浙江大学

# 面向对象程序设计-课程设计个人报告


| | |
|---|---|
| 作业名称： | 扑克牌游戏程序 |
| 姓　　名： | 朱理真 |
| 学　　号： | 3190101094 |
| 电子邮箱： | 3190101094@zju.edu.cn |
| 联系电话： | 19817862976 |
| 指导老师： | 陈奇 |


2021 年 7 月 11 日

# 一、 个人贡献

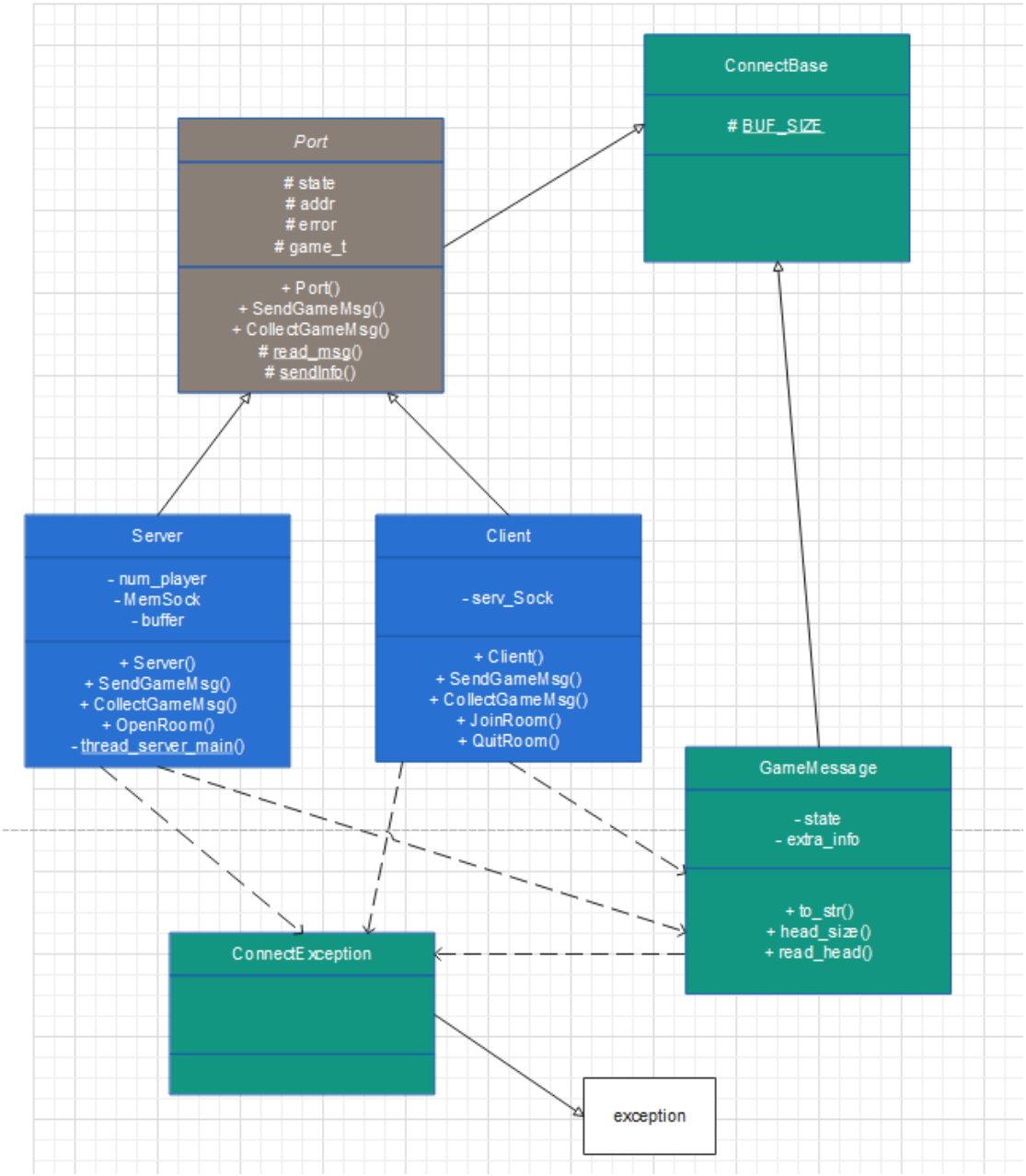
**主要负责网络模块。**

1.模块介绍

简单介绍

网络模块以 TCP/IP 协议为基础，通过调用 winsock2 库函数，利用异步 I/O 模型进行网络编程。封装实现了 Client 类和 Server 类，为其他模块提供了连接、交互基本信息和发生接收包的功能。

模块内关系




核心类介绍

ConnectBase 类

```
class ConnectBase{
    private:
    protected:
        static const int BUF_SIZE=1024;
    public:
};
```

网络模块的基本组件，提供最基础的属性。随时可以拓展，提供更多功能。

GameMessage 类：

```
class GameMessage:public ConnectBase {
    private:
        msg_t state;
        string extra_info;
        //int card_owner_index;
        //CardSet cards;
    public:
        int head_size() { return sizeof(state) + sizeof(int); }
        int read_head(const char*);
        int to_str(char*)const;
        void setPackage(const Package& pkg);
        Package getPackage();
        static Package getPackage(char* src,int len);
        const string& getName() { return extra_info; }
        static string getName(char* src, int len) { return string(src, len); }
        void addName(const string& name);
        void setSignal(msg_t ms);
        void setGameType(GameType t);
        void setGameType(char* p);
        void setInt(int integer);
        void setInt(char* p);

        GameMessage():state(ConnMsg::MSG_PAKAGE){}
        msg_t get_state() { return state; }

        void readPackage(char* src, int len) {
            setPackage(getPackage(src, len));
        }
        GameType getGameType()const;
        int getInt()const;
};
```

最基础的网络包。包含发送接收所必要的长度、状态等信息，支持 Package 类、字符串、整数和网络状态信号的发送。可以提供更多拓展信息类型的发送。

ConnectException 类：

```
class ConnException:public std::logic_error{
public:
