionNetWork, vector<string>& tempUserInfo)
- int JoinMember(SOCKET& clientSocket, CActionNetWork& actionNetWork, vector<string>&
- int ChooseMenu(char* message, SOCKET & clientSocket, CActionNetWork & actionNetWork)
- int SendMenuInfo(SOCKET & clientSocket, CActionNetWork & actionNetWork)
- int NextUserNum
+ CLobby(int NextUserNum_)
+ CLobby(const CLobby&) = delete
+ CLobby& operator=(const CLobby&) = delete
+ MessageStruct& getMessageStruct()
+ int ActionServiceLobby(SOCKET& clientSocket, CActionNetWork& actionNetWork, vector<string>&

연결 담당

CReadyNetWork
- SOCKET* hServSock
+ CReadyNetWork()
+ CReadyNetWork(const CReadyNetWork&) = delete
+ CReadyNetWork& operator=(const CReadyNetWork&) = delete
+ void Accept(SOCKET& hClientSock)

주고 받기 담당

CActionNetWork
- MessageStruct sendClientMessage
+ CActionNetWork()
+ CActionNetWork(const CActionNetWork&) = delete
+ CActionNetWork& operator=(const CActionNetWork&) = delete
+ int sendn(CLink& clientInfo, CRoomManager& roomManager, CChannelManager& channelManager, int flags = 0)
+ int sendn(SOCKET& socket, MessageStruct& MS, int flags = 0)
+ int recvn(shared_ptr<CLink> clientInfo, CCommandController& commandController, int flags = 0)
+ int recvn(SOCKET& socket, MessageStruct& MS, int flags = 0)
+ int sendMyName(SOCKET& clientSocket, CLink& clientInfo, int flags = 0)
+ int askClient(SOCKET& clientSocket, MessageStruct& MS, char* question)
+ int notificationClient(SOCKET& clientSocket, MessageStruct& MS, char* notification)

카드 뽑기 담당

CGaChar
- CGaChar()
+ CGaChar(const CGaChar& copy) = delete
+ CGaChar& operator=(const CGaChar& copy) = delete
+ int randNumber(int max = 100)
+ Card* gaCharResult(int range)

Client 받은 명령 처리

CCommandController
- CRoomHandler RoomHandler;
- CChannelHandler ChannelHandler;
- CCardManager mCardManager
- CGaChar mGacharHandler
- int cardSelect(shared_ptr<CLink> shared_clientInfo, MessageStruct* sendClientMessage)
- int readyCommand(shared_ptr<CLink> shared_clientInfo, CLink*& clientInfo, int& channelNum)
- int enterRoom(shared_ptr<CLink> shared_clientInfo)
- int changeChannel(shared_ptr<CLink> shared_clientInfo)
- int makeRoom(char * command, shared_ptr<CLink> shared_clientInfo)
- int outRoom(shared_ptr<CLink> shared_clientInfo)
- int mergeRoom(shared_ptr<CLink> shared_clientInfo)
- CRoomManager RoomManager
- CChannelManager ChannelManager
+ CCommandController( )
+ CCommandController(const CCommandController&) = delete
+ CCommandController& operator=(const CCommandController&) = delete
+ int commandHandling(shared_ptr<CLink> shared_clientInfo, char* command, MessageStruct*
+ CChannelHandler& getChannelHandler()
+ CChannelManager& getChannelManager()
+ CRoomManager& getRoomManager()
+ bool deleteClientSocket(CLink& clientInfo)

Client 채널 입/출 담당

CChannelHandler
+ CChannelHandler()
+ CChannelHandler(const CChannelHandler&) = delete
+ CChannelHandler& operator=(const CChannelHandler&) = delete
+ bool moveNextChannel(shared_ptr<CLink> shared_clientInfo, CChannelManager& channelManager, int targetChannelNo)
+ bool exitChannel(CLink& clientInfo, CChannelManager& channelManager)

Client Room 만들기 입/출 담당

CRoomHandler
+ CRoomHandler()
+ CRoomHandler(const CRoomHandler&) = delete
+ CRoomHandler& operator=(const CRoomHandler&) = delete
+ bool exitRoom(CLink* clientInfo, CRoomManager* roomManager)
+ bool makeRoom(shared_ptr<CLink> shared_clientInfo, CRoomManager* roomManager, char* roomName)
+ bool enterRoom(shared_ptr<CLink> shared_clientInfo, CRoomManager* roomManager, int targetRoomNo)
+ char* returnRoomName(char* message)

