

**实验报告**



**题目： 面向对象程序设计与实践2**

**班 级： 2018211312**

**学 号： 2018211445**

**姓 名： 朱洁**

**学 院： 计算机学院**

**2020年 8 月 25 日**

目录

[一、设计任务的描述 2](#_Toc49530271)

[二、任务要求及设计分析 2](#_Toc49530272)

[任务① 2](#_Toc49530273)

[任务② 3](#_Toc49530274)

[任务③ 5](#_Toc49530275)

[三、主要数据结构说明（数据名称、用途等） 8](#_Toc49530276)

[（一）服务器端（4种主要的类）： 8](#_Toc49530277)

[（二）客户端（5种主要的类）： 11](#_Toc49530278)

[四、模块设计与关系 16](#_Toc49530279)

[五、使用说明书 17](#_Toc49530280)

一、设计任务的描述

用面向对象的设计方法来设计一款平台类对战游戏，包含三个主任务：

①宠物小精灵的加入；

②用户注册与平台登录

③游戏对战的设计。

二、任务要求及设计分析

任务①

 设计宠物小精灵的类，为简化游戏设计，精灵的属性包括种类（力量型：高攻击； 肉盾型：高生命值； 防御型：高防御； 敏捷型：低攻击间隔，共四种）、名字、等级、经验值、攻击力、防御力、生命值、攻击间隔等（以上属性必须，其他属性可自行添加）（基本要求：本游戏中只有上面的4种类型。 进一步要求：上述4种类型可以进一步深入划分，比如力量型又可以再细分为：沙瓦朗、火爆猴、腕力等）

 每个精灵初始等级为1，满级15级，每当精灵升级的时候，宠物对应的属性值会有少量增加（主属性增加量相对较多）

 每个精灵有自己独特的攻击方式，如“闪电攻击”，“火焰攻击”等等，请设计一个精灵的基类，并将精灵的攻击方法设为虚方法以方便子类重写

**设计分析**：

设计小精灵的基类，对于要求中的属性，对以下两种做了更深入的划分：攻击力分为物理攻击力和特殊攻击力，防御力分为物理防御力和特殊防御力。相应的，关于攻击和防御的种类也拓展成了两种：力量型高攻击分成了力量型高物理攻击、力量型高特殊攻击，防御型高防御分成了防御型高物理防御、防御型高特殊防御。

除了要求中包含的属性（6个种类、名字、等级、经验值、2种攻击力、2种防御力、生命值、攻击间隔），还给基类中添加以下属性：

（1）有关种族的属性：（同一个种族的小精灵以下属性都是相同的）

① speciesID（种族ID，不同种族之间不同）；

② baseEXP（基础经验值，用于计算对战胜利后的经验值提升）；

③ growPoints（升级时生命、物攻、物防等六项指标提升的基础值）

④ PokemonType（类别：水/火/草/电）

⑤ attackSpeed（攻击速度，辅助表达攻击间隔，

攻击间隔 = 特定时间 ÷ 速度,时间单位是毫秒）

⑥ learnedMoves（可以掌握的招式，每种小精灵最多掌握四项招式）

（要求里的属性中，种类、名字属于种族属性）

（2）有关个体的属性：(即使是同一个种族，不同个体之间也会有所差异)

①个体ID，每个小精灵唯一。

（要求里的属性中，等级、经验值属于个体属性）

六项指标（生命、物攻、物防、特攻、特防、速度）有个体值也有种族值。

（3）有关对战的属性：（每次对战之前都会重新初始化）

① currentHP（对战时剩余的生命值，初始化为生命值）

② accuracyLevel（命中率等级，初始化为0，根据对战会有所变化）

③ evasionLevel（闪避率等级，初始化为0，根据对战会有所变化）

④ isAtCriticalState（是否处于易中要害状态，是的话可以提高暴击率）

⑥ criticalLeftTimes（剩余可以处于易中要害状态的次数）

⑦ lastAttackTime（上次出招的时间，用于计算间隔，判断是否能出招）

⑧ moveName（某次攻击的招式名称）

⑨ damage（某次攻击给对方带来的伤害）

⑩ isHit（某次攻击是否击中对方）

1. isCritical（某次攻击是否造成暴击）

设计小精灵的子类：从基类中派生出四个子类，分别为水系小精灵，火系小精灵、草系小精灵、电系小精灵，每类小精灵都有克制和被克制的类，即水系克火系，火系克草系，草系克电系，电系克水系。基类中的攻击函数将设计为虚函数，在这四个子类中重写。

设计升级时属性的提升：小精灵每次升级，6项属性根据种族提升基础值growPoints来提升，如果是主属性，增加的幅度会基于growPoints而增大。提升的属性值还会乘上一个[0.8,1]的随机数，来增加个体之间的差异性。

任务②

 每个用户需要注册一个账号，用户名全局唯一，不能有任何两个用户名相同，要考虑注册失败的场景时的反馈

 实现注册、登录、登出功能，均采用C/S模式，客户端和服务端用socket进行通信，服务端保存所有用户的信息

 每个用户拥有：用户名、拥有的精灵，两个属性。 用户注册成功时，系统自动随机分发三个1级精灵给用户

 用户可以查看所有成功注册用户拥有精灵，也可以查看所有当前在线的用户。

**设计分析**：

（1）分别在服务器端和客户端设计用户类User：

服务器端的用户User类包含：用户id（全局唯一）、用户名（全局唯一）、密码（只在服务器端保存）、获胜次数、失败次数、拥有小精灵总数、拥有的高级小精灵的个数、是否在线标记、小精灵背包；

客户端用户User类包含服务器端用户类除密码以外的所有数据，同时对宝可梦背包做了更详细的划分，共有五个小精灵背包，一个总精灵背包和四个类别（水/火/草/电）的小精灵背包。