    ConnException(const std::string& msg):std::logic_error(msg){}
};
```

继承于 exception 类，用于网络部分的异常抛出。被多种异常类继承，表示不同的异常

原因。

Port 类

```
class Port:public ConnectBase{
    protected:
        PortState state;
        SOCKADDR_IN addr;
        vector<string>names;
        GameType game_t;
        int error;

        msg_t read_msg(SOCKET s, GameMessage& gm, Package& pkg);

        static void SendSignal(msg_t sig, SOCKET to);
        static void SendName(const string& name, SOCKET to);
        static void SendGameType(GameType gt, SOCKET to);
        static void SendInt(int integer, SOCKET to);
        static void read_buf(SOCKET s, char* dst, int size);
        static void SendInfo(const GameMessage& gm, SOCKET clntsocket);

        static int MsgBufClear(queue<GameMessage>& buf);
        //static char* read_buf(SOCKET s,void* dst,char* buf,char* cur_read,int ,int max_size);

    public:
        Port(PortState state = GamePort::GAME_OVER, USHORT port = 9190);
        virtual void SendGameMsg(const Package& p) = 0;
        virtual Package CollectGameMsg(int sender=i_server)=0;
};
```

通信端口的抽象类。提供基础的设定套接字参数、封装并发送网络包和接收并解析网络包功能。

Server 类

```
class Server:public Port{

public:
    /* for game.h */
    void SendGameMsg(const Package &p);
    Package CollectGameMsg(int sender=i_server);
    const string& GetClientNameR(int k) const;

    /* for self */
    int OpenRoom(GameType gt,int humans,int robots);
    //int StartGame();

    Server();
    ~Server();

    GameType getGameType()const{ return game_t; }
    int getHumans()const{ return humans; }
    int getRobots()const{ return robots; }
    PortState getState()const { return state; }
    void setState(PortState state) { this->state = state; }
    int isError()const { return error; }
    int isReady()const { return ready; }
```

