“C++程序设计与训练”课程大作业
项目报告

项目名称：“雷课堂”（Thunder Class）

开发环境	QT
环境版本	5.14.2
编译器	MinGW32

姓名： 陈文泽

学号： **2019011450**

班级： 自 **95**

日期： **2020-06-21**

一、 亮点工作

我认为我的雷课堂最大亮点在收发问题的功能实现与问题类的设计。首先在类设计部份我以 Quiz 类派生出 StudentQuiz 与 TeacherQuiz 两个类，分别处理单个学生答题记录与整个课堂的答题状况统计。在类接口设计部份，我把接口的数量降到最低，且每个接口都有自己的物理意义，如回答问题、读考卷、改考卷等。此外，在设计界面时，我巧妙利用界面的功能，将用户输入错误的可能性大幅降低，以避免不正规值造成的程序崩溃。

二、 需求分析

2.1 用户登录：

a) 登入: 使用登录界面(loginWindow)，调用登录业务流程类(loginProcess)，从文件载入全部用户信息(User)，根据用户名和密码对比用户是否存在(User)，如存在(loginProcess)判断用户类型(User)，跳转到相应的界面(loginWindow)，如不存在则更新登入失败次数(loginProcess)并跳出提醒视窗(loginWindow)，若三次登入失败(loginProcess)则强制登出(loginWindow)。

b) 账号增删改: 使用者输入账号密码类型并按下新增或删除按钮(adminWindow)，调用新增/删除帐户函数(User)，该函数判断类型是否正确、是否已存在同名用户后在静态成员变量 m_UserList 中新增/删除用户并更新文件(User)，判断成功或失败(adminProcess)并显示在屏幕(adminWindow)

2.2 语音设备选择和切换：自动选择默认设备。

2.3 共享屏幕:

a) 教师:用户按下开始共享(teacherWindow)，创建屏幕共享线程每隔 40 毫秒调用截图函数(teacherProcess)，抓取屏幕截图(Image 友元函数，存成 image 对象)，教师业务流程类(teacherProcess)调用函数将 image 转换为 JPG 格式后封装成 Msg(Message)，教师业务流程类(teacherProcess)将其发送全班(Server)

b) 学生:收到屏幕共享 Msg (studentProcess)，转换为 JPG 格式 (Message)，学生业务流程类将其转换为 QImage 格式(studentProcess)，更新画面(studentWindow)

2.4 语音直播

a) 教师:用户按下开始语音直播(teacherWindow)，创建 audio 对象(teacherWindow)并创建

录音线程、开始录音、每次录音持续一秒、得到的数据以先进先出的形式存放在 m_RecordedFrames 中(audio)，创建语音共享线程持续将队列头的数据取出(teacherProcess)，将数据封装成 Msg(Message)，发送全班(Server)

b) 学生: 进入课堂时创建播放语音线程与记录带播放语音的 audio 对象(studetnProcess)，该线程持续从该 audio 对象的队列头取出数据(studetnProcess)，将该数据播放出来(audio)，下课时关闭线程(studentProcess)

c) 学生: 收到语音直播 Msg(studentProcess)，专换为语音类(Message)，插入 audio 对象的队列尾(studentProcess)

2.5 随机语音/文字提问

a) 教师一 用户按下随机提问按钮(teacherwindow)，业务流程类产生提示字串”您已被选中回答随机提问”(teacherProcess)，封装为随机提问 Msg 对象(Message)，调用教师业务流程(teacherProcess)，发送给随机对象(Server)

b) 学生 收到随机提问(studentProcess)，将提示做答文字显示在屏幕上(studentWindow)，将成员变量 isAsked 改为是(studentProcess)，每次发送文字前判断成员变量 isAsked 的值(studentProcess)，若为是(studentProcess)则封装为随机提问 Msg (Message)，若为否 (studentProcess) 则封装为一般文字 Msg (Message)，调用学生业务流程(studentProcess)，发送老师(Client)

c) 教师二 收到随机提问 Msg(teacherProcess)，显示在屏幕上(teacherWindow)，封装为一般文字 Msg(Message)，调用教师业务流程(teacherProcess)，发送全班(Server)

d) 细节:

i. 将发送给随机对象设计在 Server 类内而非在 teacherProcess 内处理掉是考虑到此功能也是 Server 能够拥有的功能。如此设计增加核心类的通用性

ii. 如此设计的出发点是将随机提问视为纯文本消息的一个特例 能够有效利用已有的文本传输机制

2.6 在线发题

a) 点击发题按钮(teacherWindow)，显示题目设计视窗(makeQuizWindow)，设计完成后按下送出题目(makeQuizWindow)，调用教师业务流程接收问题描述、选项描述、正确答案(teacherProcess)，生成学生类问题对象记录单个学生作答情况(StudentQuiz)，

将学生类问题对象封装为 Msg (Message)，发送全班(Server)，同时生成教师问题对象记录全班答题情况(TeacherQuiz)，将教师问题对象存储在业务流程类(teacherProcess)

b) 点击收题目按钮(teacherWindow)，调用教师业务流程(teacherProcess)，产生提前收题 Msg(Message)，发送全班(Server)

c) 点击显示结果按钮(teacherWindow)，弹出显示结果视窗(resultWindow)。

d) 接收到学生回答(teacherProcess)，解码为学生问题对象(Message)，通过教师业务流程调用函数(teacherProcess)以更新该教师问题对象(TeacherQuiz)，产生统计结果字串(TeacherQuiz)，更新显示结果视窗(resultWindow)

e) 设计细节

i. 学生问题类(studentQuiz)如同空白考卷，在教师端创建时填上问题、选项、正确答案，在学生端则调用 Ans 函数填上姓名、我的回答

ii. 产生统计结果字串(TeacherQuiz::getStr)将教师问题对象转换为字串类型传出，共用于两处，下课时的文件输出与显示答题统计信息界面

2.7 在线答题

a) 收到问题 Msg(studentProcess)，解码为学生问题对象(Message)，弹出并重新设置回答视窗界面(ansQuiz)，选择答案后送出(ansQuiz)，调用学生业务流程对象接收答案(studentProcess)并调用 Ans 函数更新学生问题对象(StudentQuiz)，封装为 Msg(Message)，发送老师(Client)

b) 收到提前收题 Msg(studentProcess)，操控界面按下送出问题并关闭视窗(ansQuiz)

c) 设计细节: 此处不使用类似 TeacherQuiz::getStr 设计界面是为了增加界面与用户的互动，让界面清楚区分各选项与问题的好处是能够 setEnable(false)部份控件，让程序更加鲁棒，TeacherQuiz 可以那样设计的原因是它不需要与使用者互动，因此考虑方便性就直接输出字串

2.8 学生签到

a) 学生送出包含用户账号密码的使用者登入 Msg(studentProcess)，教师端接收 Msg 并解码 Msg(teacherProcess + Message)，在用户列表中搜寻用户(User)，将搜寻到的用户账号密码分别记录在成员变量中(teacehrProcess)

b) 学生退出课堂时发出信号(studentProcess) ,教师端接收 Msg 并解码 Msg(teacherProcess + Message)，在成员变量中搜寻该用户信息并删除之(teacherProcess)

2.9 注意力

a) 学生登入课堂时创建计时器与记录学生表现成员变量(studentProcess)，定时调用函数(StudentProcess)更新学生表现对象的注意力数据(Perfromance)

b) 在学生下课前送出学生表现对象(studentProcess + Message + Client) ,在教师端将学生表现对象存储下来(teacherProcess)

c) 设计细节:

i. teacherProcess 中记录学生姓名密码的成员变量可以让老师在课堂中实时获得已经签到的同学。但无法处理重复签到问题。

ii. Performance 类则是同时处理了签到与注意力两个功能。因为每个学生每次退出课堂时都会送出学生表现(包含注意力与在线时间)数据，因此能够处理重复签到问题。但这样设计的缺点是我们只能在课堂结束后得到数据，因此要辅以 i 中变量来实现完整功能

2.10 上课/下课

a) 上课: 用户点击上课按钮(teacherWindow)，开启服务器(Server)，等待接收用户登入 Msg(teacherProcess)，若为异常登入将其下线(teacherProcess)，若为正常登入则创建 connection 并将学生信息记录在成员变量中(teacherProcess)

b) 下课: 用户点击下课按钮(teacherWindow)，调用教师业务流程(teacherProcess)，创建下课 Msg (Message)，发送全班(Server)，等待学生回传 Performance 对象(studentProcess)，解码收到学生表现 Msg (Message)，将接收到的 Performance 记录在成员变量 m_Perfor 中(teacherProcess) ,生成欲输出的字符串包含注意力与各题统计结果(Performance + teacherQuiz)，调用函数将其输出成文件与显示在画面(teacherProcess)

c) 设计细节:

i. teacherQuiz 与 Performance 都有 getStr()函数，将该类对象内存储的信息已指定字串形式输出，这样的好处是在处理部份不需要与使用这互动的界面时(如: 文本输出)可以直接调用。

ii. 教师强制下课后必须等待一小段时间以获取学生回传的信息

2.11 进入课堂/退出课堂

a) 进入课堂: 用户点击进入课堂按钮(StudentWindow)，尝试登入若超过 30 秒未响应则强制登出并提示(StudentProcess)，登入后送出帐号密码以供教师端判断是否为合法登入(StudentProcess)，创建消息处理线程(StudentProcess)