（2）关于通信的设计：

服务器端：

设计了一个server类实体代表服务器，一个endpoint类实体代表服务器与一个用户连接的TCP连接端点，每个TCP连接一个端点是客户端，另一个端点是服务端。全局只能有一个server类对象，一个server类对象中可以有多个endpoint类对象，两者相结合实现多用户同时访问。

server类中包含的主要数据为：①socket相关：监听套接字、已连接套接字、数据缓存；②用户相关：用户数据文件地址（保存用户名、密码、战斗次数、拥有宝可梦等信息）、用户列表、endpoint表。

endpoint类包含的主要数据为：①socket相关：端口号、监听套接字、已连接套接字、数据缓存；②用户相关：所连接的客户端代表的用户ID、用户名、用户在线情况、等待用户连接情况等。

server类对象主要负责实现用户的注册和登录、用户数据文件的加载和更新、对所有endpoint类对象的管理。server类对象只与客户端进行TCP短连接（即完成一次数据交互就断开连接）。当用户需要注册或登录时，需要与server类中的套接字（端口号固定为8000）进行连接和数据传输，server类正确接收到注册/登录请求时，会进行相关操作（如检查用户名的合法性、检查用户名密码是否匹配、检查用户是否重复登录、为新注册用户分配小精灵等），并更新用户数据文件，返回给客户端相关信息，随后断开与该客户端连接。

当用户成功登录后，server类会创建一个endpoint类对象，将该endpoint类对象的端口发给客户端，随后断开与客户端的连接，该endpoint对象将与客户端进行后续的连接和数据交互。有几个客户端登录了对战系统，就有几个endpoint类对象为其服务。

一个endpoint类对象只为一个客户端服务，其与客户端建立的是TCP长连接，可进行多次的数据交互。endpoint类对象主要负责处理与它连接的用户的其他请求，如登出、查看其他用户、战斗等。当客户端意外掉线，为其服务的endpoint类对象会等待该用户重新连接，若五分钟内该用户重新登录成功，endpoint类对象会继续为这个用户服务。当客户端的用户登出或意外退出且5分钟内未重连后，该endpoint对象将会被server类对象销毁。

当server类对象被关闭时，所有endpoint类对象也会被销毁。

客户端：

设计了一个client类代表用户。client类包含的主要数据有：

①通信相关：客户端套接字、要连接的服务端端口号、数据缓存、是否正在等待服务端回信标志位、请求是否失败标志位；

②用户相关：用户名、密码、本机用户（User类）、当前正在查看的用户（User类）、一些辅助界面状态转换和判断的数据；

③界面相关：提示信息、上一个界面状态

每个客户端程序只有一个client类对象代表本机用户。当用户请求注册或注册时，界面会要求用户输入用户名和密码，client类会判断输入是否合法，在都输入了合法的用户名和密码后才会与服务器端的server类对象建立连接和发送请求。登录请求得到允许后，client类对象会得到endpoint类的端口号，client将改为与该endpoint连接，登录后的所有请求都是发送给endpoint类。

每次客户端向服务器端发送完请求后会进入阻塞状态，直至收到答复。当客户端接收到服务器端“接收错误”或“拒绝”等非肯定的答复后，会提醒要求用户重新操作，然后客户端会重新发送请求给服务器端。

（3）关于界面的设计：

服务器端可以不需要设计界面，客户端设计了界面以方便与用户进行交互。本任务阶段共设计了个界面，每个界面相当于客户端的一种状态，负责实现或展示不同的功能与信息：

①起始界面：用户打开软件时的界面，主要负责用户名/密码的输入，发送注册/登录的请求和展示对用户的提示信息；

②主界面：用户登录成功后所进入的界面，负责接收用户想查看本人小精灵（点击背包区域）、查看其他用户、进行升级赛/决斗赛、登出（点击相应按钮）等的信号，并向服务器发送相应的请求。

③查看用户界面：查看其他用户的界面，用于查看其他用户的详细信息，通过此界面的背包区域还可转换到当前正在查看用户所拥有的小精灵的界面。也可切换回主界面。

④小精灵界面：展示本人/某其他用户的小精灵的详细信息。可切换回主界面/查看用户界面。

用户点击界面上的相关区域，客户端就能知道需要向服务器端发送什么请求。客户端收到服务器端的答复后，若是肯定的答复，则会转换到相应的下一界面，否则就会停留在当前界面，并提示用户请求失败，需重新操作。

任务③

 已经登录的在线用户可以和服务器进行虚拟决斗，决斗分两种：升级赛和决斗赛，两种比赛都能增长精灵宠物经验值。服务器上有一个虚拟精灵的列表，用户可以挑选其中任意一个进行比赛（升级赛或者决斗赛）。另外决斗赛中用户胜出可以直接获得该战胜的精灵，失败则系统从用户的精灵中随机选三个（不够三个精灵的情况就选择他所有的精灵），然后由用户选一个送出。

 升级赛 只是用户用来增加精灵经验值，规则开发者自定；

 累积多少经验值升一级，规则开发者自定；

 决斗赛的上述规则同升级赛，只是额外还可以赢得一个宠物或失去一个宠物。

 用户如果没有精灵（比如总是失败，已经全部送出去），则系统会随机放给他一个初级精灵。

 请让你的系统自动模拟每场比赛的每次出招。另外，为了增加不确定性，可以加入概率闪避攻击和暴击伤害机制

 比赛的过程和结果由系统根据上述规则自动模拟完成，要求结果具有一定的随机性。

 用户增加新功能，可以查看某个用户的胜率

 用户增加新属性，为宠物个数徽章（金银铜）和高级宠物徽章（金银铜），分别根据拥有的宠物个数的多少和拥有高级宠物（15级）个数的多少颁发

**设计分析**：

（1）关于客户端与服务器数据交互，并发请求处理等的设计如在任务②的设计分析中所介绍：

客户端的注册/登录请求和数据交互与服务器端的server类对象进行，登录后其他所有的请求和数据交互与专门的endpoint类对象进行，通过多线程，服务器端的server可以同时有多个endpoint类在运行，即可以同时处理多个客户端请求。每个endpoint之间各自独立，互不影响。

（2）关于小精灵类的设计和用户类的设计（包含了所需的新功能及属性）分别如在任务①和任务②的设计分析中所介绍，这里不再重复。下面介绍一下客户端Client类新增的设计：