Client 관리

CChannel (객체가 여러개)
- LinkList ClientInfos
- int ChannelNum
- MUTEX RAII_ChannelMUTEX
+ CChannel(int channelNum);
+ CChannel(const CChannel&) = delete
+ CChannel& operator=(const CChannel&) = delete
+ int getChannelNum()
+ LinkListIt getIterMyInfoBegin()
+ LinkListIt getIterMyInfoEnd()
+ void pushClient(shared_ptr<CLink> shared_client)
+ LinkListIt eraseClient(LinkListIt myInfoListIt)

채널 관리

CChannelManager
- ChannelList Channels
- MUTEX RAII_ChannelManagerMUTEX
- void pushChannel(shared_ptr<CChannel> shared_newChannel)
+ CChannelManager()
+ CChannelManager& operator=(const CChannelManager&) = delete
+ CChannelManager(const CChannelManager&) = delete
+ ChannelListIt getIterChannelBegin()
+ ChannelListIt getIterChannelEnd()
+ CChannel * getMyChannel(int ChannelNum)

Room 관리

CRoomManager
- RoomList Rooms
- MUTEX RAII_RoomManagerMUTEX
+ CRoomChannelManager()
+ CRoomManager(const CRoomManager&) = delete
+ CRoomManager& operator=(const CRoomManager&) = delete
+ void pushRoom(shared_ptr<CRoom> shared_newRoom)
+ RoomListIt eraseRoom(RoomListIt delRoom)
+ RoomListIt getMyRoomIter(int ChannelNum, int roomNum)
+ RoomListIt getIterRoomBegin()
+ RoomListIt getIterRoomEnd()
+ int getEmptyRoomNum()
+ bool isRoomListEmpty()

Client 관리

CRoom (객체가 여러개)
- LinkList ClientInfos
- char* RoomName
- int ChannelNum
- int RoomNum
- int AmountPeople
- MUTEX RAII_RoomMUTEX
- void increasePeople()
- void decreasePeople()
+ CRoom(int roomNum,int channelNum, char* roomName)
+ CRoom(const CRoom&) = delete
+ CRoom& operator=(const CRoom&) = delete
+ void pushClient(shared_ptr<CLink> shared_client)
+ LinkListIt eraseClient(LinkListIt myInfoListIt)
+ int getRoomNum()
+ int getChannelNum()
+ char* getRoomName()
+ LinkListIt getIterMyInfoBegin()
+ LinkListIt getIterMyInfoEnd()
+ int getAmountPeople()
+ bool mergeRoom(CRoom* targetRoom)

Client 정보 보관 담당

CLink (객체가 여러개)
- char* Name
- int MyRoomNum
- int MyChannelNum
- SOCKET& ClientSocket
- MessageStruct MS
- int MyMoney
- MyCardList mMyCards
- MUTEX RAII_LinkMUTEX
- const int MyPKNumber
+ CLink(SOCKET& clientSocket, string strPKNumber, char* name_)
+ CLink(const CLink&) = delete
+ CLink& operator=(const CLink&) = delete
+ MessageStruct& getMessageStruct()
+ SOCKET& getClientSocket()
+ int getMyRoomNum()
+ int getMyChannelNum()
+ char* getMyName()
+ void setDefaultName()
+ void setMyRoomNum(int myRoomNum)
+ void setMyChannelNum(int myChannelNum)
+ bool isMoneyOKGaChar()
+ void pushCard(Card* card)
+ void changeName(const char* name, int start)
+ MyCardListIt GetIterMyCardBegin()
+ MyCardListIt GetIterMyCardEnd()
+ bool isEmptyCard()
+ void EmptyCard()
+ bool isHavsCard(int cardNum, MyCardListIt& cardIter)
+ void initCard(Card* card, int amount = 1, float exp = 0.0f)
+ const int getCardNumber()
+ const int GetMyPKNumber()const

RAII

MUTEX
- mutex m
+ MUTEX()
+ void lock()
+ void unlock()

RAII
MUTEX
CRITICALSECTION
<typename T> ScopeLock

CRITICALSECTION
- CRITICAL_SECTION CS
+ CRITICALSECTION()
+ void lock()
+ void unlock()

<typename T> ScopeLock
- T& obj
+ ScopeLock(T* o)
+ ScopeLock(T& o)
~ScopeLock()

구조체

카드 정보

struct Card
+ int cardNum
+ char* name
+ int prob
+ int stat
+ Card(int num, char* cardName, int prob_, int stat_)
+ Card(const Card& copy) = delete
+ Card& operator=(const Card& copy) = delete

메세지 정보

struct MessageStruct
+ char* message
+ size_t sendRecvSize
+ MessageStruct() :message(new char[BufSize])
+ MessageStruct& operator=(const MessageStruct& copyMS)
+ MessageStruct(const MessageStruct& copyMS) :sendRecvSize(copyMS.sendRecvSize)

