C++程序设计课程设计

(2021/2022学年第一学期)

指导教师：庄巧莉、霍旭文

班级：20计算机科学与技术(4)
学号：2020333503081
姓名：陈伟剑

【工作内容及工作计划】

时间		地点	工作内容	指导教师
2021.12.8	下午	软件实验室10-308	教师讲解课题，以及C++连接数据库相关技术内容	庄巧莉、霍旭文
2021.12.11~15		软件实验室10-308、10-409	需求分析、类设计、数据库设计、部分代码编写	
2021.12.22	下午	软件实验室	中期检查	庄巧莉、霍旭文
2021.12.25~2022.1.2		软件实验室10-308、10-409	代码编写、完善课设报告	
2022.1.7	上午	软件实验室10-409	课设答辩	庄巧莉、霍旭文

目录

一、任务要求

背景

为了使同学们手中闲置的书籍能再次发挥其价值，现在需要你针对在校学生开发一款小型的二手书交易系统。每个同学既可以是卖方，也可以是买方。该系统能帮助卖方同学发布需要售卖的二手书信息，帮助买方同学购买到所需书籍，实现校园内部的便捷交易。

前提假设

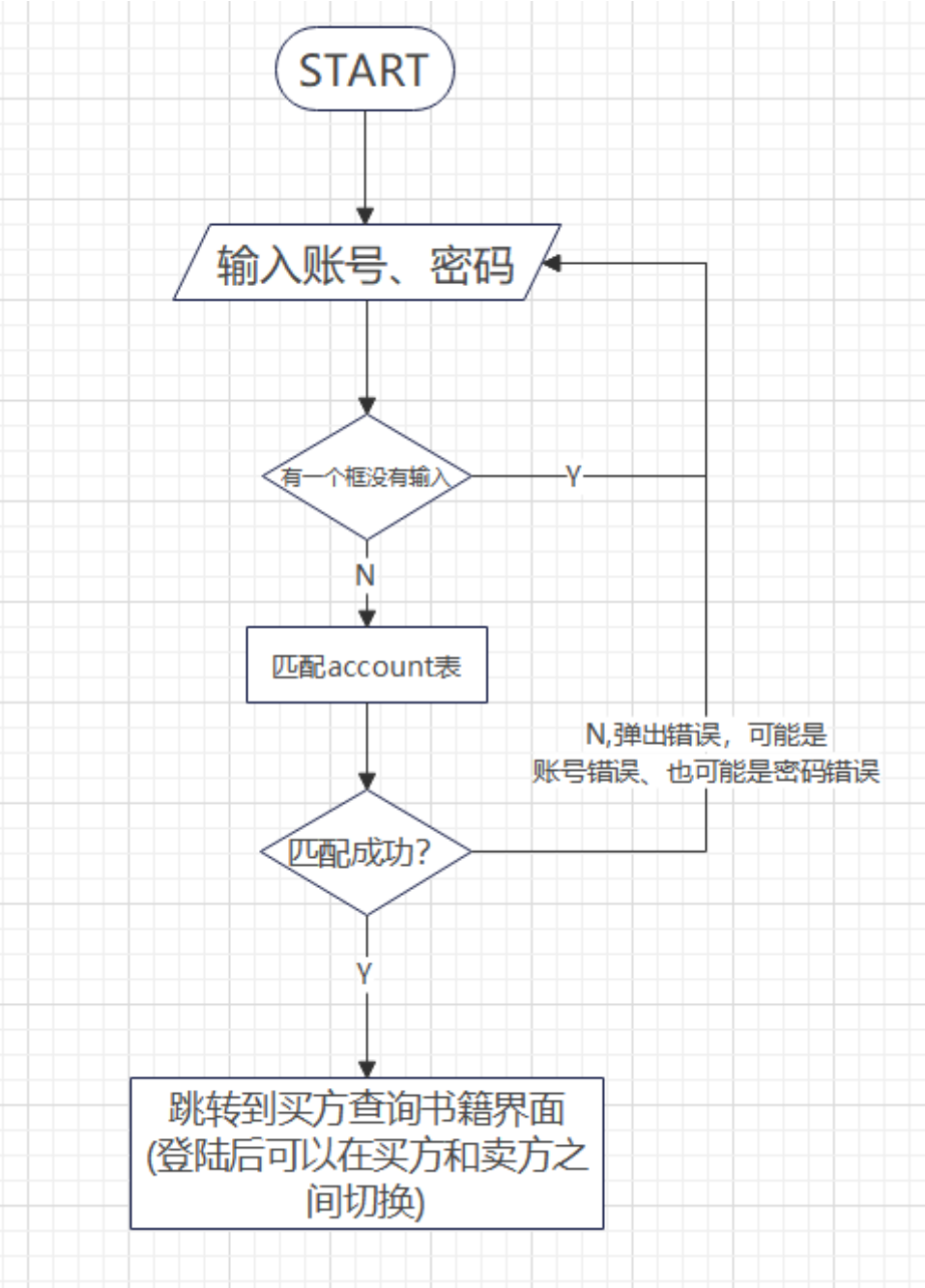
1. 一个账号既能当买方，也能当卖方，不用考虑一个账号的角色属性。用户根据账号、密码登录系统。
2. 买方一次可以购买一本书。
3. 对于一个账号，一种书默认为1本。即不考虑一个账号管理许多本同一种书的情况。因为该平台的首要目的不是实现书店出售大量书籍的功能，而是实现少部分书籍从一个用户转交给另一个用户的功能。

基本任务

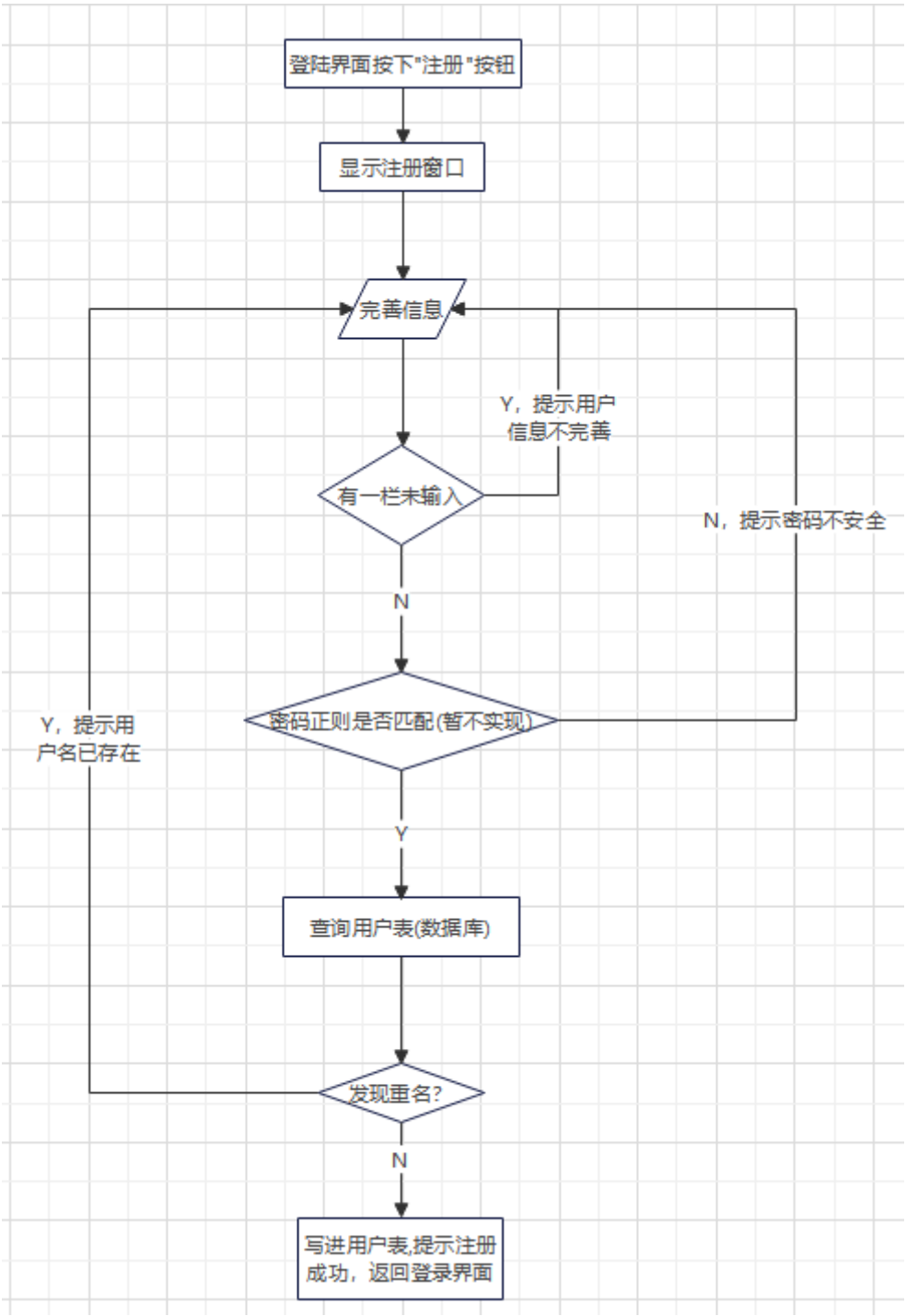
1. 卖方可将需要售卖的书籍信息录入系统，书籍信息包括 ISBN 编号、书籍名称、作者、出版社、价格、新旧程度、取书方式等。买方可查看正在售卖的所有书籍信息，并能够选择购买某一书籍，完成交易。
2. 买方、卖方信息均保存在数据库中，其中，连接数据库所需信息（数据库服务器地址、用户名、密码、数据库名）存放在文件中，程序通过从文件中读取这些信息获得与数据库的连接。实现用户的注册功能，注册信息包括用户名、密码、联系地址、联系电话。
3. 书籍信息和订单信息保存在数据库中，卖方能够对自己的书库进行增删改查。
4. 系统可以查询并统计订单。对于买方，可以查询自己的购买订单并统计数量；对于卖方，可以查询自己的售出订单并统计数量。购买订单和售出订单是订单在买卖方两个不同角度的不同说法。