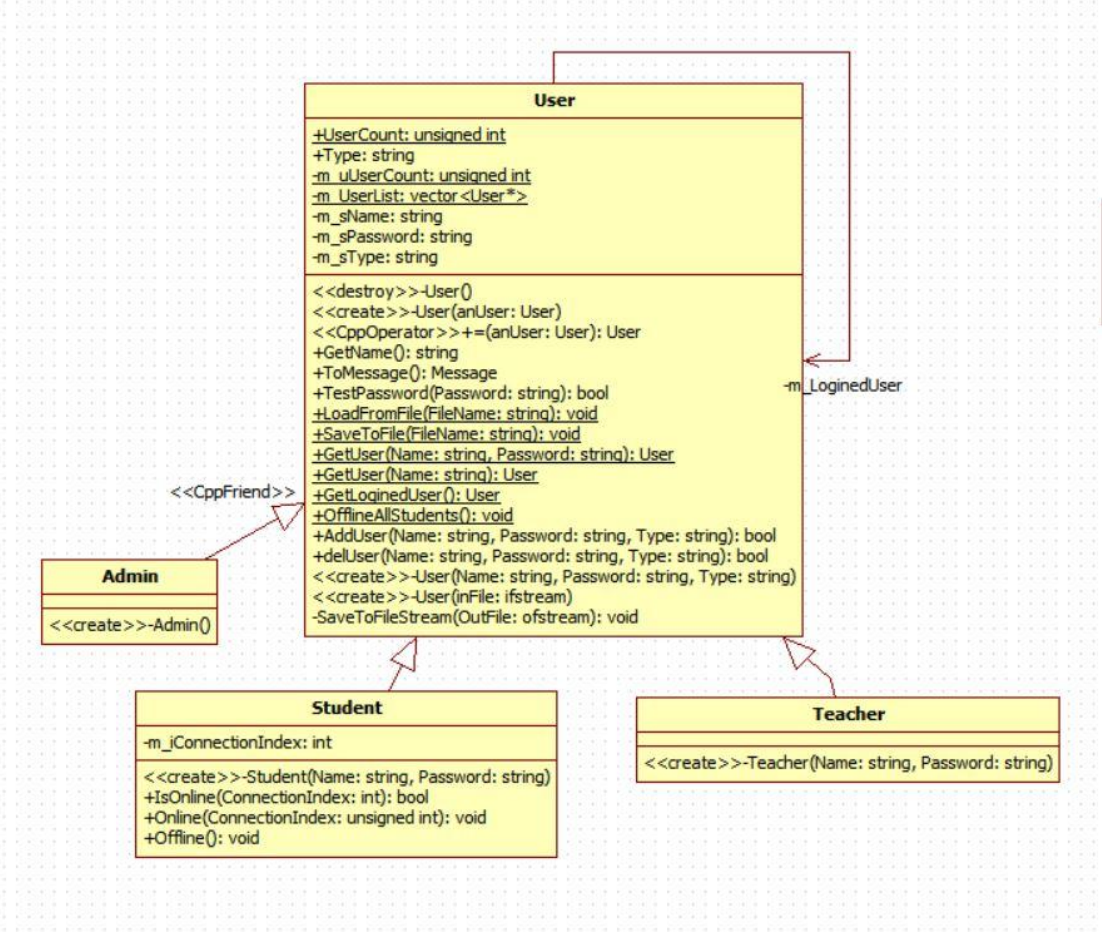
b) 主动下课: 将学生表现对象转为 Msg (Message)，发送老师(Client)，产生下课 Msg(Message)，发送老师(client)

c) 教师下课: 接收到下课 Msg(StudentProcess)，后续操作同主动下课

d) 设计细节:

i. 接收到强制下课的信号后以主动下课的方式处理，能够有效增加程序的方便性与可读性，而代价只是多发送一次退出课堂信号

三、 类与类结构设计



1. User(核心类)

(1) 概述: 用户类用于处理使用者登入、使用者增删改的相关事宜，派生出Admin,Teacher,Student 等三个类。

(2) 功能介绍(仅挑选部份函数与成员介绍)

+SaveToFile(FileName: string): void 将全部用户信息存入文件

+LoadFromFile(FileName: string): void 从文件读入一批用户

以上两个函数提供更新用户信息到文件的接口

+TestPassword(Password: string): bool 用于测试给定密码是否为用户密码

+GetUser(Name: string, Password: string): User 静态函数按用户名密码搜索用户并返回用户指针

+GetUser(Name: string): User 静态函数按用户名搜索用户并返回用户指针

+GetLoginedUser(): User 获得最后一次登录的用户指针

以上四个函数提供查找用户接口，分别针对不同情况下的查找方式

+AddUser(Name: string, Password: string, Type: string): bool 增加用户

+delUser(Name: string, Password: string, Type: string): bool 删除用户

以上两个函数为增删用户的接口

+GetName(): string 提供只读的用户名的接口

+Type: string

以上两个函数为获取用户信息接口

(3) 设计思路:

① User 类必须提供遍历查询用户接口、得到用户信息接口、增删改用户列表接口，基于上述考量，封装成如上的形式

② 特别的是，将用户列表设为静态成员，如此好处为当程序的某处更新后，其他地方也能同时更新

③ 仅提供姓名与种类的查询，考量安全性不提供密码查询接口

④ 基于安全性的考量，部份操作只能由指定类型用户完成，因此派生出三类Admin、Teacher、Student，以便于区分权限

2. Admin(核心类):

(1) 概述: 派生自 User 类，管理员类，此类对象能够操作账户的增删改

(2) 功能介绍(仅挑选部份函数与成员介绍)

+AddUser(Name: string, Password: string, Type: string): bool 增加用户

+delUser(Name: string, Password: string, Type: string): bool 删除用户

此函数重载了基类函数的 AddUser 函数，目的是确保只有 Admin 类对象能操作账户增删改

(3) 设计思路:

① 确保增修改受到权限限制，增加如上的函数重载，显著提升安全性

3. Teacher(核心类): 派生自 User 类，教师类，因为有 Admin 跟 Student 类，确保设计的完整性，因此有此类

4. Student(核心类):

(1) 概述: 派生自 User 类，学生类，用于教师端统计学生是否在线情形

(2) 功能介绍(仅挑选部份函数与成员介绍)

<<create>>-Student(Name: string, Password: string)

-m_iConnectionIndex: int 用户端口号

+IsOnline(ConnectionIndex: int): bool 返回是否学生在线

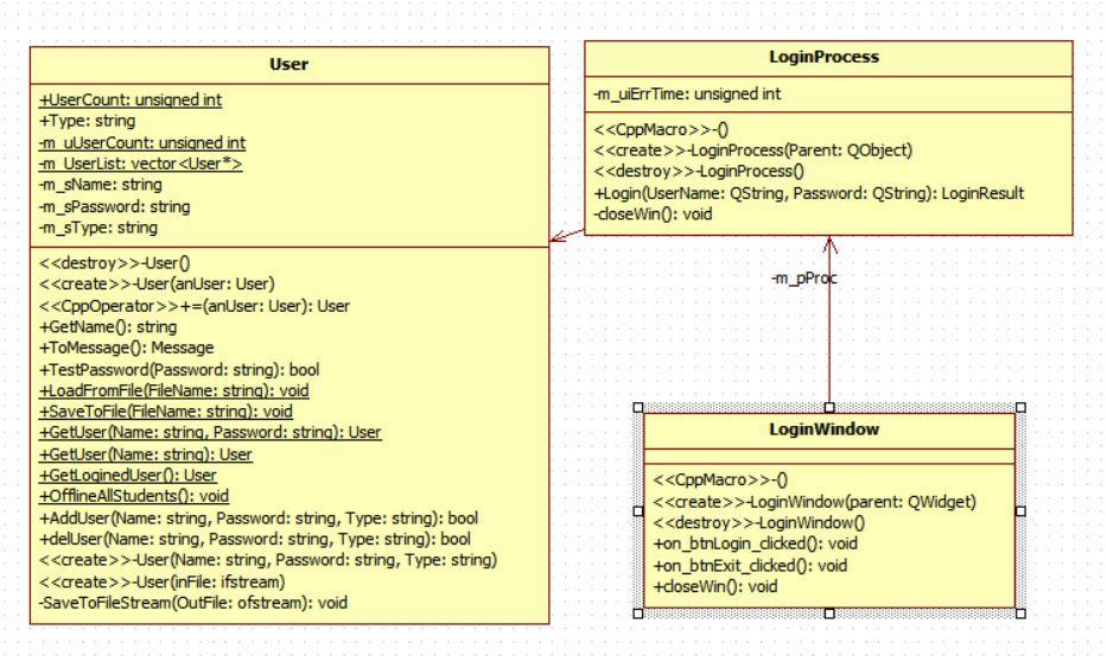
+Online(ConnectionIndex: unsigned int): void 设定学生在线，并使从 Server 的 ConnectionIndex 号连接收发数据