作为服务端，继承于 Port 类，可以进行游戏房间的开放（即接收客户端）和网络包的收发。为不阻塞程序，使用异步 I/O 来处理来自多个客户端的信息，同时将其分流，各自缓存。其他模块通过 Server 接收信息时，实际是从缓存中取得的。

Client 类

```
class Client:public Port{
private:
        SOCKET serv_Sock;
        //queue<GameMessage> Buf;
public:

    Client();
    const vector<string>& JoinRoom(
        const char* IP,
        const string& name,
        GameType& gameType,
        int&pos); // 接收或超时后才返回
    int QuitRoom();

    void SendGameMsg(const Package &p);
    Package CollectGameMsg(int sender=i_server);
};
```

作为客户端，继承于 Port 类，提供加入房间，退出房间和网络包的收发功能。

GameLauncher 类：

```
class GameLauncher
{
public:
    // return: success if 0 else failure occured
    int OpenRoom(GameType gt, int humans, int robots);

    // return: success if 0 else failure occured
    virtual int StartGame();

    GameLauncher():s(),game_ptr(nullptr){}
    ~GameLauncher();
protected:
    void* game_ptr;
    Server s;
    virtual void setGame();
    virtual void clearGame();
    static unsigned WINAPI thread_main(void* LPgamelauncher);
};
```

提供服务器的打开、游戏的启用功能。

2.运行测试

为独立于其他模块，自行编写了一个模拟轮流报数游戏，以测试网络模块在游戏的各个环节是否都运行正常。

测试代码：

```
int main() {
    char c;
    cin >> c;
    switch (c) {
    case 's':test1(); break;
    case 'c':test2(); break;
    default: cout << "no operation" << endl;
    }

    return 0;
}

void test1() { // 代表服务端运行（当然作为房主，也会运行客户端）
    test_game_launcher gl; // 该类继承于GameLauncher,实现报数游戏
    Client cl;
    GameType gt;
    char c;
    int pos;
    vector <string> name;
    Package pkg(Header(1, 0), "fuckyou");

    for (int i = 0; i < imax; i++) {
        gl.OpenRoom(GameConn::Landlords_3, 3, 0);
        cout << "wait..." << endl;
        while ((name = cl.JoinRoom("127.0.0.1", "fuck", gt,
pos)).size() == 0) {
            cout << "connect failed" << endl;
            cout << "try again? (Y/n)" << endl;
            cin >> c;
            if (c == 'y' || c == 'Y') continue;
            exit(0);
        }
        cout << "connected" << endl;
        cout << pos << " : " << name[pos] << endl;


        test_player tp(cl);

        tp.Play();

        system("pause");
        cl.QuitRoom();
    }
    system("pause");
}

void test2() { // 代表客户端运行
    Client cl;
    test_game_launcher gl;
    GameType gt;
    char c;
    int pos;
    vector <string> names;
    Package pkg(Header(1, 0), "fuck");
    string name;
    cout << "your name:" << endl;
    cout << ">> ";
    cin >> name;
```