二、业务流程图

2.1 登录流程

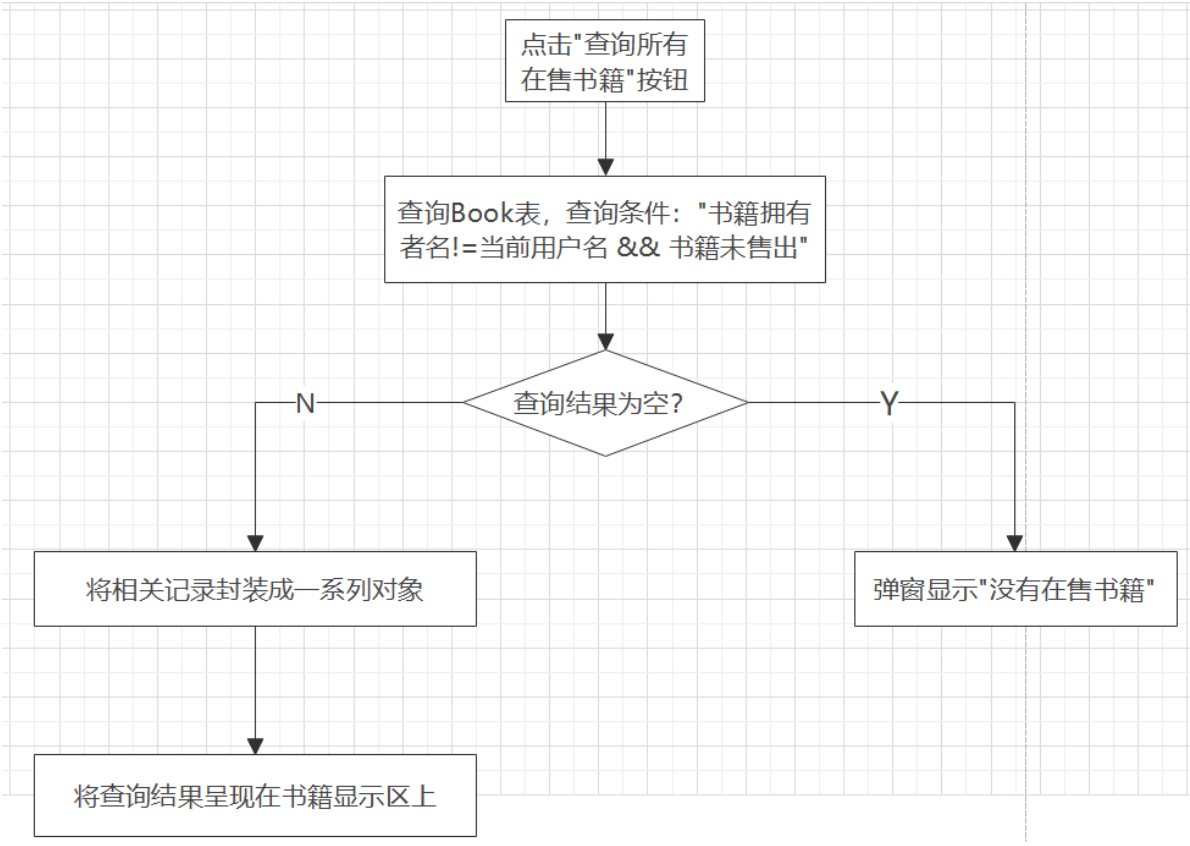


2.2 注册流程

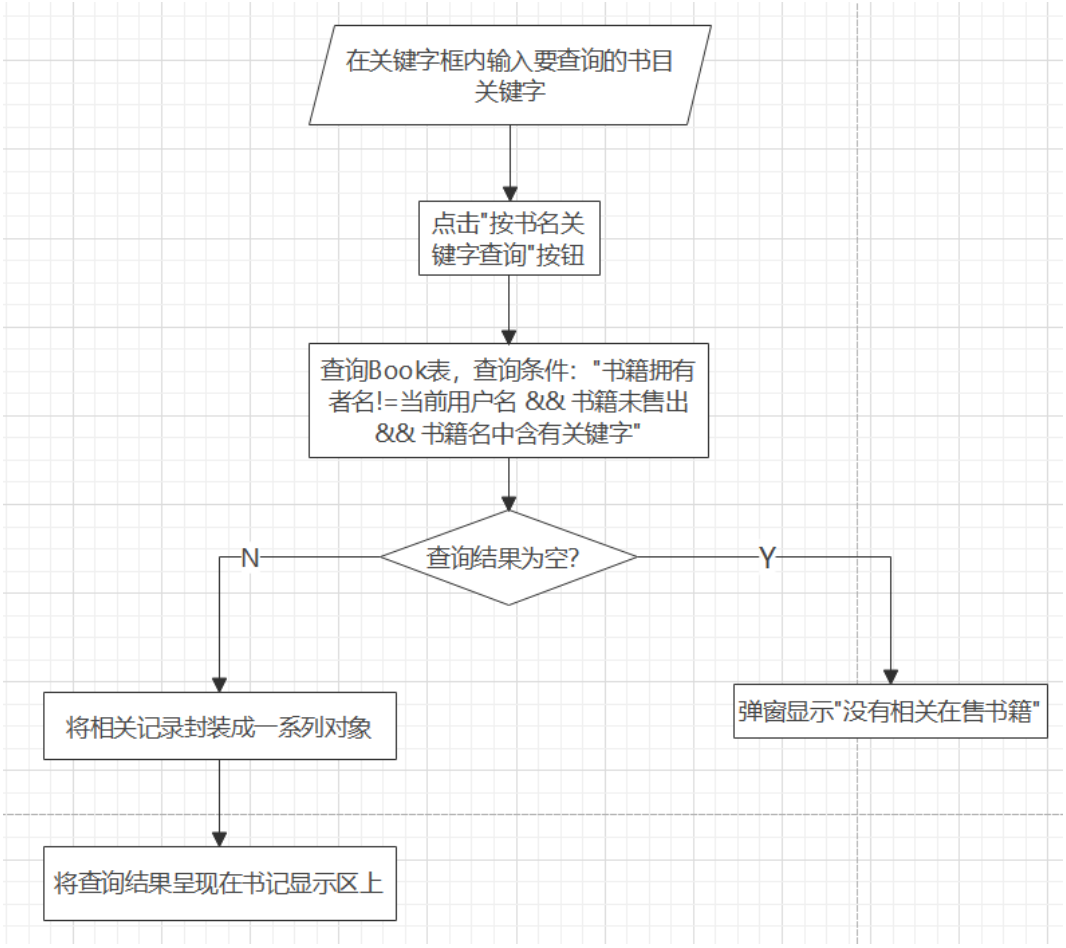

登陆界面按下"注册"按钮
显示注册窗口
完善信息
Y，提示用户
信息不完善
有一栏未输入
N，提示密码不安全
N
密码正则是否匹配(暂不实现)
Y，提示用
户名已存在
Y
查询用户表(数据库)
发现重名?
N
写进用户表,提示注册
成功，返回登录界面