+Offline(): void 设定学生离线

此类接口设计以提供查询与修改用户是否在线为主

(3) 设计思路

① 教师端有用户列表，因此设想能以此为基础记录学生在线情形。因此在学生类设计各种接口，以得到/修改学生在线情形



5. LoginProcess(业务流程类):

(1) 概述: 登入业务流程类，处理登入相关事宜

(2) 功能介绍(仅挑选部份函数与成员介绍)

<<create>>-LoginProcess(Parent: QObject) 构造函数

<<destroy>>-LoginProcess() 析构函数

+Login(UserName: QString, Password: QString): LoginResult 登入函数(回传登入者类型)

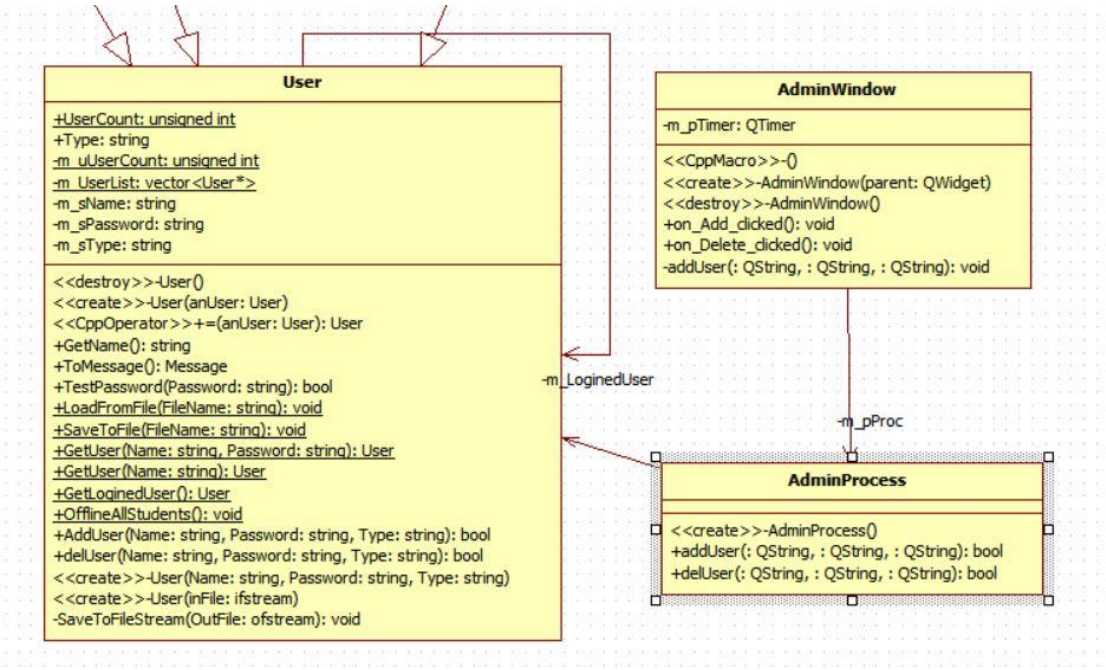
以上一个函数处理用户登入相关事宜

-closeWin(): void 请求强制跳出程序

-m_uiErrTime: unsigned int 登入失败次数

以上两个函数处理: 若用户登入超过三次强制退出

(3) 设计思路:仅对外提供等入接口，其他机制与数据都封装在类内



6. AdminProcess(业务流程类)

(1) 概述: 管理员界面业务流程类，负责处理账户增删改等功能

(2) 功能介绍(仅挑选部份函数与成员介绍)

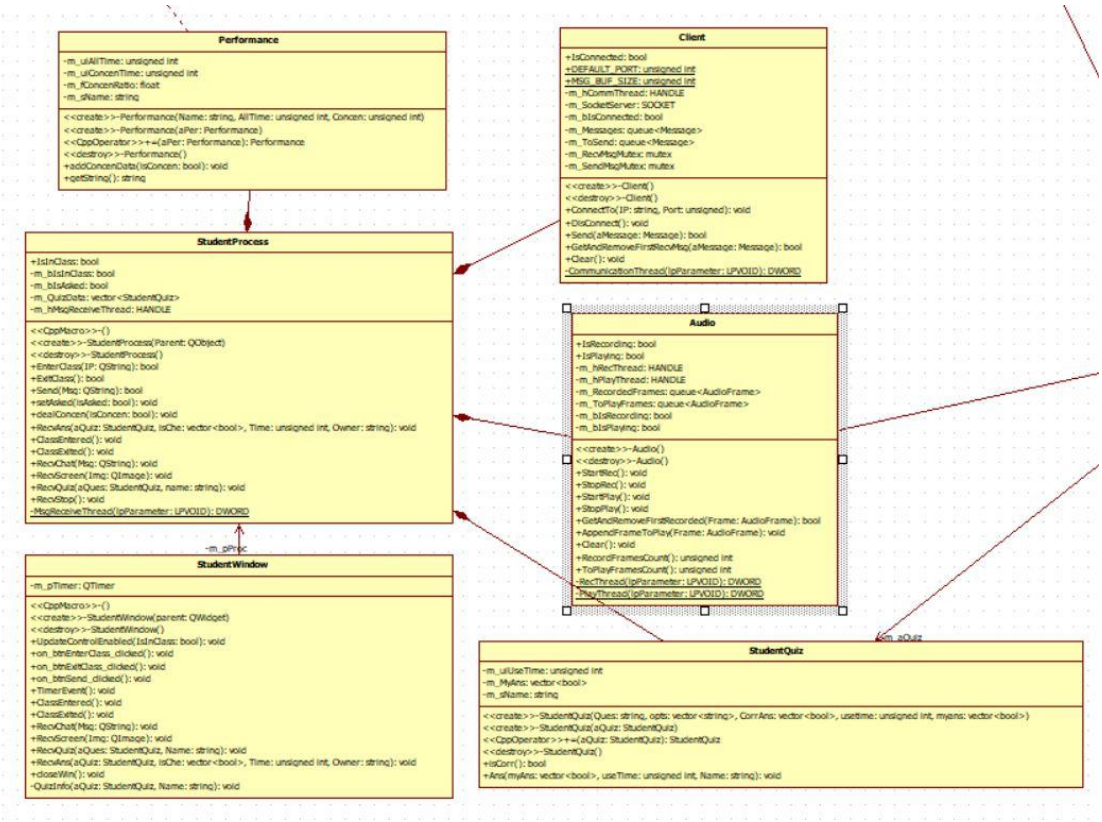
<<create>>-AdminProcess()构造函数

+AddUser(Name: string, Password: string, Type: string): bool 新增用户

+DeleteUser(Name: string): bool 删除用户

(3) 设计思路:

① 需要提供两个功能，因此设计两个接口，分别对应 User 类函数



7. StudentProcess(业务流程类):

(1) 概述: 学生业务流程类，处理所有与学生相关的功能

(2) 功能介绍(仅挑选部份函数与成员介绍):