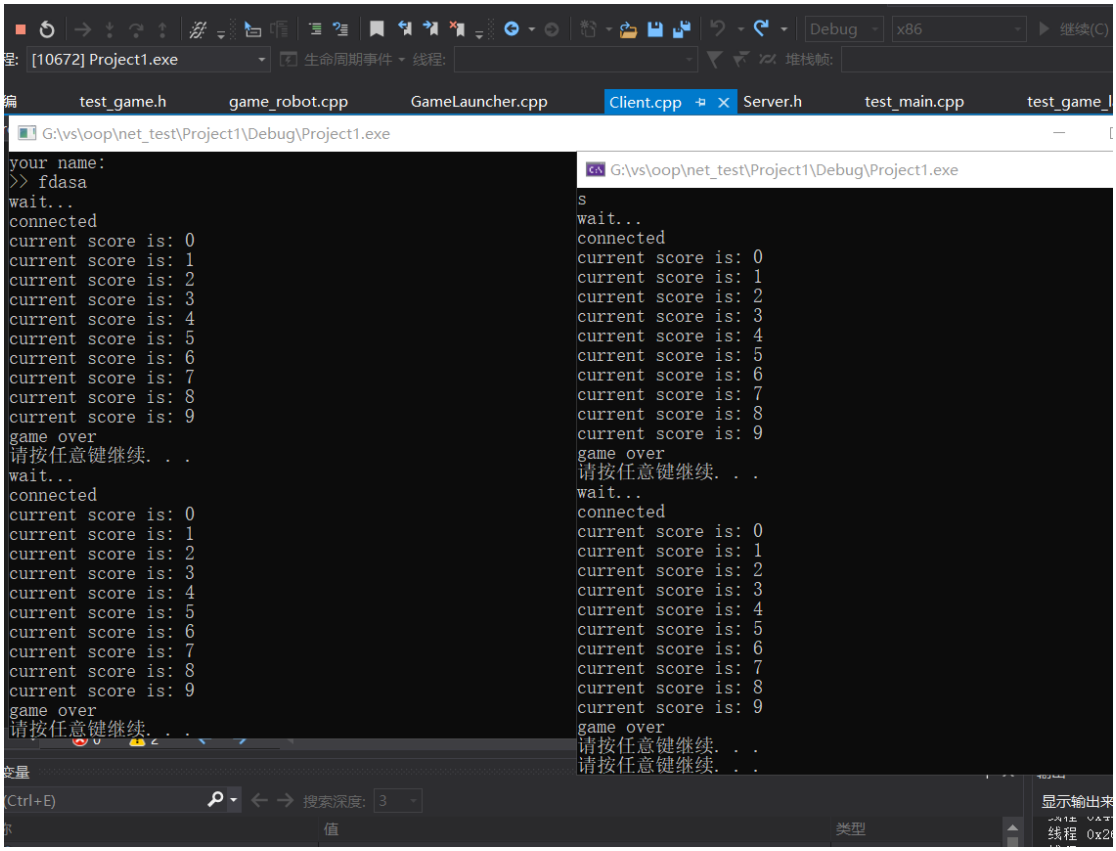
```
    for (int i = 0; i < imax; i++) {
        cout << "wait..." << endl;
        while ((names = cl.JoinRoom("127.0.0.1", name, gt, pos)).size()
== 0) {
            cout << "connect failed" << endl;
            cout << "try again? (Y/n)" << endl;
            cin >> c;
            if (c == 'y' || c == 'Y') continue;
            exit(0);
        }
        cout << "connected" << endl;
        cout << pos << " : " << names[pos] << endl;


        test_player tp(cl);
        tp.Play();

        system("pause");
        cl.QuitRoom();
    }

}
```

测试截图：




如图，使用两个客户端，进行了两局游戏，说明网络模块运行正常。

## 二、 总结与感受

此次小组合作，令我印象深刻。我不仅训练了面向对象的编程经验，还学习了网络编程和多线程编程，大大扩展了我的能力范围。同时我作为组长，与组员之间的交流又要求我具有一

定沟通能力和统筹能力。虽然期末临近，工作紧张，我们还是顶住了压力，坚持不懈地调试，最终完成了工作。对此，我还是感到开心的。

然而尽管如此，实践的过程中也有许多令人不安的部分。组队之初大家的沟通是不畅的，项目基本没有开工。只有到临近了期末，我们才因紧迫感而行动起来。甚至到了最后关头，一部分同学的工作也远远没有完成。作为组长，我认为我还欠缺规划统筹的能力，而领导者的魄力。想要在技术合作中做一个领导者，我显然还需要锻炼。无论如何，这次的合作体验，将对我以后在面向对象程序设计的编程合作中有极大的影响。