在任务②设计的基础上新增了有关虚拟战斗的数据设计：本机用户出战的小精灵、对手小精灵、选择挑战的类别（升级赛/决斗赛）、挑战的模式（初/中/高级小精灵）、挑战成功本方获得的经验，挑战失败本方失去的小精灵等

（3）关于小精灵招式类Move的设计：

Move类对象为小精灵战斗时的出招，小精灵战斗时伤害的高低，能否命中，能否产生暴击受Move类对象的影响。

Move类包含主要数据：

①招式ID、名字（全局唯一）；

②招式命中（不是命中率，命中率还需进一步演算）；

③招式威力（影响伤害的高低）

④招式的类别（水/火/草/电/一般）

⑤招式的物理性（物攻/特攻）

⑥招式是否容易暴击

一个小精灵最多学会四项招式（同一种族的小精灵的招式相同），初始1级的时候只学会第一项招式，等级提升后可自动学会新的招式，等级提升至到4/8/12级时可分别学会第2/3/4项招式。战斗时，每次攻击的招式由系统自动模拟，即在小精灵已学会的招式中随机选一个作为出招。

（4）关于伤害计算方法的设计

计算公式为：伤害 =（（等级+5）/20 × 攻击/防御 × 威力 +2）× 加成

其中：

①等级是攻击方宝可梦的等级；

②攻击是攻击方的攻击或者特攻（取决于招式是物理/特殊招式），

防御是防守方的防御或特防（取决于招式是物理/特殊招式）；

③威力为出招的招式的威力；

④加成的计算公式：

加成=类别一致加成×类别相克造成的倍率×击中要害的倍率×其他加成×随机数（随机数∈[0.85，1]），其中：

a.当有类别一致加成效果时该值为1.5，否则为1。

（即招式的类别和小精灵的类别相同，如招式x的类别为水，小精灵y的类别也为水，则小精灵y攻击时出招式x可以有类别一致加成效果）

b.类别相克造成的倍率可以为0.5、1、2。

类别相克表：水克火、火克草、草克电、电克水。

（如水系小精灵攻击火系小精灵可以造成2倍伤害，攻击电系小精灵只能造成0.5倍伤害，攻击水系/草系造成1倍伤害）

c.当暴击时，该值为1.5,如果没有产生暴击，该值为1。

（暴击率的计算方法在下文中介绍）

d.其他加成是指天气加成:

共有四种天气：雨天（水系小精灵加成）、晴天（火系/草系小精灵加成）、打雷（电系小精灵加成）、普通（无加成）。

每次战斗初始，系统会随机选取一个天气作为本场战斗的战斗天气，当天气对小精灵有加成时，小精灵的命中等级和闪避等级会提升（命中率、闪避率的计算方法在下文中介绍）。

e.最后伤害要无条件舍去（即向下取整），但至少会造成1ＨＰ的伤害

（能命中的情况下，当招式未命中，对方闪避，最终伤害为0）。

（5）关于暴击率的设计：

战斗时小精灵每次出招，暴击等级初始化为0。若小精灵出招的招式是易于暴击的招式，暴击等级加一，且小精灵会处于易产生暴击的状态（状态可持续两次出招，次数可累加）；若小精灵处于易产生暴击的状态，暴击等级加二。每个暴击等级对应不同的暴击率，暴击等级越高，暴击率越高。暴击率取值（0，1）。

每次出招计算伤害时，会先计算一次暴击率，然后产生一个（0，1）的随机数，若随机数小于暴击率，则该次出招视为产生暴击。

（6）关于命中率的设计：

每次对战前小精灵的命中等级会被初始化为0，等级最高为6，超出6按6计算。根据对战时的天气和自身类别的适配性可以有提升。不同的命中等级对应不同的命中率，命中等级越高，命中率越高。

（7）关于闪避率的设计：

每次对战前小精灵的闪避等级会被初始化为0，等级最高为6，超出6按6计算。根据对战时的天气和自身类别的适配性可以有提升。不同的闪避等级对应不同的闪避率，闪避等级越高，闪避率越高。

（8）关于是否命中/闪避的设计：

战斗时每次出招是否命中对方/对方是否闪避由出招招式的命中、攻击方命中率、防御方闪避率共同决定。根据公式A = 招式的命中 × 攻击方命中率× 防御方闪避率，求出A后，生成一个[1,127]的随机数，若随机数小于A，则本次出招命中对手，否则视作未命中，对手闪避。

（9）关于战斗模拟的设计（包含部分客户端与服务端的交互设计）：

当用户想要进行战斗时，首先需要选择本方要出战的宝可梦。随后系统会要求用户选择要挑战的模式，共有三种模式：初级小精灵、中级小精灵、高级小精灵，三种模式的挑战难度逐步增大（初/中/高级小精灵指的是小精灵的各项属性值高低，等级都是一样的，与本机用户出战的小精灵相同）。选择完模式后，用户需要选择该模式下要挑战的小精灵。都选择完毕后，客户端会将用户的选择传给服务器端，服务器端将生成一个新的个体小精灵然后将相关信息返回给客户端，客户端收到小精灵信息后可以构造一个相同的小精灵，战斗即可正式开始。

战斗正式开始时，本方出战小精灵和敌方小精灵都会对战斗相关数值进行初始化。随后启动计时器，每到达小精灵的攻击间隔，系统将自动模拟小精灵的出招，进行伤害计算和更新剩余血量。当有一方血量不足时，战斗结束。战斗的模拟由客户端自行完成，战斗过程中不需要服务器端的参与。战斗结束后，客户端需更新用户小精灵的经验值属性等信息，再将结果传给服务器端，服务器端要将这些内容同步到数据文件中，随后返回给客户端“完成”的信号接触客户端的阻塞状态。