+IsInClass: bool 常引用数据成员 IsInClass 表示是否在上课

+EnterClass(IP: QString): bool 进入课堂

+ExitClass(): bool 退出课堂

+ClassEntered(): void 进入课堂信号

+ClassExited(): void 退出课堂信号

+RecvChat(Msg: QString): void 收到文字消息信号

+RecvScreen(Img: QImage): void 收到屏幕共享消息信号

+RecvQuiz(aQues: StudentQuiz, name: string): void 收到题目

以上函数目的为更新界面，以下函数分别对应到一个需要实现的功能

-m_bIsAsked: bool 是否处于被随机抽考模式

+RecvStop(): void 提前收题

-m_QuizData: vector<StudentQuiz> 答題紀錄

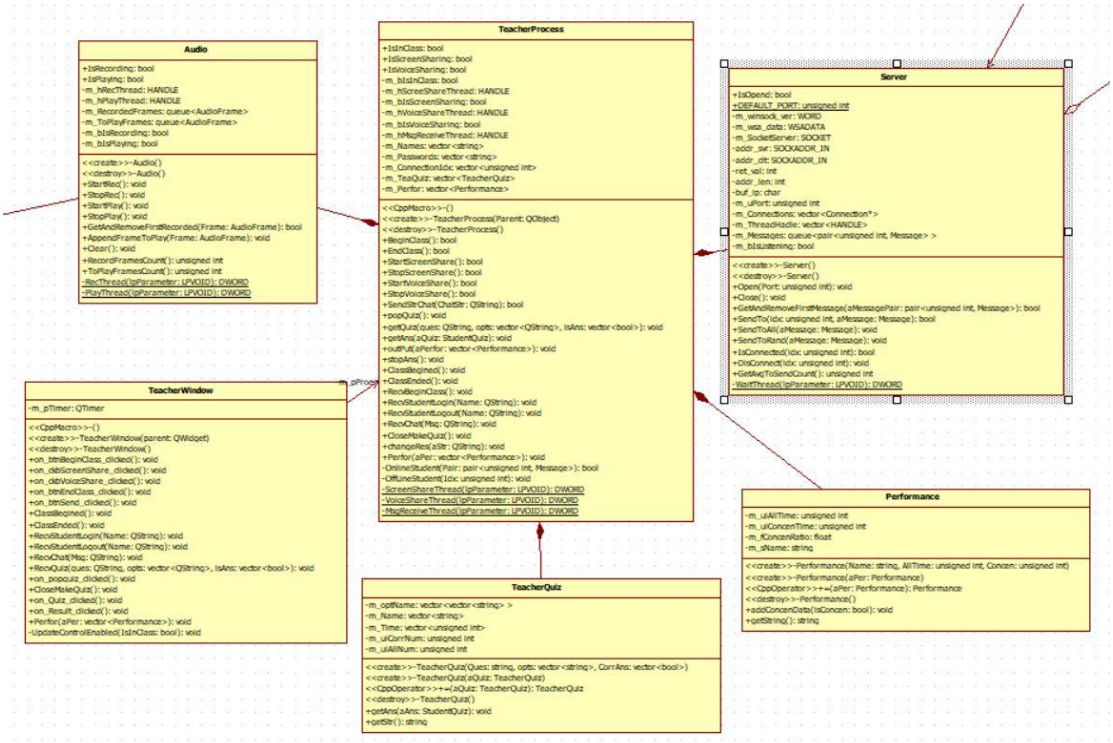
+Send(Msg: QString): bool 发送文本消息给教师

+setAsked(isAsked: bool): void 设为被提问状态

+dealConcen(isConcen: bool): void 处理专心度问题

+RecvAns(aQuiz: StudentQuiz, isChe: vector<bool>, Time: unsigned int, Owner: string): void 收到回答

(3) 设计思路: 每个功能对应到一个函数，若该功能牵涉到更新界面，则需要在设置额外的函数



8. TeacherProcess(业务流程类):

(1) 概述: 教师业务流程类，用来处理各种教师功能

(2) 功能介绍(仅挑选部份功能介绍)

+BeginClass(): bool 上课

+EndClass(): bool 下课

+StartScreenShare(): bool 开始屏幕共享

+StopScreenShare(): bool 停止屏幕共享

+StartVoiceShare(): bool 开始语音直播

+StopVoiceShare(): bool 停止语音直播

+SendStrChat(ChatStr: QString): bool 群发文本信息

+popQuiz(): void 向随机对象发送随机提问提示

+getQuiz(ques: QString, opts: vector<QString>, isAns: vector<bool>): void 得到新的

问题

+getAns(aQuiz: StudentQuiz): void 得到答案

+outPut(aPerfor: vector<Performance>): void 输出学生表现信息

+stopAns(): void 收题目

以上每个函数分别对应一个功能

+ClassBegined(): void

+ClassEnded(): void

+RecvBeginClass(): void

+RecvStudentLogin(Name: QString): void

+RecvStudentLogout(Name: QString): void

+RecvChat(Msg: QString): void

+CloseMakeQuiz(): void

+changeRes(aStr: QString): void

+Perfor(aPer: vector<Performance>): void

以上函数分别对应一个功能，但目的是为了更新使用者界面

-OnlineStudent(Pair: pair<unsigned int, Message>): bool

-OffLineStudent(Idx: unsigned int): void

以上函数处理学生登入问题

-ScreenShareThread(lpParameter: LPVOID): DWORD

-VoiceShareThread(lpParameter: LPVOID): DWORD

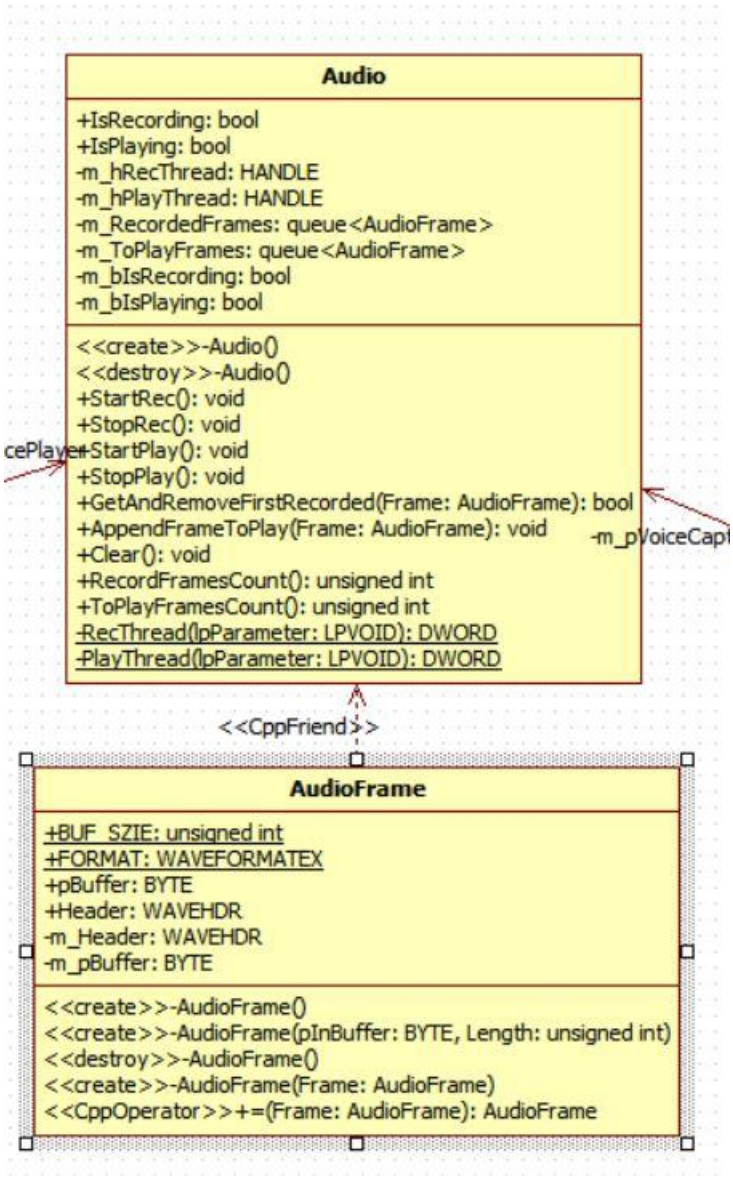
-MsgReceiveThread(lpParameter: LPVOID): DWORD

以上线程为处理各种信号的收发

(3) 设计思路:

① 业务流程类设计思路: 一个功能对应一个函数，若牵涉到界面则对应两个函数

② “has a”Server/Audio/TeacherQuiz/Performance/Connection，以上类对象都是教师业务流程类的成员变量，通过更改这些变量，可以达到记录的效果。



9. Audioframe(核心类)

(1) 概述: 音频帧类，等同于一个音频帧对象，主要功能用来构建与记录录下来的音频帧，处理语音直播的相关事宜

(2) 功能简介(仅挑选部份功能)

<<create>>-AudioFrame(pInBuffer: BYTE, Length: unsigned int) 构造特定音频数据的音频帧

(3) 设计思路

① 设定 Audio 为其友元对象，因为 Audio 类会频繁调用它的数据，且其余类并不需要调用此类数据，因此若设计多个接口函数提供数据的读取，反而会增加安全的隐忧

② 此类代表一个音频物件，因此接口的设计主要有大小、类型等

10. Audio(核心类)

(1) 概述: 音频类，等同于一个音频队列，其功能包含构建、记录与播放音频

(2) 功能介绍(仅挑选部份介绍

+IsRecording: bool 常引用 IsRecording 表示是否正在录音

-m_RecordedFrames: queue<AudioFrame> 已录制音频队列

-m_hRecThread: HANDLE 录音线程句柄

+StartRec(): void 函数 StartRec 开始录音

+StopRec(): void 函数 StopRec 停止录音

+StartPlay(): void 函数 StartPlay 开始播放

+GetAndRemoveFirstRecorded(Frame: AudioFrame): bool 函数 GetAndRemoveFirstRecorded 获取并删除最早录制的一段音频