2.3 买方界面

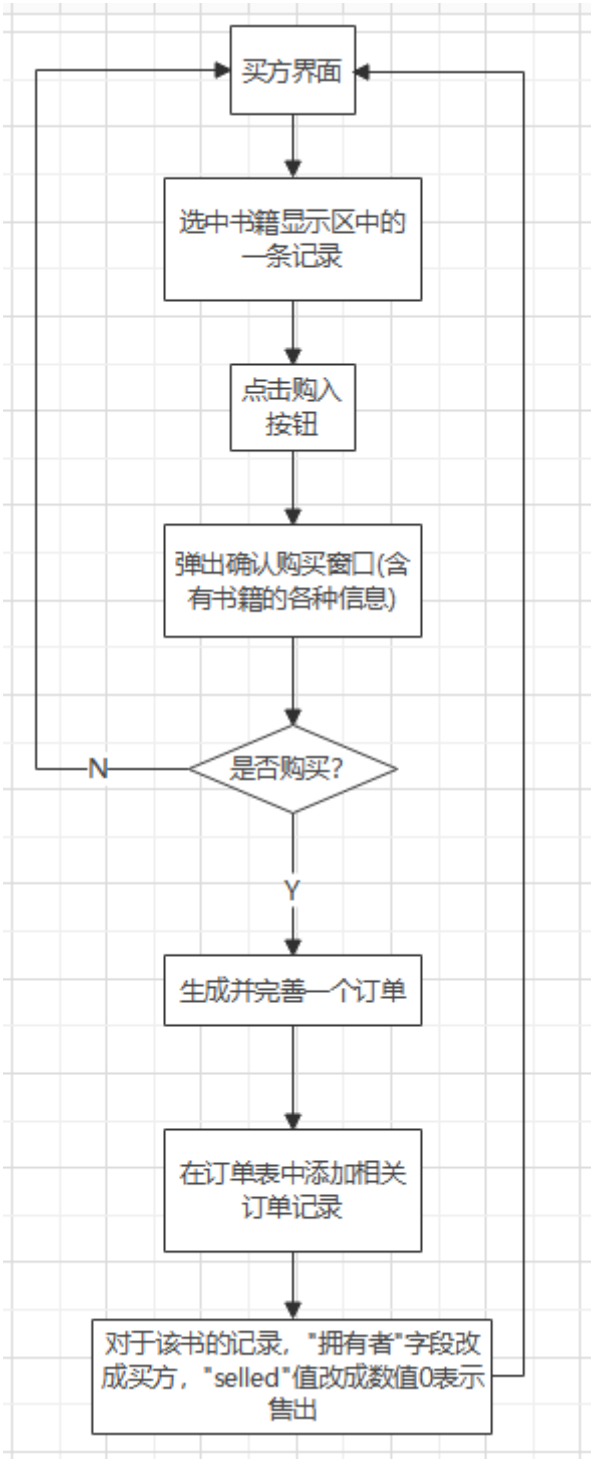
2.3.1 查看正在售卖的书籍



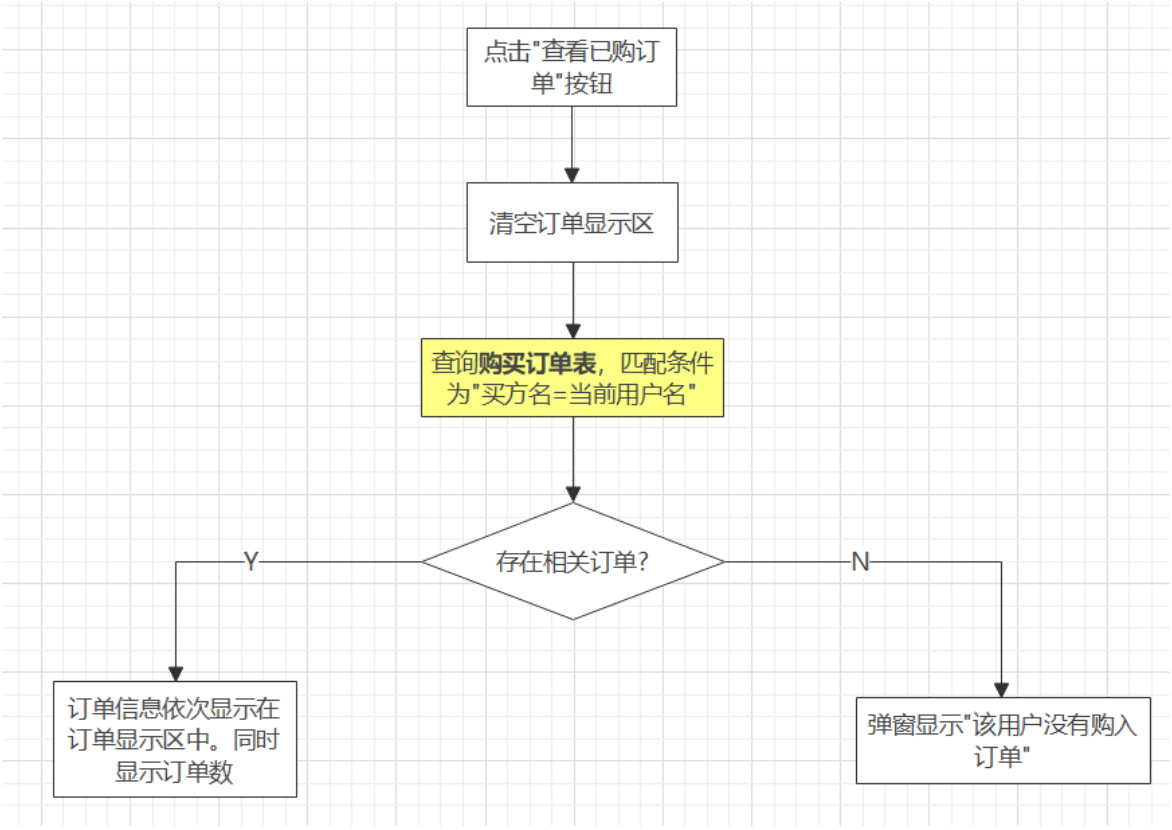
2.3.2 按关键字查看正在售卖的书籍



2.3.3 买书流程

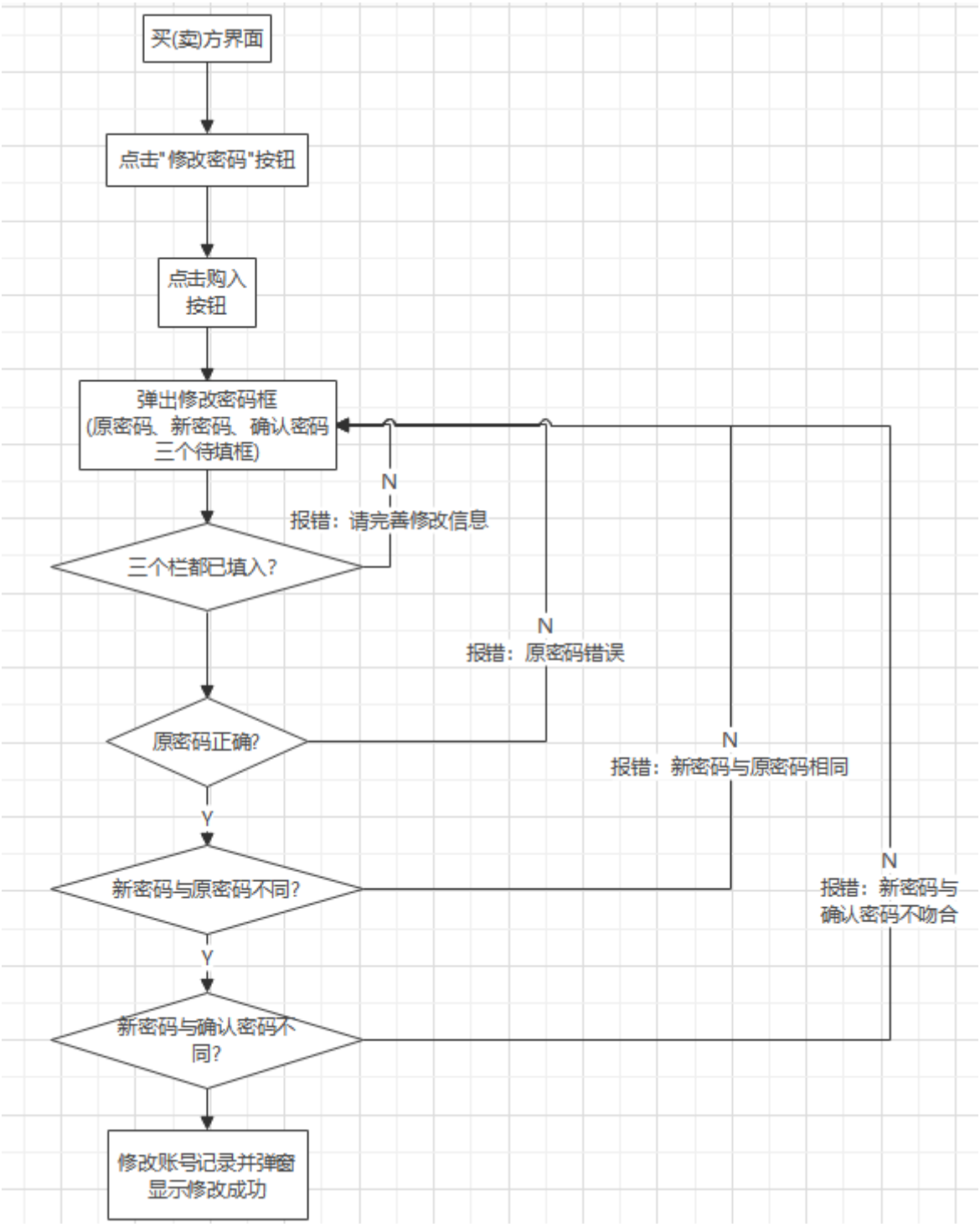

买方界面
选中书籍显示区中的一条记录
点击购入按钮
弹出确认购买窗口(含有书籍的各种信息)
是否购买?
N
Y
生成并完善一个订单
在订单表中添加相关订单记录
对于该书的记录，"拥有者"字段改成买方，"selled"值改成数值0表示售出