（10）关于界面设计：

在任务②的基础上，新增四个界面：战斗选择界面（共三个阶段）和战斗界面。

选择界面的前两个阶段（挑选本方出战小精灵、选择挑战模式）不会跟服务器端交互，只有当第三个阶段（挑选挑战小精灵）也点击确认后，才会与服务器端进行数据交互。

三、主要数据结构说明（数据名称、用途等）

（一）服务器端（4种主要的类）：

**①Server类：**

**变量**：**（均为private）**

**//通信相关**

SOCKET listenFd; //监听套接字（本质上是文件描述符）,接受来自客户端的连接请求

SOCKET connectFd; //已连接套接字（本质上是文件描述符）,被用来与客户端进行通信

char socketBuffer[1024]; //数据缓存

volatile bool isRunning;//服务器是否正在工作

**//用户数据相关**

std::string userFileLoc;//用户数据文件地址，保存用户名、密码及拥有宝可梦等信息

std::vector<User\*> userList;//用户表

std::vector<Endpoint\*> serverEndpoints;//用于实现多用户访问，TCP连接的服务端点

std::mutex serverMutex; //互斥锁，多线程，用来保护数据

**方法：**

**（private）**

**//以下两种功能由客户端连接到服务器主端口（SERVER\_PORT）完成**

void SignUp(std::string userName, std::string password);//注册

void LogIn(std::string userName, std::string password);//登录

int CheckExistence(std::string userName) const; //检查某用户名是否存在

**//构造函数**

Server(const Server&) = delete; //禁用拷贝构造函数,只允许有一个服务器

Server& operator=(const Server&) = delete; //禁用拷贝赋值操作符,只允许有一个服务器

**//与线程有关的函数**

void Communicate();//连接并通信

void End();//终止运行

void ManageEndpoint(Endpoint\* endpoint);//管理endpoint，监视和销毁

std::vector<std::string> SplitBuf() const; //分割接收到的数据

**//有关文件读写**

void LoadUsersData();//加载文件内容到内存

void LoadSpeciesPokemon()const;//加载种族宝可梦到内存

void LoadIndividualPKM()const;//加载个体宝可梦到内存

**（public）**

Server();//构造

~Server();//析构

bool InitServer();//初始化并启动服务器，成功则返回true，失败返回false

void Run();//运行服务器：执行线程以及在线程执行完毕后销毁连接端点,并关闭服务器

void SetUserIsOnline(int id, bool isOnline);//标记用户是否在线

std::string GetUserInfo(int id) const;//返回某id的详细信息

std::string CreateBattlePkm(int species, int lv,bool isUpgrateBattle)const;//给客户端构造一个对手并返回

void RewriteUserFile();//重写文件内容

void RewritePkmFile();//重写个体宝可梦文件

std::string UpdateUser(int userid, int pkmid, bool isAdd);//更新用户信息

void SetUserTimes(int userID,bool isWin);//更新用户战斗信息

**②Endpoint类：**

**变量**：**（均为private）**

Server\* server;//管理该endpoint的服务器server

int endpointPort; //端口

SOCKET listenFd; //监听套接字,可以接受来自客户端的连接请求

SOCKET connectFd; //已连接套接字,可以被用来与客户端进行通信

char socketBuffer[1024];//数据缓存

volatile bool isRunning;//本端点是否正在运行

**//所连接的用户**

int userId;//用户ID

std::string userName;//用户名

volatile bool isUserOnline;//本端点连接的用户是否在线

volatile bool isWaitingConnect;//是否正在等待用户连接

//线程

std::mutex endpointMutex;//互斥锁

std::condition\_variable conditionVariable;//条件变量，阻塞线程直到通知恢复为止

**方法：**

**（private）**

**//多线程相关函数**

void Communicate();//连接并通信

void WaitingTimer();//管理计时器

std::vector<std::string> SplitBuf() const;//分割接收到的数据

**（public）**

Endpoint(Server\* server, int id);//构造

int InitEndpoint();//初始化成功返回端口号，失败返回-1

void Run();//运行本端口

int GetUserID() const;//获得本端点连接的用户的ID

bool GetIsOnline() const;//获得本端点连接的用户的在线情况

int GetPort() const;//获得本端点的端口号

**③Pokemon类：**

**变量**：**（均为private）**

**//个体相关（个体之间有所差异的值）**

int myID; //个体小精灵编号

int level; //等级

int EXP; //经验值

**//属性值**

int healthPoint; //生命值

int attackPoint; //攻击值--物理

int defencePoint; //防御值--物理

int specialAttack; //特攻值--特殊

int specialDefence; //特防值--特殊

int attackSpeed; //速度

int attackInterval; //攻击间隔(= 固定时间 ÷ 速度),时间单位是毫秒

**//种族相关（值与所有该种类个体宝可梦一致）**

int speciesID; //种族ID

int baseEXP; //基础经验值，用于计算对战胜利后的经验值提升

int growPoints; //升级时各项指标提升的值

std::string name;//精灵名字

std::string mainAttribute;//主属性：高攻/高防/高血/高速

std::string PokemonType;//类别：水/火/草/电

int learnedMoves[4];//该小精灵掌握的招式,一只宝可梦最多只能同时记住4个招式

**方法：**

**（public）**

Pokemon(const Pokemon& copy); //构造

Pokemon(int id, std::string name, std::string type, std::string mainA, int base, int grow, int attr[6],int move[4]);//种族的

Pokemon(int myID, int speciesID, int level, int EXP, int attr[6]);//个体的

void InitIndividual(int individualID, int level);//初始化个体id和等级

void SetAttr(int attr[8]);//根据客户端传来的数据设置和更新宝可梦信息

void UpdateAttribute(int \_level);//更新宝可梦属性值

int GetIndividualID()const;//获得宝可梦个体ID

int GetLevel() const;//获得宝可梦等级

double GetRandomFactor(int min) const;//返回一个随机小数，范围是min/100 - 1

void AppendIndividualFile() const;//追加宝可梦文件

bool CheckEmpty(std::string) const;//检查文件是否为空

std::string GetPokemonInfo() const;//获得宝可梦信息

**④User类：**

**变量**：**（均为private）**

int id; //用户ID，必须唯一

int winTimes; //获胜次数

int loseTimes;//失败次数

int pokemonSum; //拥有宝可梦数

int fullLevelSum;//拥有高级宝可梦数（15级）

bool isOnline; //该用户当前是否在线

std::string name; //用户名，必须唯一

std::string password; // 密码，是否要限定标准

std::vector<Pokemon\*> pokemonBags;//该用户的精灵背包

**方法：**

**（public）**

User(int id, std::string name, std::string password);//构造

User(int id, std::string name, std::string password,int winTimes, int loseTimes,int fulllevel); //构造