내가 가지고 있는 Card 정보

MyCardInfo
- shared_ptr<Card> card
- int amount
- float exp
+ MyCardInfo(const shared_ptr<Card>& card_, int amount_ = 1, float exp_ = 0.0f)
+ MyCardInfo(const MyCardInfo& copy) = delete
+ MyCardInfo& operator=(const MyCardInfo& copy) = delete
+ void increaseCard()
+ int getAmount()
+ char* getCardName()

정적 클래스 객체

static
+ CReadHandler* ReadHandlerStatic
+ CWriteHandler* WriteHandlerStatic
+ CCard* CardStatic

txt읽기 담당

CReadHandler
- CReadHandler()
- bool ReadUserCardLine(const string& textFileName, const int& userPKNum, vector<string>&
- const string GetLastLine(const string& textFileName)
+ CReadHandler(const CReadHandler&) = delete
+ CReadHandler& operator=(const CReadHandler&) = delete
+ static CReadHandler* getInstance()
+ bool Search(const char* textFileName, vector<string>& tempUserInfo, int count, ...)
+ vector<string> Parse(const string& str, const char& ch)
+ bool ReadUserCard(CLink* client, const string textFileName)
+ bool ReadUserCard(CLink* client, const string& textFileName)
+ const string GetNextUserNum(const string& textFileName)

txt쓰기 담당

CWriteHandler
- CWriteHandler()
+ CWriteHandler(const CWriteHandler&) = delete
+ CWriteHandler& operator=(const CWriteHandler&) = delete
+ static CWriteHandler* getInstance()
+ bool write(const char* textFileName, int count, ...)
+ void WriteCard(const char * textName, int offset, int cardNum, int
+ writeNextJoinUserNum(const string& textFileName,const int&

Card 정보 담음

CCard
- CardList mCards
- CCard()
+ CCard(const CCard& copy) = delete
+ CCard& operator=(const CCard& copy) = delete
+ static CCard* getInstance()
+ void pushCard(shared_ptr<Card> card)
+ CardListIt getCardListIterBegin()
+ CardListIt getCardListIterEnd()

Error 처리 담당

CErrorHandler
- static EnumErrorCode CriticalError(EnumErrorCode code)
- static EnumErrorCode TakeError(EnumErrorCode code)
- CErrorHandler()
- CErrorHandler(const CErrorHandler&) = delete
- CErrorHandler& operator=(const CErrorHandler&) = delete
+ static EnumErrorCode ErrorHandler(EnumErrorCode code)

Util.h

함수 이름
+ static void IntToAlphabet(const int num, char* chResult)
+ static const string IntToString(const int& targetInt)
+ static const int AddCipHer(const int Number)

ErrorCode.h
```
enum EnumErrorCode
{
// 취소
Cancel = 4444,

// recv관련 10~19 return
SUCCES_RECV = 10,
// recv 에러
ERROR_RECV = 11,

// enum으로 만들어라

// send관련 20~29 return
SUCCES_SEND = 20,
// send 에러
ERROR_SEND = 21,

////////// 기타 관련 60~89 return//////////
// 잘못 입력
ERROR_WRONG_INPUT = 69,
// 클라이언트 묻기 성공/
SUCCES_ASKCLIENT = 70,
// 클라이언트 통보 성공
SUCCES_NOTIFICATION = 72,
// 로그인 성공
SUCCES_LOGIN = 74,
// 로그인 실패
ERROR_LOGIN = 75,
OVERLAPID = 76,
SUCCES_JOIN = 78,
ERROR_JOIN = 79,
SUCCES_MENUOUT = 80,
ERROR_MENUOUT = 81,
ERROR_GET_CHANNEL = 83,
ERROR_GET_ROOM = 85,
ERROR_ENTER_CHANNEL = 87,
SUCCES_COMMAND = 88,
ERROR_COMMAND = 89,
ERROR_EXIT_CHANNEL = 91,
ERROR_EXIT_ROOM = 93,
ERROR_MAKE_ROOM = 95,
ERROR_ENTER_ROOM = 97,
ERROR_DELETE_SOCKET = 99,
ERROR_SHARED_COUNT_ZORO = 101,
ERROR_MONEY_FAIL = 103,
SUCCES_COMMAND_MESSAGE = 104,
ERROR_GACHAR = 105,
SUCCES_GACHAR = 106,
 ERROR_NEWCARD = 107,
 ERROR_SAVE_CARD = 109, // 유저 보유 카드 저장 에러
ERROR_INIT_USER_CARD = 111, // 유저 로그인시 보유 카드정보 불러오기 실패
//////////////////////////////////
// 예외처리가 안된 오류
ERROR_EXCEPTION = 9876
```