2.3.4 查看购买订单


点击"查看已购订单"按钮
清空订单显示区
查询**购买订单表**，匹配条件为"买方名=当前用户名"
存在相关订单?
Y
N
订单信息依次显示在订单显示区中。同时显示订单数
弹窗显示"该用户没有购入订单"

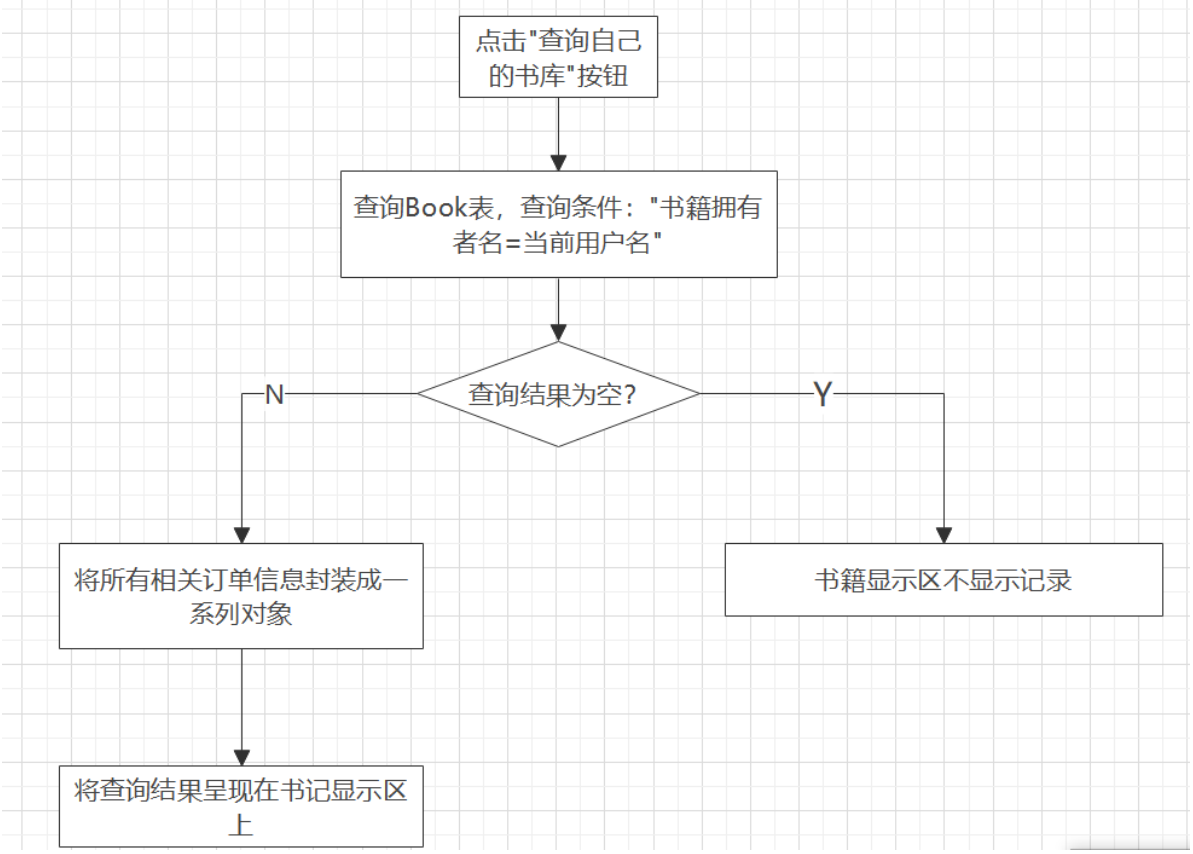
2.3.5 修改密码

注：卖方的修改密码流程同理，故卖方部分会省略修改密码相关内容

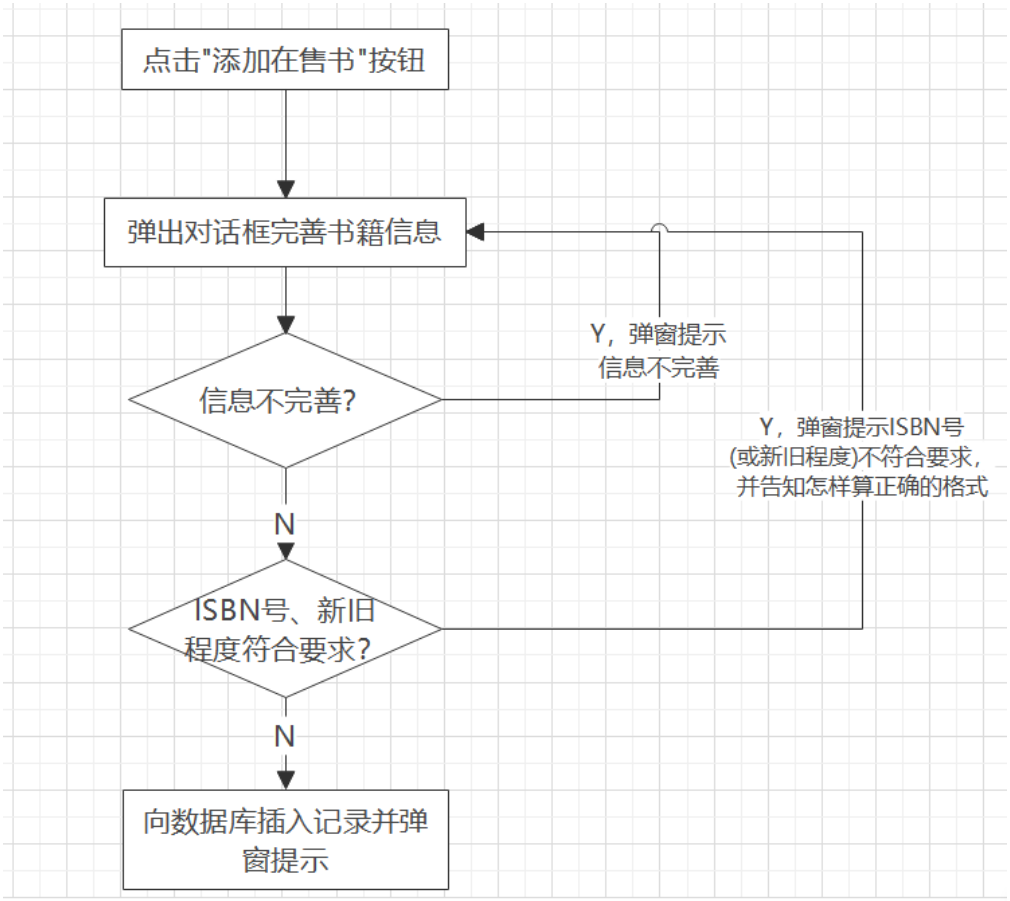


2.4 卖方界面

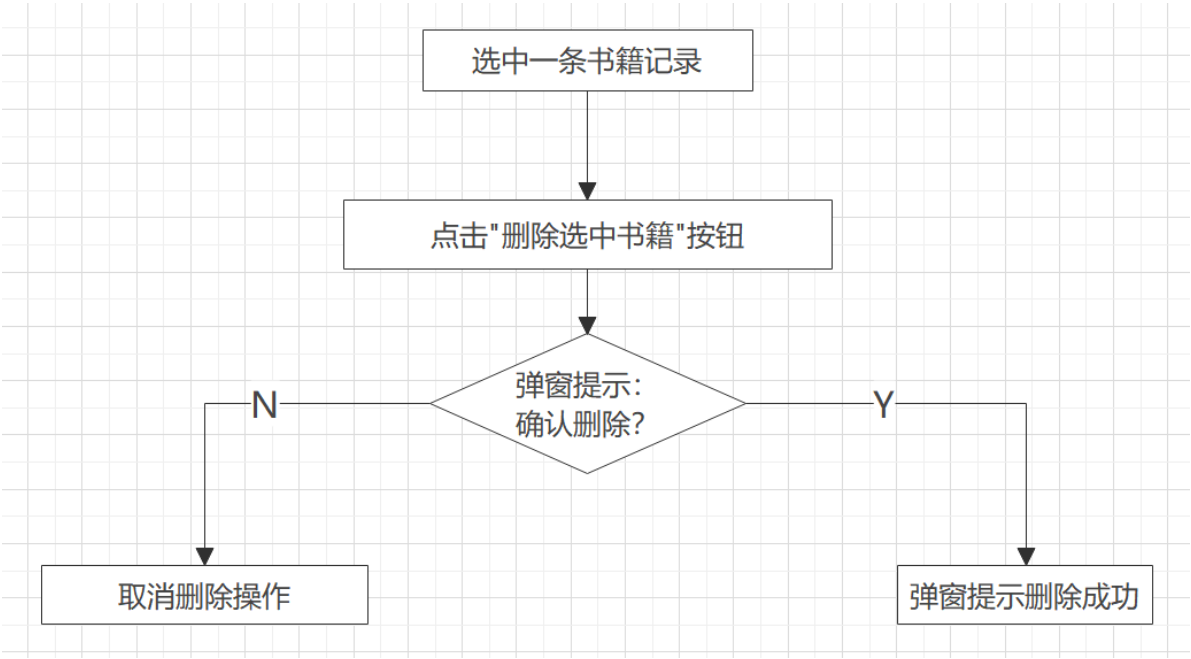
2.4.1 查看自己的书库



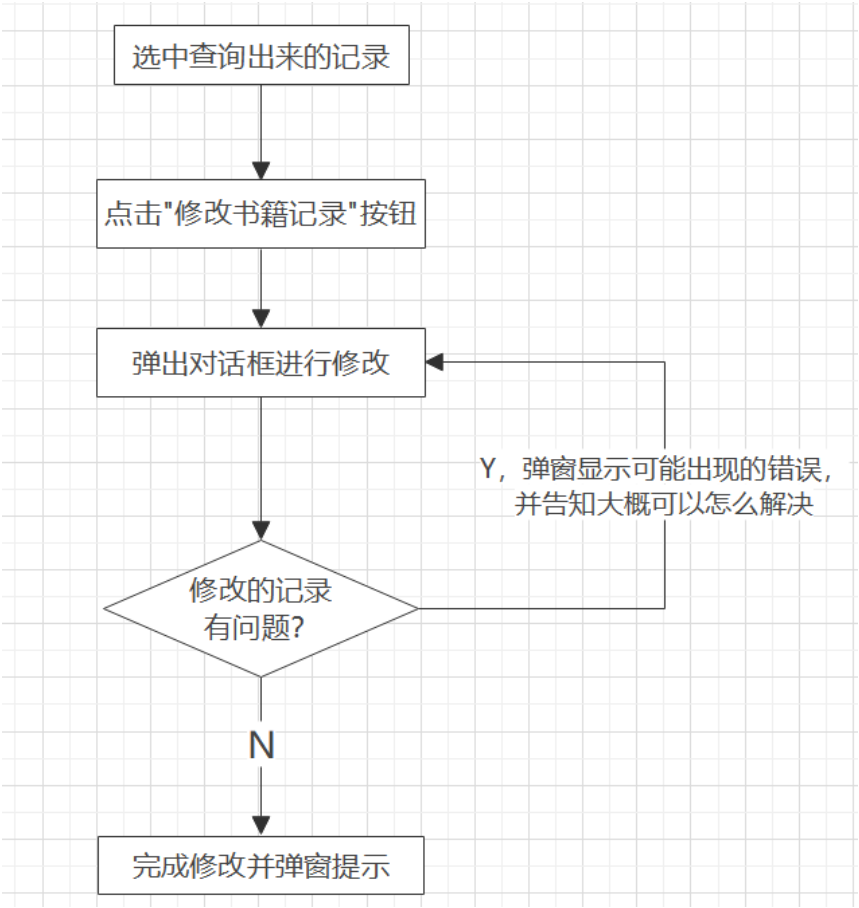
2.4.2 增加一本书