int GainPokemon(int first, int second, int level);//初始获得宝可梦,返回值为获得的宝可梦的种族ID

std::string GetName()const;//获取该用户的名字

int GetID()const;//获取该用户的ID

std::string GetPassword() const;//获取该用户的密码

std::string GetUserInfo(bool isForFile) const;//获取该用户的信息

void SetIsOnline(bool isUserOnline);//设置该用户在线

void SetBattleTimes(bool isWin);//更新胜利/失败次数

void AddPokemonToBags(Pokemon\* newPokemon);//给该用户的背包添加宝可梦

int LosePokemon(int pkmID);//用户战败失去宝可梦

void AppendUserFile(std::string loc) const;//增加用户

（二）客户端（5种主要的类）：

**①Client类：**

**变量**：**（均为private）**

**//有关通信**

SOCKET sockFd; //客户端套接字

char socketBuffer[1024];

bool isWaitingMsg;//标记位，是否正在等待服务端的回信

bool isRequestFailed;//请求是否失败

int endpointPort;//负责本客户端的服务端点的端口号

**//有关用户**

std::string name;//本客户端用户名

std::string password;//本客户端用户密码

User\* localUser;//本机用户

User\* curCheckUser;//当前正在查看的用户

**//有关界面**

int curUserPage;//当前查看的用户页

int totalUserSum;//系统中的用户总数

int curCheckPkmSeq;//当前查看的宝可梦在列表的下标号

Type curCheckType;//当前查看的宝可梦的类别

int curPkmPage;//当前查看的宝可梦页码

int totalPkmSum;//当前查看的宝可梦背包所拥有的宝可梦总数

std::string hintMsg;//界面提示信息

State lastState;//上一个界面状态

**//有关战斗**

bool isUpgradeBattle;//是否是升级赛，true是升级赛，false是决斗赛

Mode chooseMode;//选择挑战的模式：初级/中级/高级

int myPkmID;//出战的宝可梦的种族ID

int enemyID;//对手的种族id

Pokemon\* myBattlePkm;//出战的宝可梦

Pokemon\* enemyPkm;//对手宝可梦

int gainedEXP;//战胜获得的经验值

int loseSeq[3];//挑选的三只即将失去的宝可梦在总背包中的下标,为-1时代表不存在

int loseID;//失去的宝可梦的个体ID

bool isChooseLose;//是否点击确认要失去的宝可梦

bool isOkRet;//是否可以返回

**方法：**

**（private）**

**//文件加载**

void LoadSpeciesPokemon();//加载种族宝可梦到资料库

void LoadIndividualPKM();//加载个体宝可梦库

void LoadMoves();//加载招式库

**//有关战斗结束时的处理**

void HandleWinEvent();//战斗胜利

void HandleLoseEvent();//战斗失败

void SendLoseMsg();//发送给服务端战败信息

**//与界面绘制有关**

void InitCanves();//初始化画布并装载图片资源

void SetFont();//设置字体

**//从服务器返回的消息构造相关数据结构**

void CreateUser(std::vector<std::string> recvReponse, User& user);//构造用户

void CreateEnemyPkm(std::vector<std::string> recvReponse);//构造敌方宝可梦

**//退出登录**

void LogOut();

**//点击事件**

void HandleCheckPokemon(int typeNum);//处理点击“查看宝可梦”的事件

void HandleCheckUser(std::string mode,int id);//处理点击“查看用户”的事件

**//关于登录/注册时的输入**

std::string InputName(int mode);//输入用户名

std::string InputPassword(int mode);//输入密码

bool IsNameValid(char userName[40]) const;//检查用户名是否合法

bool IsPasswordValid(char userPass[30]) const;//检查密码是否合法

**//图像**

bool IsBetween(int left, int right, int top, int bottom, int x, int y) const;

**//与服务器的连接**

bool ConnectHost(int port);

**（public）**

Client();//初始化客户端

~Client();

bool InitWSA();//初始化套接字库

**//需要根据当前状态的函数**

void HandleClickEvent();//处理用户点击界面事件

void RecvServerMsg();//根据当前所处的状态处理服务端返回的信息

void Draw();//绘制画面

**//自动模拟战斗**

void AutoBattle();//自动出招对战

**③Pokemon类：**

**变量**：**（均为protected）**

**（服务器端Pokemon类的变量客户端全有，这里只介绍客户端的类新增的变量）**

**//对战相关**

int currentHP; //对战时剩余的生命值

int accuracyLevel;//命中率等级

int evasionLevel; //闪避率等级，

bool isAtCriticalState;//是否处于易中要害状态，是的话可以提高暴击率

int criticalLeftTimes; //剩余可以处于易中要害状态的次数

clock\_t lastAttackTime;//上次出招的时间，用于计算间隔

std::string moveName; //某次出招的招式名称

int damage; //某次出招给对方带来的伤害

bool isHit; //是否击中对方

bool isCritical;//是否暴击

**方法：**

**（public）**

**//战斗相关**

void UpdateAttribute();//升级函数

int AutoMove() const;//随机出招,返回值为招式在招式库下标

virtual int Attack(Pokemon\* enemy);//攻击函数,计算给对方的伤害加成,并返回伤害值

void Damage(int damage);//小精灵受到攻击时的伤害计算

int UpdateExp(Pokemon\* enemyPokemon);//对战胜利后，更新经验值,并自动升级

void Reset();//战斗开始前重置战斗相关数值

int ComputeFinalDamage(double bonus, Move\* attackMove, Pokemon\* enemy) const;