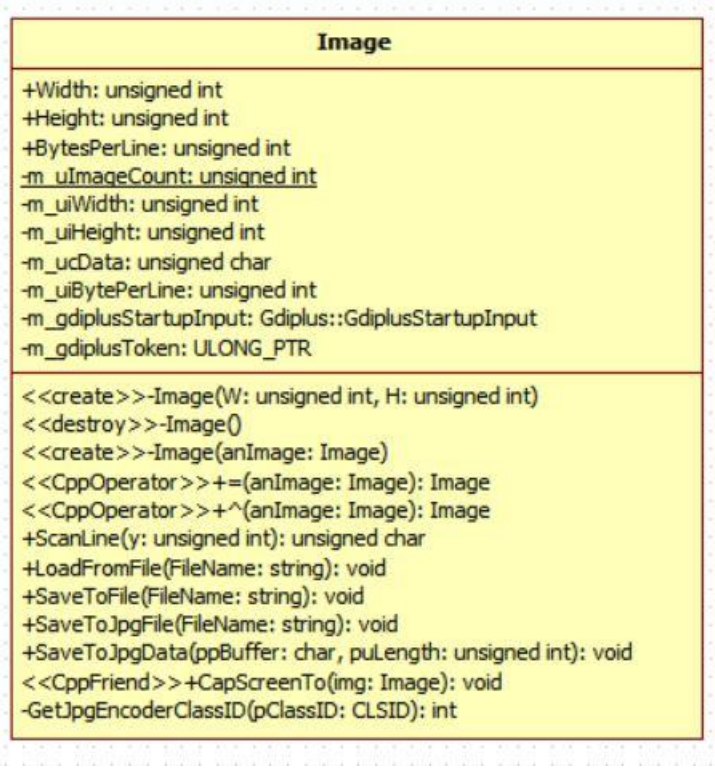
+RecordFramesCount(): unsigned int 函数 RecordFramesCount 已录制音频队列里的音频数量(单位为秒，整数)

以上函数处理录音功能(播放功能基本相同，此处省去): 按下开始录音，将状态改为正在录音。若正在录音，录音音频每个固定时间将 AudioFrame 对象加到录制音频列表，拥有接口函数可以获取列表的头

(3) 设计思路

① 在设计时主要有三个方向，第一，Audio 类身为一个音频类，必须要有基本信息，如音频长度等，以供记录数据的完整性。第二，必须要设计接口函数提供外界合法读取数据，此处共友三个接口，分别是开始录音(播放)、获取音频队列头、获取音频对列长度，分别对应到的是共享视频时需要处理的三个问题，何时开始?如何得到数据?如何避免队列塞车?

② 为了避免信号线拥塞导致程序崩溃，设计接口函数，得到音频队列长度，若以经过长则停止录音(播放)



11. Image(核心类)

(1) 概述: 图片类，表示一个图片对象，主要用来处理屏幕共享功能

(2) 功能介绍(仅挑选部份功能介绍)

+Width: unsigned int 常引用数据成员 Width 表示以像素为单位的图像宽度

+Height: unsigned int 常引用数据成员 Height 表示以像素为单位的高度

+BytesPerLine: unsigned int 常引用数据成员 BytesPerLine 表示每行图像占了多少字节（4 字节整数倍）

-m_uiWidth: unsigned int 图像宽度

-m_uiHeight: unsigned int 图像高度

-m_ucData: unsigned char 用一维数组模拟二维存储结构

-m_uiBytePerLine: unsigned int 每行像素占用多少字节（4 字节整数倍）

以上记录图片相关基础信息

<<CppOperator>>+^(anImage: Image): Image 异或运算符函数

+SaveToJpgFile(FileName: string): void 保存图像到 Jpg 格式图像文件

+SaveToJpgData(ppBuffer: char, puLength: unsigned int): void 保存图像到 Jpg 原始数据到指定内存区域

以上函数用来处理时间空间冗余

+ScanLine(y: unsigned int): unsigned char 返回指定行的原始数据区指针

+LoadFromFile(FileName: string): void 读取本类自定义的图像格式文件

+SaveToFile(FileName: string): void 保存图像到本类自定义的图像格式文件

以上函数为读取图片数据的接口函数

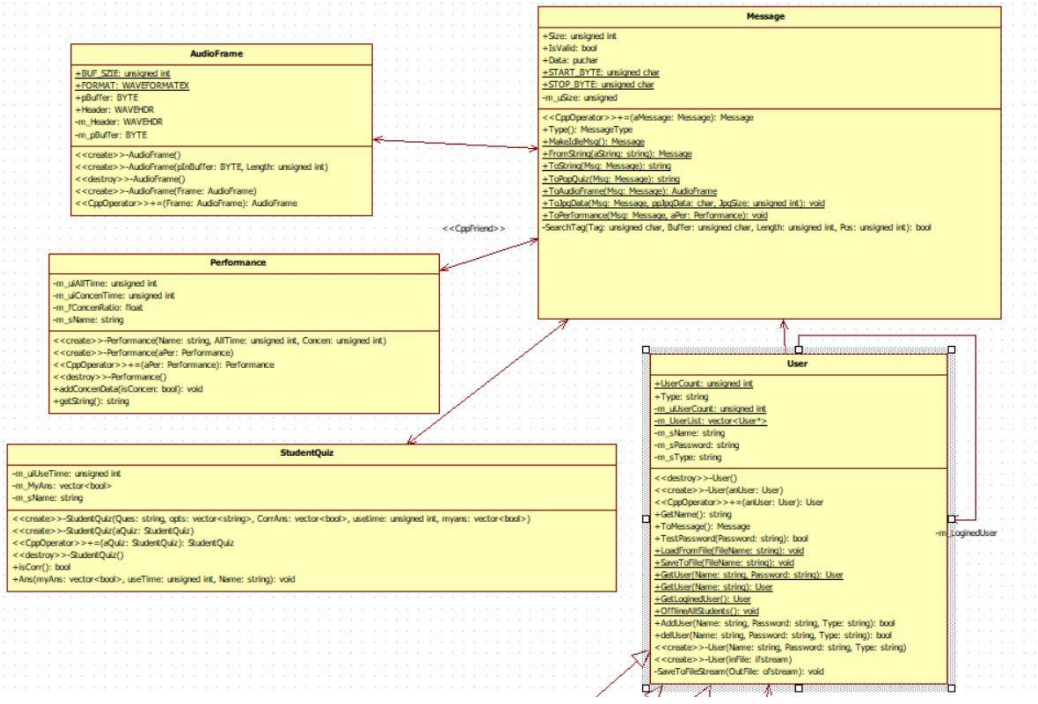
<<CppFriend>>+CapScreenTo(img: Image): void 友元函数 抓取当前屏幕到指定图像对象

(3) 设计思路

① 设计时分为四个部份。第一，image 类作为一个图片对象，必须要有基本的信息，如长宽高与图片数据等。第二，必须要有接口函数提供外界获取此图像。第三，因为此图像的目的是用于视频播放，因此应该设计部份用于处理时间空间冗余的函数。最后，必须要有抓取屏幕截图的函数

② 将抓取屏幕截图函数设为友元函数的原因是 :此函数不应该是图片类的一个功能(一张图片不会自己去劫取屏幕)，因此基于此方面考量设为友元

③ 屏幕保存的方式，是以 unsigned char 形态逐行保存，并以一维数组的方式存放在成员变量中



12. Message(核心类):

(1) 概述: 消息类，是网络传输的基本单位

(2) 功能介绍(仅挑选部份功能):

+static Message MakeExitMsg(); 构造退出课堂消息

+MakeIdleMsg(): Message 构造空消息

+FromString(aString: string): Message 构造文字通信消息

+FromAudio(aFrame: AudioFrame): Message 构造共享语音消息

+FromImageJpg(anImage: Image): Message 构造共享屏幕消息

+FromPopQuiz(aString: string): Message 构造随机提问消息

+FromPerformance(aPer: Performace): Message 构造学生表现消息

+FromQuiz(aQuiz: StudentQuiz): Message 构造发提消息

+MakeStopAns():void 构造强制收题消息

以上函数皆为静态成员函数，用来建构消息

+ToAnswerString(Msg: Message): string 还原文字消息

+ToAudioFrame(Msg: Message): AudioFrame 还原语音消息

+ToJpgData(Msg: Message, ppJpgData: char, JpgSize: unsigned int): void 还原共享屏幕消息

+ToStudentNameAndPassword(Msg: Message, Name: string, Password:string): string 还原使用者登入消息

+ToPopQuiz(Msg: Message): string 还原随机提问消息

+ToPerformance(Msg: Message, aPer:Performance) 还原学生表现消息

+ToQuiz(Msg Message, aQuiz: StudentQuiz) 还原发题目消息

以上函数用来还原消息对象

+Size: unsigned int 消息数据区存储了大小（单位字节）

+IsValid: bool 是否是有效消息

+Data: puchar 消息数据区的只读指针

+START_BYTE: unsigned char 静态常成员,消息的开始符和结尾符

+STOP_BYTE: unsigned char 静态常成员,消息的开始符和结尾符

-m_uSize: unsigned 编码后的消息长度

-Encode(Type: MessageType , Input: unsigned char, Length: unsigned int):bool 将一个字符序列编码成一个消息（加开始符和结尾符号,展开 Input 中的开始符 结尾符 0x80）

-Decode(Type: MessageType , Output: unsigned char, Length: unsigned int):bool 将一个消息解码为字符序列（去掉消息中的开始符和结尾符号,还原消息原始内容）