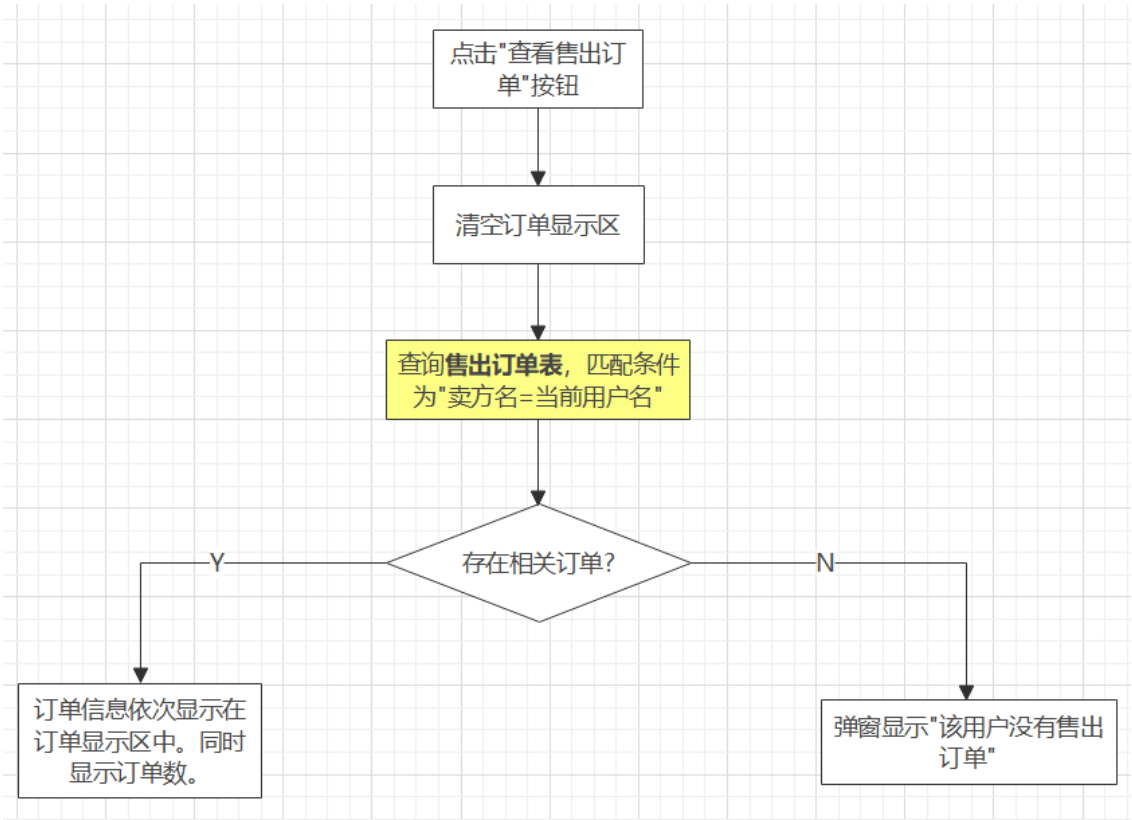
2.4.3 删除一本书



2.4.4 修改一本书

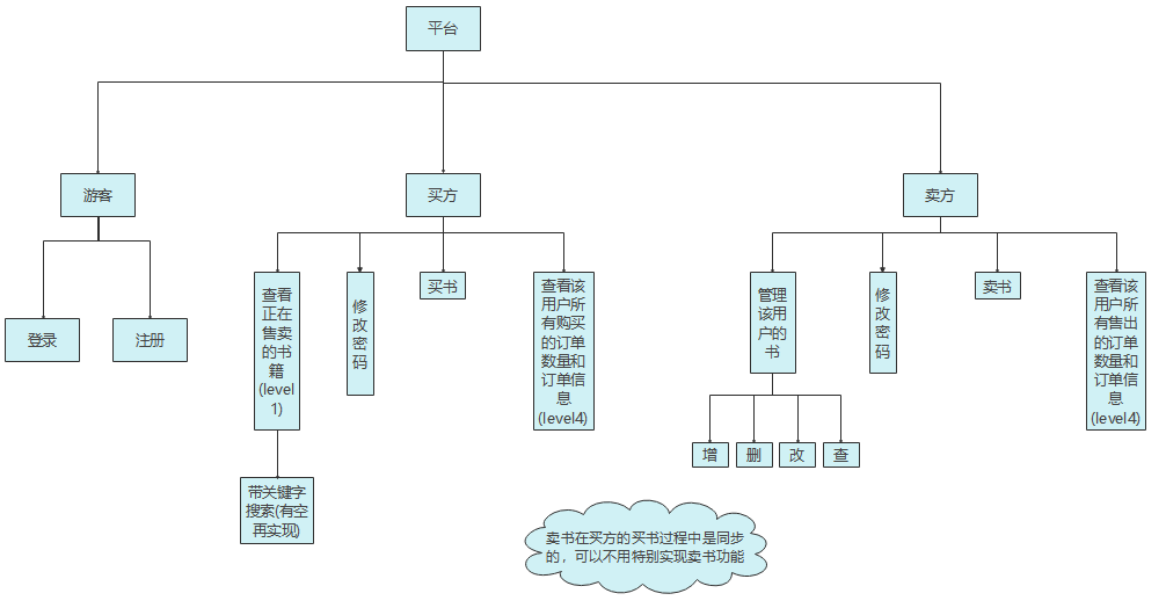


2.4.5 查看售出订单

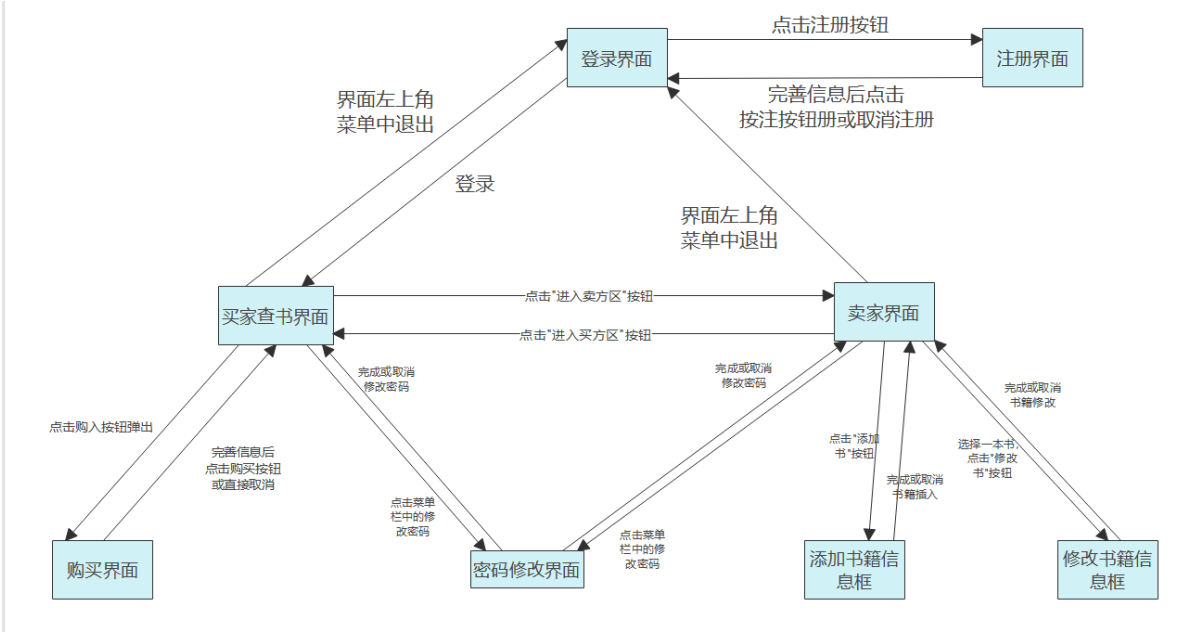


三、系统功能结构图

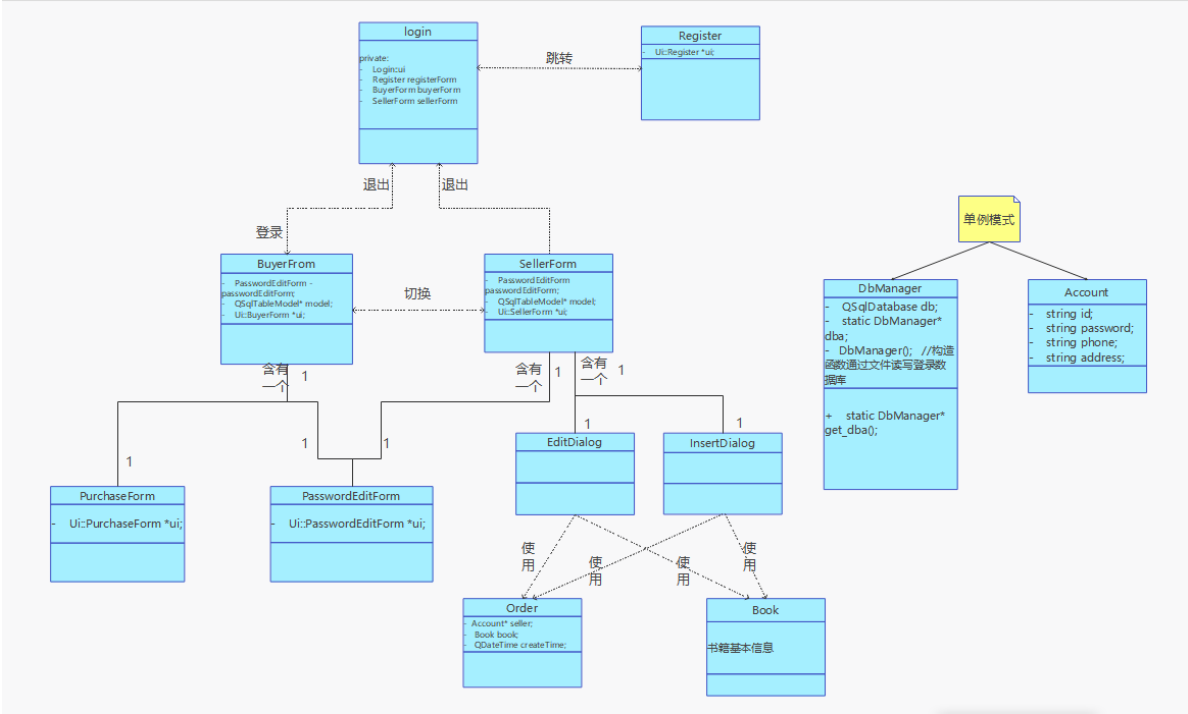
3.1 总体功能布局



3.2 界面联系布局



四、类的设计



涉及的类：用户类(Account)、书籍类(Book)、订单类(Order)、数据库连接类(DbManager)、登陆界面类(Login)、注册界面类(Register)、买方界面类(BuyerForm)、卖方界