double ComputeAccuracyRate() const;//计算并返回命中率

double ComputeEvasionRate(Pokemon\* enemy) const;//计算并返回闪避率

double ComputeCriticalRate(Move\* attackMove);//计算并返回暴击率

bool whetherHit(Pokemon\* enemy, int moveAccuracy) const;//判断对战时是否命中对方

int GetCurrentHP() const;//返回剩余血量

**//获得宝可梦信息**

Type GetType()const;//类别

int GetLevel() const;//等级

int GetIndividualID()const;//个体ID

int GetSpeiesID()const;//种族ID

std::string GetName() const;//名字

int GetSpeed() const;//速度

std::string GetPokemonInfo() const;//综合信息

**//界面相关**

void DrawPkmInfo() const;//绘制综合信息到界面上

void DrawPkmList(int curPage, int total, int curCheckSeq) const;//绘制背包列表

void DrawLose(int seq) const;//打印要失去的宝可梦

void DrawBattle(bool isAtFront) const;//打印战斗相关信息到界面

**③WaterPokemon类、FirePokemon类、GrassPokemon类、ElectricPokemon类：均为公有继承Pokemon类**

**变量**：**（均为private）**

Type advantageType; //本类别克制的类别

Type disadvantageType;//克制本类别的类别

**方法：**

**（public）**

int Attack(Pokemon\* enemy) override;//攻击函数,计算给对方的伤害,并返回伤害值

**④User类：**

**变量**：**（均为private）**

**（服务器端User类的变量客户端全有，这里只介绍客户端的类新增的变量）**

std::vector<WaterPokemon\*> waterPokemonBags;//水系宝可梦背包

std::vector<FirePokemon\*> firePokemonBags;//火系宝可梦背包

std::vector<GrassPokemon\*> grassPokemonBags;//草系宝可梦背包

std::vector<ElectricPokemon\*> electricPokemonBags;//电系宝可梦背包

**方法：**

**（public）**

void SetUser(std::vector<std::string> recvReponse);//根据服务端传来的数据设置用户的相关信息

void SetIsOnline(bool isonline);//设置用户在线情况

void ClearBags();//清空所有宝可梦背包（共五个）

void AddPokemonToBags(Pokemon\* newPokemon);//把宝可梦添加到总背包和类别背包里

void LosePokemon(int individualID);//删除总背包和类别背包中战败的失去的宝可梦

void UpdateFullLevel();//更新拥有的满级宝可梦数

void SetBattleTimes(bool isWin);//更新胜利/失败次数

void SetPkmSum(int sum);//设置用户拥有的宝可梦的总数

double GetWinRate() const; //获取该用户的胜率

int HasPokemon(Type type) const;//返回某背包中的宝可梦数

int GetID() const;//获取用户的ID

Pokemon\* GetCurPkm(Type type, int seq) const;//获取当前查看的宝可梦指针

int GetPkmSum() const;//获取用户拥有的宝可梦数

void DrawUserInfo(int localID, int state) const;//打印用户信息到界面上

void DrawUserList(int curPage, int total) const;//打印用户列表到界面

int RandPickLosePkm(int first, int second) const;//随机选取要失去的宝可梦

**⑤Move类：**

**变量**：**（均为private）**

int id;//标号

int accuracy;//招式的命中（不是命中率）

int power;//招式的威力

Type moveType;//招式的类别，招式的类别若与小精灵一致，则可以增大伤害

bool isCriticalHit; //是否是容易产生暴击的招式

bool isPhysical;//是否是物理攻击，true是物理攻击，false是特殊攻击

std::string moveName;//招式名字

**方法：**

**（public）**

Move(int id, int accuracy, int power, bool isCri, bool isPhy, std::string type, std::string name);