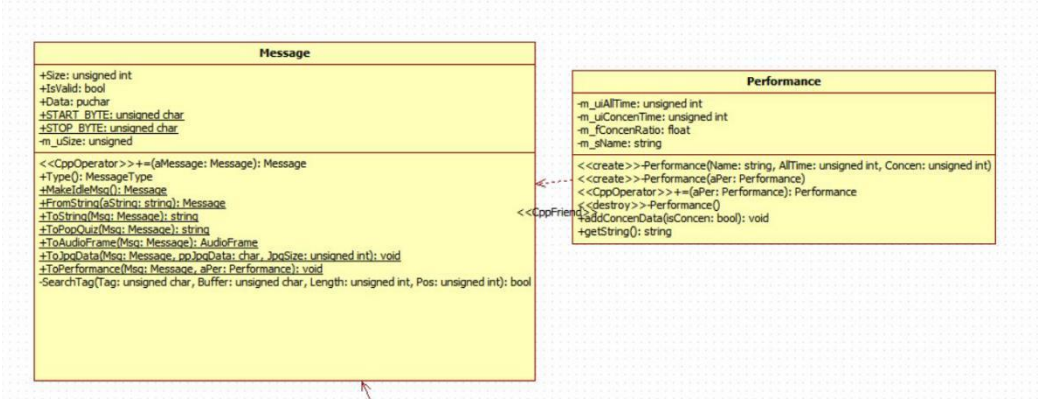
-SearchTag(Tag: unsigned char, Buffer: unsigned char, Length: unsigned int, Pos: unsigned int): bool 搜索 Buffer 中是否包含特定标签（开始符/结尾符）

以上函数用来处理消息的编码与解码问题,主要用来确保网络传输时的数据可靠性

(3) 设计思路:

① 主要有三个方向。第一，Message 函数作为一个具体的消息对象，必须有存储基本信息的变量，如长度、消息内容等。第二，Message 类作为消息中转机制，必须有与各类型的转换接口，每个类都对应到两个函数 from/to。第三，Message 类作为网络传输用消息，应该设有识别机制，在收发消息时以此机制作为判断何时应该开始/停止收发信息，具体实现于 encode/decode 两个私有成员函数

② 编码方式。在发送消息前编码，在收到消息后解码，编码方式为在消息的头尾分别插入特殊字符，并将消息中间该类字符替换掉。



13. Performance(核心类)

(1) 概述: 用来统计学生课堂表现(专注度)

(2) 功能介绍(仅挑选部份功能)

-m_uiAllTime: unsigned int 在课堂总时长

-m_uiConcenTime: unsigned int 总专注时长

-m_fConcenRatio: float 专度时间占比

-m_sName: string 学生姓名

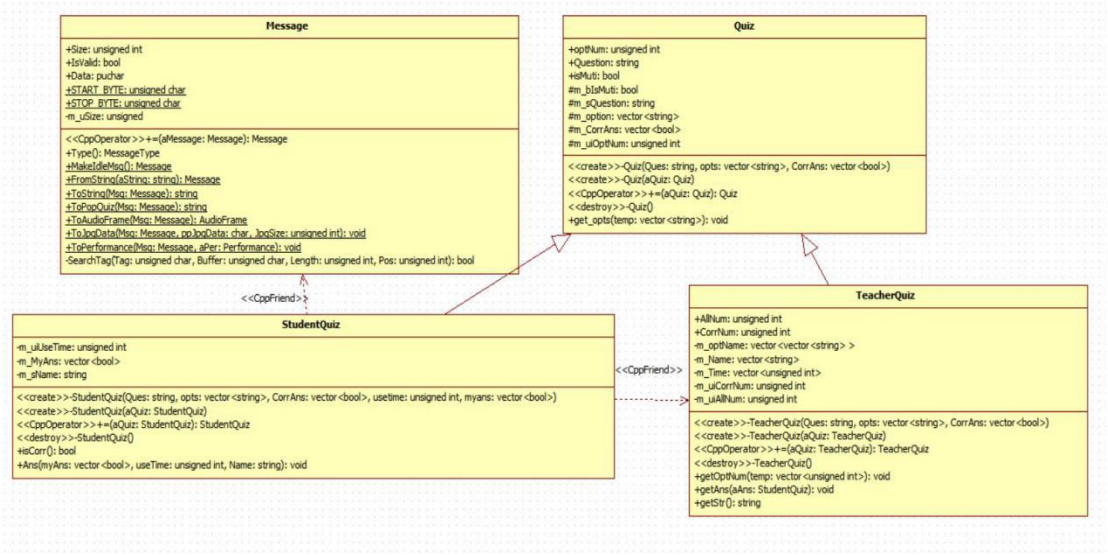
+addConcenData(isConcen: bool): void 添加专注度信息

+getString(): string 得到字串类输出

(3) 设计思路

① 将 Message 类作为友元原因是 Message 类会频繁调用 Performance 类数据，若专门为其设计多个数据读取接口，会导致更大的数据曝露风险

② 唯一的数据获取接口是得到字符串型输出，因为专注度输出并不需要在界面上与使用者互动，因此只要输出为特定格式的字符串，就能让使用者获取全部的信息，如此能减少不必要的数据曝露



14. Quiz(核心类)

(1) 概述: 是 StudentQuiz 与 TeacherQuiz 的基类，记录一个问题的基本信息，如同真实世界题库中的题目

(2) 功能介绍(仅挑选部份功能)

+optNum: unsigned int 选项个数

+Question: string 问题描述

+isMuti: bool 是否为多选题

#m_bIsMuti: bool 是否为多选题

#m_sQuestion: string 问题描述

#m_option: vector<string> 选项描述

#m_CorrAns: vector<bool> 正确答案

#m_uiOptNum: unsigned int 选项个数

+get_opts(temp: vector<string>): void 获得选项描述

(3) 设计思路:

① 此类成员包含了一个问题应该有的信息，并且对正确答案外的所有 数据都提供了获取的接口，基于安全考量对外隐藏了正确答案。

② 在设计时考量到除了正确答案外 ,其他数据如问题选项等都是能够公诸于世的信息，因此都获取接口，且如此设计能让使用者界面更具有动态性

15. StudentQuiz(核心类)

(1) 概述: 学生问题类，处理单一学生回答问题的相关事宜，类似现实世界的考卷

(2) 功能介绍(仅挑选部份功能)

-m_uiUseTime: unsigned int 答题时间

-m_MyAns: vector<bool> 学生答案

-m_sName: string 答题人姓名

+isCorr(): bool 是否正确

+Ans(myAns: vector<bool>, useTime: unsigned int, Name: string): void 回答问题

(3) 设计思路:

① 将 Message 与 TeacherQuiz 设为友元原因: 因为这两个类都需要频繁调用 StudentQuiz 的内容(Message::FromQuiz 与 TeacherQuiz::getAns)，若不设为友元，则必须设计多个接口以提供各种信息，如我的答案、答题时间等，如此将导致许多信息不必要的曝露，因此设计为友元类。且我确保上述两个函数都为 const 函数，以保证数据安全性。

② 此类函数对外两个接口分别对应现实两种动作: 批改与回答考卷

16. TeacherQuiz(核心类)

(1) 概述: 继承自 Quiz 类，记录每题整个课堂的整体答题情况，类似于现实世界的成績单

(2) 功能介绍(仅挑选部份功能)

-m_optName: vector<vector<string> > 每个选项的答题名单

-m_Name: vector<string> 所有答题者名单

-m_Time: vector<unsigned int> 答题人费时

-m_uiCorrNum: unsigned int 正确人数

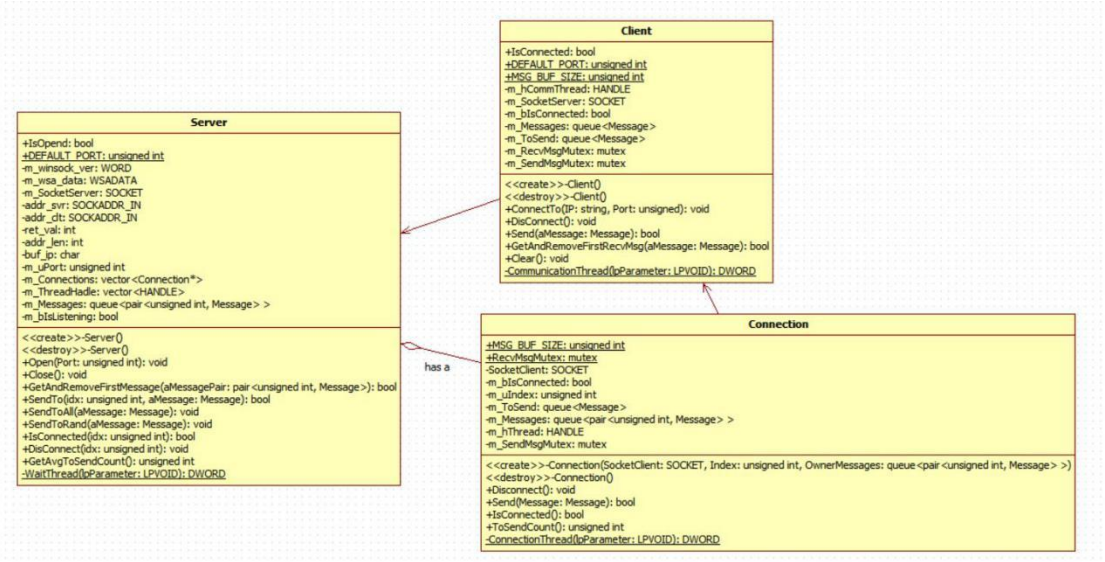
-m_uiAllNum: unsigned int 总答题人数

+getAns(aAns: Student) 得到新的学生回答，更新统计结果

+getStr(aStr: string) 得到文本形式输出

(3) 设计思路:

① 函数得到文本输出是将类对象内容以指定形式的字符串输出 ·因为此类对象的输出窗口不要求与用户互动 ·因此可以以文字形式的输出取代对外提供各个变量接口地方法，如此能够减少不必要的信息曝露，增加安全性。



17. Server(核心类)

(1) 概述: 服务器类，用于监听客户端的各种信号，并做出相对应的行动，用来处理网络通信中教师端的各种通信功能

(2) 功能介绍(仅挑选部份功能)

-m_uPort: unsigned int 端口号

-m_Connections: vector<Connection*>用于存放已经连接进来的全部客户端的 Connection 对象指针

+DEFAULT_PORT: unsigned int 静态成员默认端口号

以上为服务器基本信息

-m_ThreadHadle: vector<HANDLE>监听线程句柄

-m_Messages: queue<pair<unsigned int, Message> >接收消息的队列

+IsOpend: bool 是否处于监听状态

+GetAndRemoveFirstMessage(aMessagePair: pair<unsigned int, Message>): bool 从消息队列里提取并删除第一条。如果有第一条的话，返回 ture 否则返回 false

以上为处理监听消息相关函数

+Open(Port: unsigned int): void 打开端口，等待客户端连入。Port 默认值 DEFAULT_PORT 是类的静态成员

+Close(): void 关闭端口，断开现有连接，并拒绝后续客户端连入

+IsConnected(idx: unsigned int): bool 判断指定连接是否在线

+DisConnect(idx: unsigned int): void 断开指定连接

以上为处理用户端登入登出相关函数

+SendTo(idx: unsigned int, aMessage: Message): bool 向 idx 指定的客户端发送一个消息，idx 从 0 开始

+SendToAll(aMessage: Message): void 群发消息

+SendToRand(aMessage: Message): void 发消息给随机对象

以上为处理消息发送相关函数

(3) 设计思路:

① 仍是分为两个设计方向。首先基本信息方面，Server 端的基本信息如端口号、所有 connection 对象应该设置为私有成员并设定部份接口函数提供如端口号等必要信息。此外必须要有四个接口函数对应所需的四个功能，分别是发送 Message 给各户端、连接/断开客户端、开启/关闭接收线程与获得接收到的消息。

② 发送消息时需调用接口函数，通过间接调用指定 Connection 的成员函数，完成消息发送

③ 接收 Client 发来的信息时，线程会将收到的信息存储在队列，外界可调用接口函数获取队列头

④ 拥有(has a)connection 列表，每个 connection 代表的是一个 client，发送消息时通过 connection 中转

18. Connection(核心类)

(1) 概述: 连接类，代表服务器下的客户端，具体来说，Server 将利用 Connnection 给 Client 发送消息，并且利用 Connection 记录 Client

(2) 功能介绍

-m_bIsConnected: bool 表示是否处于与客户端的连接状态

-m_uIndex: unsigned int 此 Connection 对象在 Server 的 Connections 里的下标

+IsConnected(): bool 返回值表示是否与客户端处于连接状态。断开连接后，此 Connection 对象仍然存在，直到被销毁。但断开后，发送消息的功能就不可用了

以上为 Connection 类的基本信息

-m_Messages: queue<pair<unsigned int, Message> >服务器的(接收)消息的队列的引用，向此成员添加消息，实际是添加到了服务器的消息队列中

-m_hThread: HANDLE 收发消息句柄

以上为辅助 Server 端处理消息监听的相关函数

-m_ToSend: queue<Message>要发送给客户端的信息队列

-m_SendMsgMutex: mutex 用于实现发消息队列的有序插入和删除

+Send(Message: Message): bool 用于向客户端发送一个消息

+ToSendCount(): unsigned int 返回还有多少个消息没发送出去，如已断开连接返回 0

以上为处理消息发送的相关函数

+Disconnect(): void 用于断开与客户端的链接

(3) 设计思路

① 主要分为两个部份。第一，Connection 类的基本信息，如是否处于连接状态，在 Server 端的序号。第二，用于发送消息的相关接口函数，主要以两个函数完成两个功能，送出消息与断开客户端

19. Client(核心类)

(1) 概述

(2) 功能介绍(仅挑选部份功能)

+IsConnected: bool 连接状态，对-m_bIsConnected: bool 的常引用

+DEFAULT_PORT: unsigned int 静态数据成员，默认端口号

+MSG_BUF_SIZE: unsigned int 静态数据成员，默认消息缓冲区大小，只在接收时有用

-m_hCommThread: HANDLE 通信线程句柄

-m_SocketServer: SOCKET Server 端 Socket

-m_bIsConnected: bool 储存连接状态

以上为 Client 类基本信息

-m_Messages: queue<Message>收到的消息队列

+Send(aMessage: Message): bool 发送一个消息

以上为接收消息相关函数

-m_ToSend: queue<Message>待发的消息队列

+GetAndRemoveFirstRecvMsg(aMessage: Message): bool 获取并移除最早收到的消息

以上为发送消息相关函数

+ConnectTo(IP: string, Port: unsigned): void 连接到指定 IP 和 Port

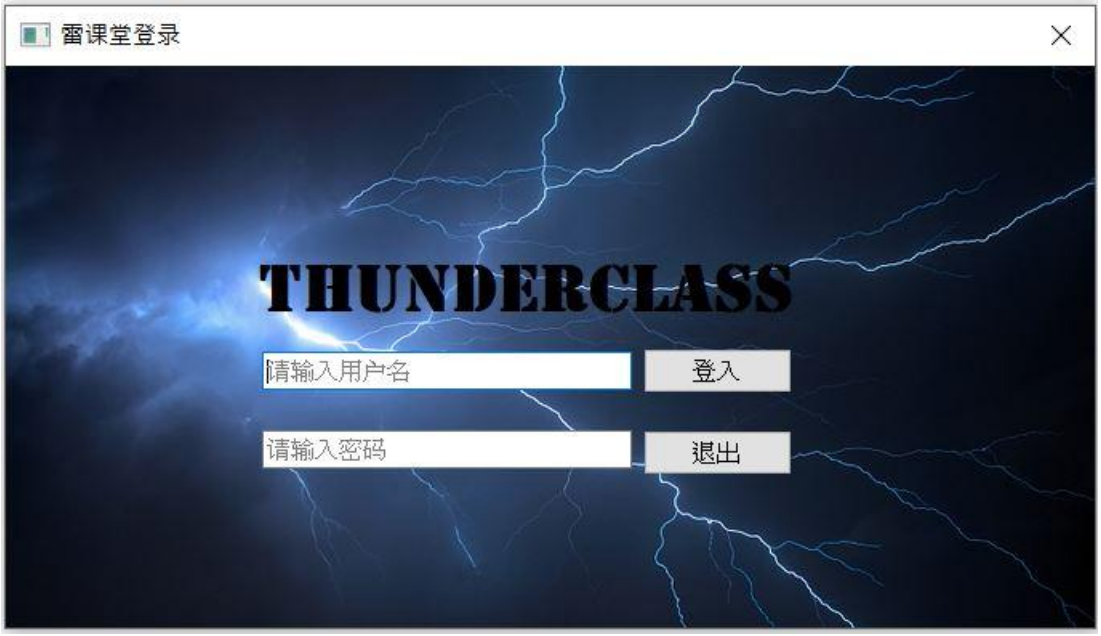
+DisConnect(): void 断开连接

以上为连接/断开与伺服器的连线相关函数

(3) 设计思路

① 除了记录基本信息外，Client 类负责三项工作:接收消息、发送消息与管理与伺服器的连接，因此在每个功能都设计相对应的接口函数

四、 界面设计



1. 登入界面: 采用最简单的页面布局，提供使用者输入账号密码的栏位，并且有登入与退出两个按钮。功能十分简单，业务流程类只对应了一个登入函数。

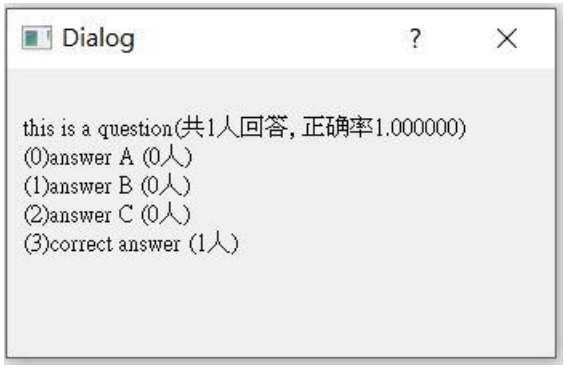
2. 教师界面:

每个功能设计相对硬的按钮，布局共两行。上方为各种功能选单，包含上课/下课

按钮，共享屏幕/语音勾选栏，文本消息填写栏与发送钮，随机提问钮，出题钮(出题后会改为收题钮)，显示答题结果钮。左下方栏位显示目前在线学生清单，右方栏位显示学生所发送的文字。教师业务流程类，设计为每个功能对应一个接口函数(用来处理问题)与一个信号槽函数(用来更改界面)，目前界面设计也是一个功能对应一个函数，因此看似完全耦合。但若未来需要更改界面功能，可以使用旧功能的组合实现新功能(因为每个功能在设计时都是有物理意义的)，因此我认为这样的架构是符合界面与业务流程类的解耦的。

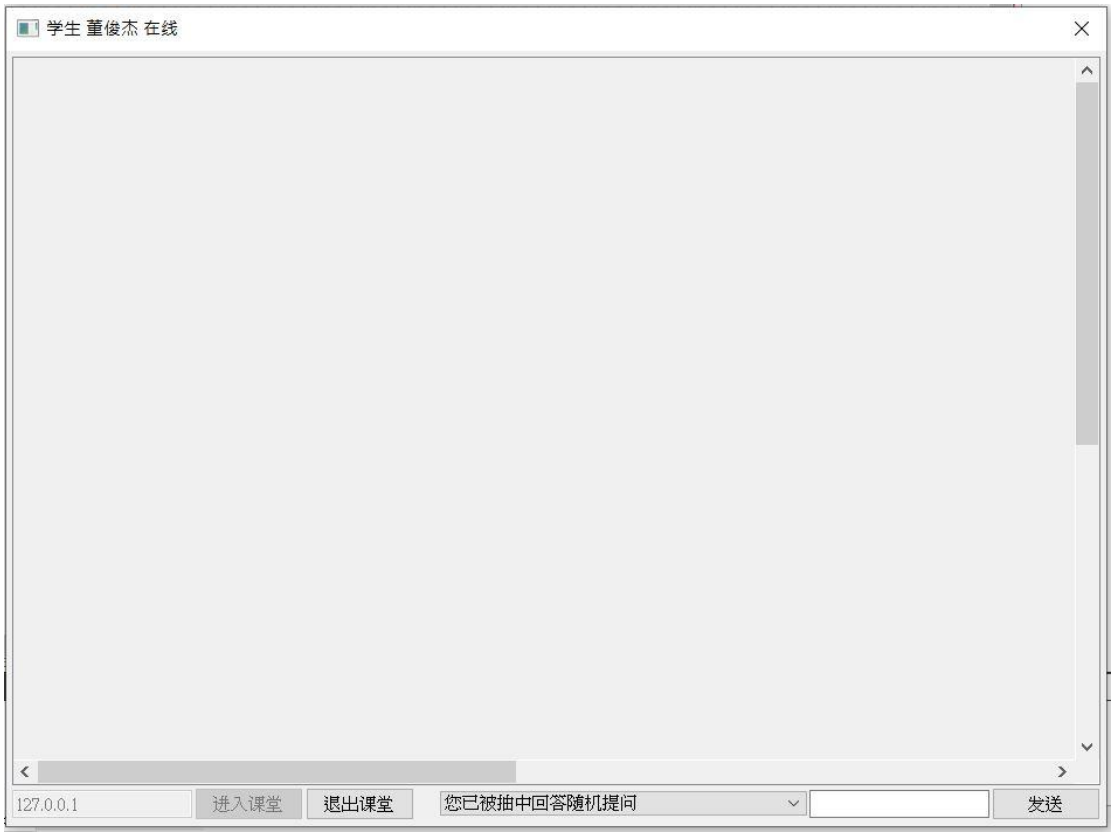
3. 创建问题界面

左侧为输入题目输入选项栏位，右侧可勾选正确答案，并有出题钮。此处设定判断:若没有填写问题或没有勾选答案则无法送出 ·若尚未填入前一个选项则此选项无法填写也无法设为正确答案(如目前的左下方栏位)。这样的判断有助于辅助增加核心类的鲁棒性。



4. 问题结果统计界面

将文字形式统计结果输出在视窗中。



5. 学生界面

中间大面积的区域用来接收屏幕共享。下方有一排功能列表，分别是(左到右): 输入教师端 IP，进入/退出课堂，显示教师端发出的文本消息栏，填写/发送文本消息栏位/按钮。信号的耦合与解耦同教师界面。

6. 管理员界面

左侧为输入使用者账号、密码、类型栏位。右侧可以选择要删除或是新增对象。新增与删除分别对应业务流程类函数。

五、 测试与排错

报错 must use class tag to refer to type in this scope
原因 变量或函数与类型名称重复
修复方法 更改变量或函数名称

报错 无

原因 double = int / int 得到结果为 0
修复方法 将后面变量强制转换为 double

报错 'Message' does not name a type
原因 两个类文件互相引用
修复方法 在某个类中先声明 class A, 并去除其引用

报错 No rule to make target 'questionwindow.ui', needed by 'ui_questionwindow.h'. Stop.
原因 删除挡案却没有从项目中移除
修复方法 打开项目的.pro 文件,删除重复和无效的部分:

报错 无
原因 更改物件名称后，信号槽函数无法使用
修复方法 重新命名信号槽函数

报错 'std::__cxx11::string' (aka 'basic_string<char>') to 'const QString
原因 格式不符规范
修复方法 使用 Qstring::FromStdString()转换

报错 called object type 'int' is not a function or function pointer
原因 int 变量名与调用函数重名
修复方法 更改变量名

报错 undefined reference...
原因 没有在 process 类内加 Q_OBJECT
修复方法 加入 Q_OBJECT 后 Qmake

以上编译器能够报错的问题，主要通过上网查找资料(将报错贴上查找)得到解答。另外遇到许多闪退的状况，此时我主要使用 qDebug 在疑似出错的附近输出可能有误的值(之所以不使用调试 debug 是因为这学期初参加新生 C 语言大赛时，因为所写的 AI 无法在调试的环境下运行，因此养成了以 print 寻找错误的习惯)，并在程序闪退后前往 log 日志查看何处开始值出错了，例如我曾经因为 connect 函数内类型名称输错，导致某变量值无法修改，于是我在每个变量出没的函数中输出该变量的值雨涵数名，最后寻找出问题所在点。

六、 总结与体会

我认为这次大作业与平常编程最大的不同是我会开始是考虑类设计的安全性与整体程序架构的设计问题，过去我们都以实现功能为编程首要目的，许多地方在打代码时能省则省，导致虽然最后能够运行但整体来说代码的架构松散，可读性不高且许多功能或多或少使用不规范的方式实现。这样做最明显的后果有几个，第一是若是碰上程序出问题想找他人协助，

其他人很难理解你的程序逻辑。第二则是自己过了几点再去看，也会因为遗忘了部份细节，导致自己也看不懂自己代码的窘境。第三则是不利于团队编程。其实上学期对此我还不太有感觉，直到这学期初参加新生 C 语言比赛，与同学合作编程时，才意识到其重要性。

我觉得这学期以来，或许我自己编程的能力没有进步，但我"为别人而编程的能力"可说是突飞猛进，肤浅来说注释量显著增加、变量名称开始具有实际意义、该 const 的地方不再遗漏，但我觉得我自己最大的进步是在类设计的安全性考量上。在这次的编程中，我进我自己最大的努力将各个累的公有成员数降到最低，我曾经为了思考如何删去一个常引用变量，大幅修改我的整个程序架构，或许这样对我整个程序的功能没有实际的提升，但在这学期的课中我学到"安全性"的重要性。

此外，第二点进步是在思考类对象与实际物理意义的关系上我也下了不少公夫，在设计每个类时我会思考他的物理意义，并依他在实际生活中的操作来思考他的功能设计。利如在设计问题类时，我就认为正确答案是不应该让外界知道的，因此我用了各式方法巧妙的避开不让外界得到获取正确答案的接口。另外，我对友元类也有更深一步的了解，也对它大为改观，过去的我认为友元类是破坏代码安全性的最大凶手，但在这次编程中 fromMessage 这类函数，使我发现友元的重要性，我一开始为了不使用友元，写了许多常引用与获取私有数据的接口函数，后来我发现这样做对安全性的影响其实已经远远超过友元，因此后来才改回以友元设计。值得一题的是，我一开始认为应该可以只使用友元 Message 类成员函数，但后来试了许多方式，都会因为头文件互相引用报错而作罢。

最后，我想说一些过程中出现的困难，最大的问题是，我在 6/19 中午思考我的程序架构时越想越觉得这样的架构不符合类封装的概念，因此从 12.起从头开始写起，当然我用了许多过去的函数模块，但过程还是比较仓促，不过最后还是写出了我自己认为比较好的成果(类封装部份)。而成果与理想间最大的差异的部份，我原本以为这次编程后我对网络通信等功能会有比较深刻的认知，但或许因为付出的精力还不太足够，目前的我对范老师提供的代码还有许多部份没有完全厘清，许多函数最后都直接当做黑箱函数来用了。此外，我本来想用部份时间做优化共享屏幕的部份，但最后也因为时间关系而没有实现。