面类(SellerForm)、修改密码界面类(PasswordEditForm)、购买订单对话框类(PurchaseDialog)、插入书记信息对话框类(InsertDialog)、修改书记信息对话框类(EditDialog)。(共12个)

五、数据库设计

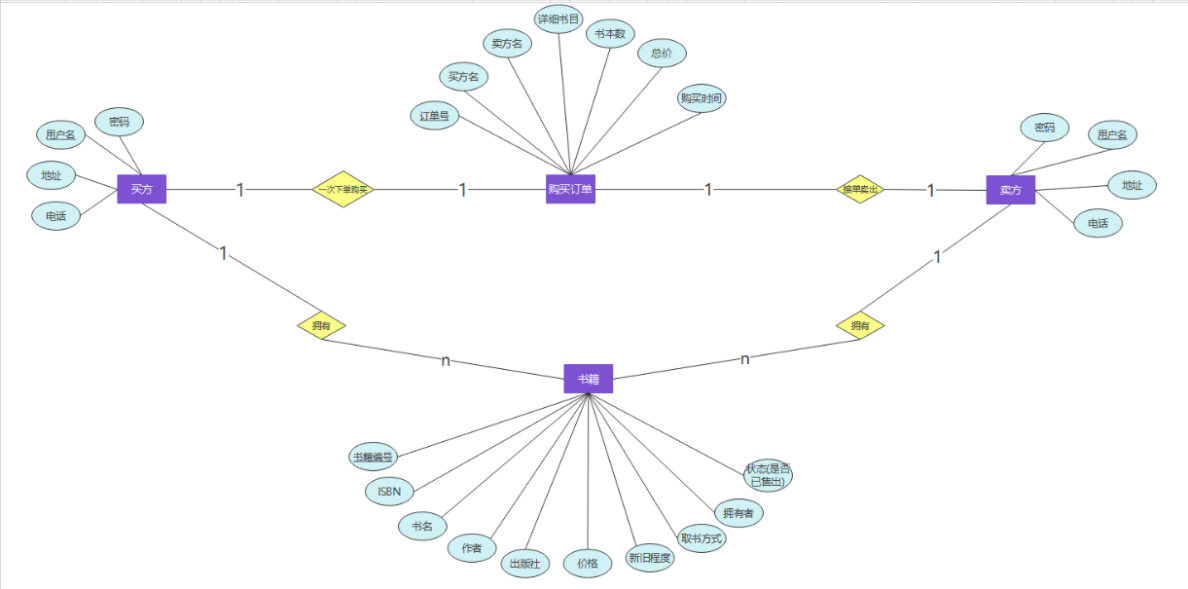
5.1 对象提取和关系说明

涉及三个对象：账号(买方和卖方)对象，书籍对象，订单对象

其中：

1. 一个账号可以是卖方，也可以是买方。
2. 买方一次购买一本书籍。
3. 一次购买生成一个订单。
4. 一个卖(买)方可以拥有多本书，且不考虑书籍重复。
5. 买方买到书后可能会将书的记录删除，如果订单直接以书的内容为外键，则会被级联删除。因此订单的内容可以适当冗余，以防级联删除。