int GetAccuracy() const;//返回招式的命

Type GetType() const;//返回招式的属性

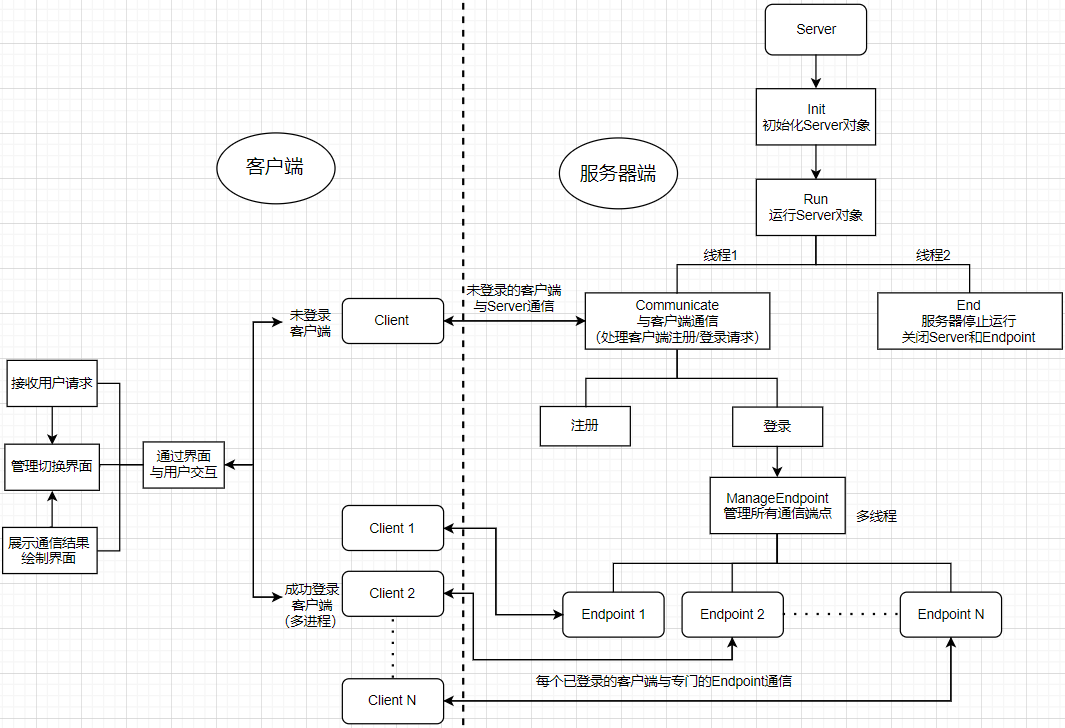
int GetPower() const;//返回招式的威力

bool GetIsPhysical()const;//返回招式的攻击类型

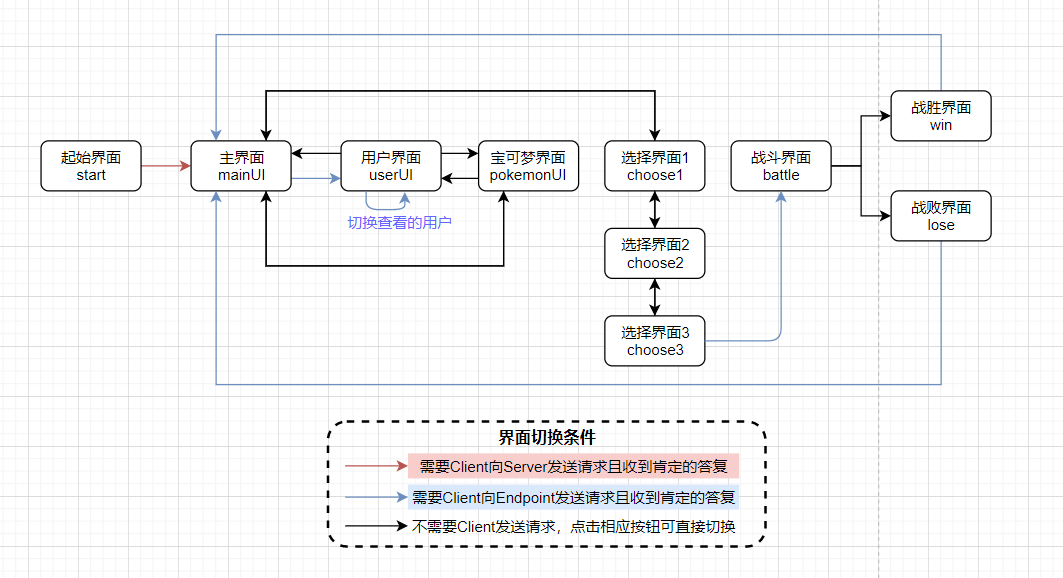
std::string GetName() const;//返回招式名

四、模块设计与关系

客户端和服务器端的模块设计与关系图如下：



客户端的界面之间的切换关系如下图：



五、使用说明书

一、打开服务器程序（只需打开一个）。

二、打开客户端程序即可开始游戏体验（可以打开多个客户端进行多用户访问）。

**游戏操作方法：**



（注册/登陆界面）

**（1）注册：**

点击界面上的“注册”按钮，程序会弹出文本框，按文本框上的提示信息输入用户名和密码即可。若注册成功，可以在界面上看到“注册成功”的提示字样；若失败，系统会提示用户重新操作。

**（2）登录：**

点击界面上的用户名文本框和密码文本框，在弹出的框中输入用户名和密码即可。若登陆成功，用户将进入游戏主界面，可以开始游戏；若失败，系统会提示用户重新操作。



（游戏主界面：显示本机用户的信息）

进入游戏主界面后，可以在界面上看到用户自己的信息，包括：用户名、拥有宝可梦数量、拥有高级宝可梦（满级宝可梦）数、获胜概率（获胜次数/战斗总数）、获得勋章、宝可梦背包。

**（3）查看宝可梦：**

点击界面上的类别“宝可梦背包”（水系/火系/草系/电系），如果用户拥有这个类别的宝可梦，即该类别背包的宝可梦数量不为0，即可进入宝可梦界面查看相应类别的宝可梦。例：想查看电系宝可梦，点击电系“宝可梦背包”即可。

进入宝可梦界面后，可以看到宝可梦的详细信息，包括：宝可梦个体ID、宝可梦名、等级、经验值、距离下一级的经验、类别和主属性、6项属性值、招式列表。

点击左侧的“宝可梦表”可切换当前查看的宝可梦。



（宝可梦界面：本例中为电飞鼠的介绍）

**（4）查看其他用户：**

点击界面上的“查看用户”按钮，即可进入查看用户界面，点击左侧的用户列表，可以切换当前查看的用户。点击宝可梦背包，可以进入宝可梦界面查看当前正在查看的用户拥有的宝可梦。



（两个用户同时登录，右侧为查看用户界面）

**（5）战斗：**

在主界面上，根据自己的意愿，点击“升级赛”（战胜只获得经验）或“决斗赛”（战胜获得经验和挑战的宝可梦，战败失去一只背包中的宝可梦）即可进入选择界面第一阶段。（选择界面共有三个阶段）

阶段一：选择本方出战的宝可梦

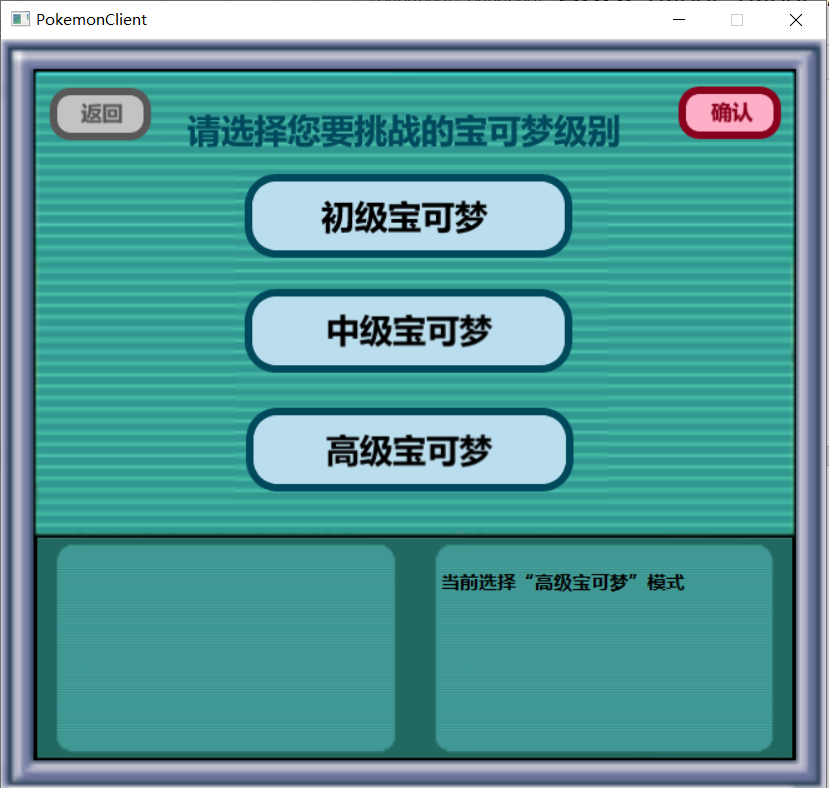
点击宝可梦列表中的序号可切换当前选择出战的宝可梦。点击“确认”按钮进入下一阶段。点击“取消”按钮放弃战斗。