5.2 ER图



5.3 表定义及相关SQL语句

5.3.1 表设计

注：横线为主键、加粗字为外键

$$account(\underline{id}, password, phone, address);$$

$$book(\underline{id}, isbn, writer, publisher, \boldsymbol{owner}_{account}, takeMethod, price, status, selled);$$

$$bookOrder(\underline{id}, \boldsymbol{buyerName}_{account}, \boldsymbol{sellerName}_{account}, \boldsymbol{bookName}_{book}, price, createtime);$$

5.3.2 SQL建表

```sql
/*创建account表*/
CREATE TABLE `account` (
  `id` varchar(40) NOT NULL,
  `password` varchar(20) NOT NULL,
  `phone` varchar(20) NOT NULL,
  `address` varchar(255) NOT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb3;



/*创建book表*/
CREATE TABLE `book` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `isbn` varchar(80) NOT NULL,
  `bookName` varchar(255) NOT NULL,
  `writer` varchar(40) NOT NULL,
  `publisher` varchar(40) NOT NULL,
  `owner` varchar(40) NOT NULL,
  `takeMethod` varchar(255) NOT NULL,
  `price` float(9,2) NOT NULL,
  `status` tinyint NOT NULL,
  `selled` tinyint NOT NULL,
  PRIMARY KEY (`id`),
  KEY `owner` (`owner`),
  CONSTRAINT `owner` FOREIGN KEY (`owner`) REFERENCES `account` (`id`) ON
DELETE CASCADE ON UPDATE RESTRICT
) ENGINE=InnoDB AUTO_INCREMENT=13463 DEFAULT CHARSET=utf8mb3;



/*创建bookOrder表*/
CREATE TABLE `bookorder` (
  `id` int unsigned NOT NULL AUTO_INCREMENT,
  `buyerName` varchar(40) NOT NULL,
  `sellerName` varchar(40) NOT NULL,
  `bookName` varchar(255) NOT NULL,
  `Price` float(9,2) NOT NULL,
  `createTime` datetime NOT NULL,
```

```
  PRIMARY KEY (`id`),
  KEY `buyerName` (`buyerName`),
  KEY `sellerName` (`sellerName`),
  CONSTRAINT `buyerName` FOREIGN KEY (`buyerName`) REFERENCES `account` (`id`) ON DELETE CASCADE,
  CONSTRAINT `sellerName` FOREIGN KEY (`sellerName`) REFERENCES `account` (`id`) ON DELETE CASCADE
) ENGINE=InnoDB AUTO_INCREMENT=19 DEFAULT CHARSET=utf8mb3;
```

5.4 视图定义及相关SQL语句

详细订单视图(连接account表和bookOrder表，在bookOrder表的基础上加入买(卖)方的联系电话和联系地址)

$$detailOrder(id,\\ buyerName, buyerPhone, buyerAddress,\\ sellerName, sellerPhone, sellerAddress,\\ bookName, price, purchaseTime);$$

```
CREATE VIEW `detailorder` AS
select `bookorder`.`id` AS `id`,
        `buyer`.`id` AS `buyerName`,`buyer`.`phone` AS `buyerPhone`,`buyer`.`address` AS `buyerAddress`,
        `seller`.`id` AS `sellerName`,`seller`.`phone` AS `sellerPhone`,`seller`.`address` AS `sellerAddress`,
        `bookorder`.`bookName` AS `bookName`,`bookorder`.`Price` AS `price`,
        `bookorder`.`createTime` AS `purchaseTime`
from (
    (`bookorder`
        join `account` `buyer` on((`buyer`.`id` = `bookorder`.`buyerName`)))
        join `account` `seller` on((`seller`.`id` = `bookorder`.`sellerName`))
)
```

六、程序代码与说明

6.0 all_headers.h

```
#pragma once
#ifndef ALL_HEADERS_H
#define ALL_HEADERS_H


//CPP
#include<iostream>
#include<fstream>
#include<algorithm>
#include<cstring>
#include<ctype.h>
#include<malloc.h> /* malloc()等 */
#include<limits.h> /* INT_MAX等 */
#include<cstdio> /* EOF(=^Z或F6),NULL */
#include<cstdlib> /* atoi() */
#include<cstdio>
#include<string>
#include<queue>
#include<stack>
#include<vector>
#include<map>
#include<set>
#include<list>
#include<io.h> /* eof() */
#include<process.h> /* exit() */



//QT-->BASIC
#include <QMainWindow>
#include <QWidget>
#include <QMessageBox>
#include <QApplication>
#include <QCoreApplication>
#include <QLocale>
#include <QTranslator>
#include <QObject>
#include <QFile>   //QT文件读写
#include <QFileDevice>
#include <QString>
#include <QStringList>
#include <QDateTime>
#include <QRegularExpression>
#include <QDateTime>
#include <QRegExp>  //正则表达式

//QT-->SQL
#include <QApplication>
#include <QLocale>
#include <QTranslator>
#include <QStandardItemModel>
```

```
#include <QSqlQuery>
#include <QSqlQueryModel>
#include <QSqlDatabase>
#include <QSqlError>
#include <QDebug>  //QT控制台输出
#include <QSqlTableModel>
#include <QSqlRecord>

#define OK 1
#define ERROR 0
#define TRUE 1
#define FALSE 0
#define MAXSIZE 2
#define INF 1e8

using namespace std;


#endif // ALL_HEADERS_H
```

6.1 Account(用户类)

6.1.1 account.h

```
#ifndef ACCOUNT_H
#define ACCOUNT_H
#include "all_headers.h"

class Account
{
public:
    static Account* get_account(){
        if(account==NULL) account=new Account("0","0","0","0");
        return account;
    }

    Account();
    Account(QString id,QString password,QString phone, QString address);

    //设置内容
    void renew_account(QString id,QString password,QString phone, QString
address);
    void renew_account(Account& user);
    void set_id(QString id);
    void set_password(QString password);
    void set_phone(QString phone);
    void set_address(QString address);


    //获取内容
    QString get_id();
    QString get_password();
    QString get_phone();
    QString get_address();
```

```
30
31
32  private:
33      QString id;
34      QString password;
35      QString phone;
36      QString address;
37      static Account* account;
38  };
39
40  #endif // ACCOUNT_H
41
```

6.1.2 account.cpp

```
1   #include "account.h"
2
3   Account::Account(){ }
4
5   Account::Account(QString id, QString password, QString phone, QString
    address)
6       :id(id),password(password),phone(phone),address(address){
7   }
8
9
10  //============================设置内容==============================
11
12
13  void Account::renew_account(QString id, QString password, QString phone,
    QString address)
14  {
15      set_id(id);
16      set_password(password);
17      set_phone(phone);
18      set_address(address);
19  }
20
21  void Account::renew_account(Account &user)
22  {
23      set_id(user.get_id());
24      set_password(user.get_password());
25      set_phone(user.get_phone());
26      set_address(user.get_password());
27  }
28
29  void Account::set_id(QString id){this->id=id;}
30  void Account::set_password(QString password){this->password=password;}
31  void Account::set_phone(QString phone){this->phone=phone;}
32  void Account::set_address(QString address){this->address=address;}
33
34  //============================获取内容==============================
35  QString Account::get_id(){return this->id;}
36  QString Account::get_password(){return this->password;}
37  QString Account::get_phone(){return this->phone;}
38  QString Account::get_address(){return this->address;}
```

```
39
```

6.2 Book(书籍类)

6.2.1 book.h

```
#ifndef BOOK_H
#define BOOK_H

#include "all_headers.h"

class Book
{
public:
    Book();

    Book(int id, QString ISBN, QString bookName, QString writer, QString publisher,QString owner, float price,  QString takeMethod, int status, int selled );

    //设置内容
    void renew_book(QString ISBN, QString bookName, QString writer, QString publisher,QString owner, float price,  QString takeMethod, int status, int selled);
    void set_isbn(QString isbn);
    void set_bookName(QString bookName);
    void set_writer(QString writer);
    void set_publisher(QString publisher);
    void set_takeMethod(QString takeMethod);
    void set_owner(QString owner);
    void set_price(float price);
    void set_status(int status);
    void set_selled(int selled);


    //获取内容
    int get_id();
    QString get_isbn();
    QString get_bookName();
    QString get_writer();
    QString get_publisher();
    QString get_takeMethod();
    QString get_owner();
    float get_price();
    int get_status();
    int get_selled();



private:
    int id;
    QString isbn;
    QString bookName;
    QString writer;
```

```
    QString publisher;
    QString owner;
    float price;
    QString takeMethod;
    int status;
    int selled;
};

#endif // BOOK_H
```