（战斗前的选择阶段一：选择本方出战宝可梦）

阶段二：选择要挑战的模式

点击界面中的模式进行对挑战模式的选择。点击“确认”按钮进入下一阶段。点击“返回”按钮，返回到阶段一。



（战斗前的选择阶段二：选择要挑战的模式）

阶段三：选择要挑战的宝可梦

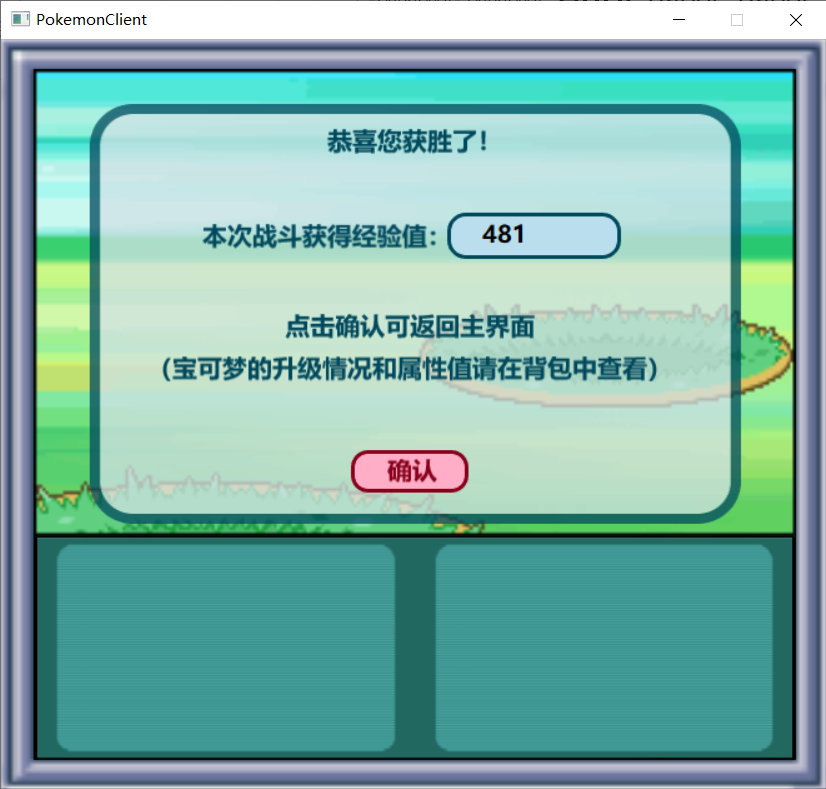
点击界面中的宝可梦来挑选要挑战的宝可梦。点击“确认”按钮结束选择阶段，可进入战斗界面，正式进入战斗（若界面中出现加载失败图标，请重新进行操作）。点击“返回”按钮，返回到阶段二。



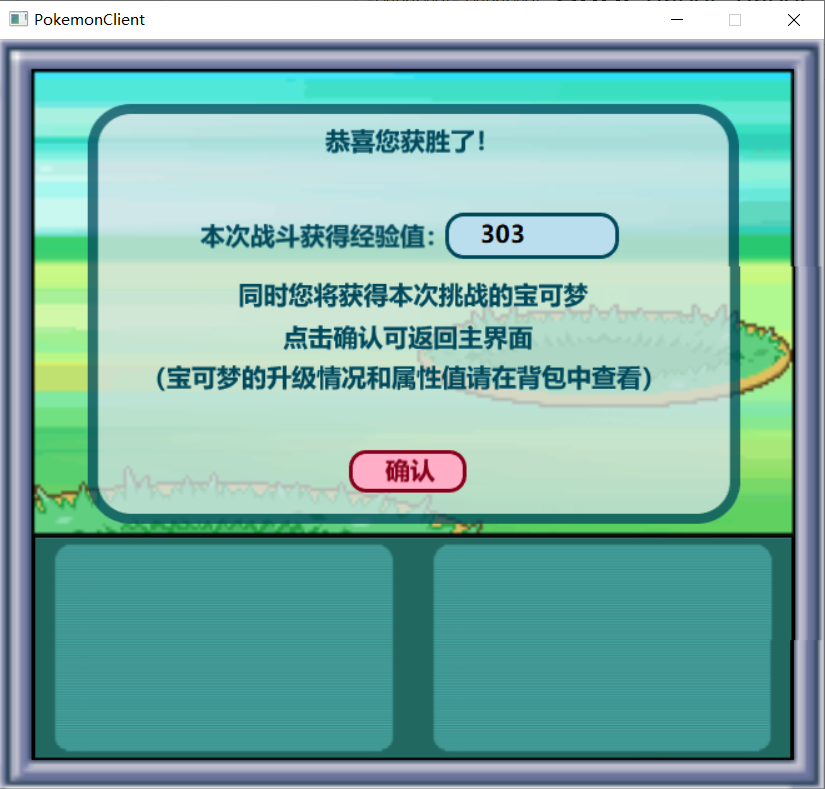
（战斗前的选择阶段三：选择要挑战的宝可梦）



（战斗界面）



（升级赛战斗结果展示：左侧战斗胜利，右侧战斗失败）



（决斗赛战斗结果展示：战斗胜利）



（决斗赛战斗结果展示：战斗失败）

若决斗赛战斗失败，系统自动随机从用户背包中挑选三只宝可梦，用户需从中挑选一只宝可梦失去。

战斗结束后，点击“确认”按钮即可返回主界面。