6.2.2 book.cpp

```
#include "book.h"

Book::Book(){}

Book::Book(int id, QString ISBN, QString bookName, QString writer, QString publisher,QString owner, float price,  QString takeMethod, int status, int selled)

:id(id),isbn(ISBN),bookName(bookName),writer(writer),publisher(publisher),owner(owner),price(price),takeMethod(takeMethod),status(status),selled(selled)
{
}



//============================设置内容==============================
void Book::renew_book(QString ISBN, QString bookName, QString writer, QString publisher,QString owner, float price,  QString takeMethod, int status, int selled){
    set_isbn(ISBN);
    set_bookName(bookName);
    set_writer(writer);
    set_publisher(publisher);
    set_takeMethod(takeMethod);
    set_owner(owner);
    set_price(price);
    set_status(status);
    set_selled(selled);
}
void Book::set_isbn(QString isbn){this->isbn=isbn;}
void Book::set_bookName(QString bookName){this->bookName=bookName;}
void Book::set_writer(QString writer){this->writer=writer;}
void Book::set_publisher(QString publisher){this->publisher=publisher;}
void Book::set_takeMethod(QString takeMethod){this->takeMethod=takeMethod;}
void Book::set_owner(QString owner){this->owner=owner;}
void Book::set_price(float price){this->price=price;}
void Book::set_status(int status){this->status=status;}
void Book::set_selled(int selled){this->selled=selled;}


//============================获取内容==============================
int Book::get_id(){return this->id;}
QString Book::get_isbn(){return this->isbn;}
```

```cpp
QString Book::get_bookName(){return this->bookName;}
QString Book::get_writer(){return this->writer;}
QString Book::get_publisher(){return this->publisher;}
QString Book::get_takeMethod(){return this->takeMethod;}
QString Book::get_owner(){return this->owner;}
float Book::get_price(){return this->price;}
int Book::get_status(){return this->status;}
int Book::get_selled(){return this->selled;}
```

6.3 Order(订单类)

6.3.1 order.h

```cpp
#pragma once
#ifndef ORDER_H
#define ORDER_H

#include "all_headers.h"
#include "book.h"
#include "account.h"
#include "dbmanager.h"

class Order
{
public:
    Order();
    //交易完成后，订单便不可再修改
    Order(QString sellerName,Book book);

    Account* get_seller();
    Book* get_book();
    QDateTime get_time();

private:
    Account* seller;
    Book book;
    QDateTime createTime;
};

#endif // ORDER_H

```

6.3.2 order.cpp

```cpp
#pragma once
#include "order.h"

Order::Order(){}

Order::Order(QString sellerName,Book book)
    :book(book){
    //设置卖方信息
```

```
    QString search=QString("select * from account"
                           " where id='%1'").arg(sellerName);
    DbManager* dba=DbManager::get_dba();
    QSqlQuery res(dba->db);
    res.exec(search);
    res.next();
    this->seller=new
Account(res.value("id").toString(),res.value("password").toString(),

res.value("phone").toString(),res.value("address").toString());
}

Account *Order::get_seller(){return this->seller;}
Book *Order::get_book(){return &(this->book);}
QDateTime Order::get_time(){return this->createTime.currentDateTime();}


```

6.4 DbManager(数据库连接类)

6.4.1 dbmanager.h

```
#pragma once
#ifndef DBMANAGER_H
#define DBMANAGER_H
#include "all_headers.h"

class DbManager
{
private:
    static DbManager* dba;

    DbManager();
    ~DbManager();
public:
    static DbManager* get_dba(){
        if(dba==NULL) dba=new DbManager();
        return dba;
    }
    QSqlDatabase db;
};

#endif // DBMANAGER_H

```

6.4.2 dbmanager.cpp

```
#pragma once
#include "dbmanager.h"


/*
```

```
函数名:DbManager
功能:构造函数。通过读取txt文件中的账号信息连接数据库
*/
DbManager::DbManager()
{
    db=QSqlDatabase::addDatabase("QODBC");

    QFile ifs("./database_user/accounts.txt");
    if(!ifs.open(QIODevice::ReadOnly | QIODevice::Text)){
        qDebug()<<"数据库账号文件打开失败";
    }

    ifs.readLine();//跳过第一行
    QString a=ifs.readLine();
    QStringList list=a.split(' ');    //空格作为分隔符拆开一行文本

    db.setHostName(list[0]);
    db.setDatabaseName(list[1]);
    db.setUserName(list[2]);
    db.setPassword(list[3]);
    db.setPort(list[4].toInt());

    if(!db.open()){
        QMessageBox::information(NULL,"DBA","数据库连接失败");
    }
    else {
        QMessageBox::information(NULL,"DBA","数据库连接成功");
    }

    ifs.close();
}
DbManager::~DbManager(){}
```

6.5 Login(登录界面类)

6.5.1 login.h

```
#pragma once
#ifndef LOGIN_H
#define LOGIN_H

#include "all_headers.h"
#include "register.h"
#include "buyerform.h"
#include "sellerform.h"
#include "dbmanager.h"
#include "account.h"

QT_BEGIN_NAMESPACE
namespace Ui { class Login; }
QT_END_NAMESPACE

class Login : public QMainWindow
{
```

```
    Q_OBJECT

public:
    Login(QWidget *parent = nullptr);
    ~Login();



private slots:
    //自定义控件事件
    void register_button_click();   //按下注册按钮
    void login_button_click();   //按下登录按钮

    //接收其他窗口发出的信号
    void receive_from_register();
    void receive_from_buyerOrseller();

signals://信号量只用声明不用实现
    void show_registerForm();
    void show_buyerForm();

private:
    Ui::Login *ui;
};
#endif // LOGIN_H
```

6.5.2 login.cpp

```
#pragma once
#include "login.h"
#include "ui_login.h"
#include "buyerform.h"


Login::Login(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::Login)
{
    ui->setupUi(this);

    //限制输入长度
    this->ui->accountTextbox->setMaxLength(40);
    this->ui->passwordTextbox->setMaxLength(20);

    this->ui->accountTextbox->setText("Mitchell");
    this->ui->passwordTextbox->setText("123456");

}

Login::~Login()
{
    delete ui;
}


```

```
//=========================================private slots=========================================
//----------------------------------------自定义控件事件----------------------------------------


/*
函数名：register_button_click()
功能：进入注册界面
*/
void Login::register_button_click()
{
    this->hide();
    emit show_registerForm();
}


/*
函数名：login_button_click()
功能：登录流程
*/
void Login::login_button_click()
{
    //存储输入的账号信息
    QString id=this->ui->accountTextbox->text();
    QString passwd=this->ui->passwordTextbox->text();
    if(id.isEmpty() || passwd.isEmpty()){
        QMessageBox::warning(NULL,"error","用户名或密码未输入");
        return ;
    }

    //匹配数据库中的账号
    QString search=QString("select * from account"
                           " where id='%1' and password ='%2'").arg(id).arg(passwd);
    DbManager* dba=DbManager::get_dba();
    QSqlQuery res(dba->db);
    res.exec(search);
    if(!res.next()){
        QMessageBox::warning(NULL,"error","账号或密码错误");
        return ;
    }

    //登录成功后，获取该用户的基本信息并初始化account对象
    QString userId=res.value("id").toString();
    QString userPassword=res.value("password").toString();
    QString userPhone=res.value("phone").toString();
    QString userAddress=res.value("address").toString();
    Account* user=Account::get_account();
    user->renew_account(userId,userPassword,userPhone,userAddress);
    qDebug()<<userId<<userPassword<<userPhone<<userAddress;


    this->hide();
    emit show_buyerForm();

}

```

```
83
84
85
86  //-----------------------------------------接收其他窗口发出的信号-----------
    ----------------------------
87
88  /*
89  函数名：receive_from_register()
90  功能：注册操作完成后返回登录界面
91  */
92  void Login::receive_from_register()
93  {
94      this->ui->accountTextbox->clear();
95      this->ui->passwordTextbox->clear();
96      this->show();
97  }
98
99
100 /*
101 函数名：receive_from_buyer()
102 功能：从买方或卖方界面退出
103 */
104 void Login::receive_from_buyerOrseller()
105 {
106     this->ui->accountTextbox->clear();
107     this->ui->passwordTextbox->clear();
108     this->show();
109 }
110
```

6.5.3 login.ui

6.6 Register(注册界面类)

6.6.1 register.h

```
#pragma once
#ifndef REGISTER_H
#define REGISTER_H

#include "all_headers.h"
#include "account.h"
#include "dbmanager.h"

namespace Ui {
class Register;
}

class Register : public QWidget
{
    Q_OBJECT

public:
    explicit Register(QWidget *parent = nullptr);
    ~Register();

private slots:
    //自定义控件事件
    void register_button_click();
    void cancel_button_click();

    //接收其他窗口发出的信号
    void receive_from_login();


signals:
    void show_loginForm();

private:
    Ui::Register *ui;
};

#endif // REGISTER_H

```

6.6.2 register.cpp

```
#pragma once
#include "register.h"
#include "ui_register.h"

Register::Register(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::Register)
{
    ui->setupUi(this);

    //限制输入长度
```

```
    this->ui->userNameTextbox->setMaxLength(35);
    this->ui->passwordTextbox->setMaxLength(20);
    this->ui->phoneTextbox->setMaxLength(20);
    this->ui->addressTextbox->setMaxLength(250);
}

Register::~Register()
{
    delete ui;
}




//=============================================private slots=============================================
//-----------------------------------------------自定义控件事件-------------------------------------------


/*
函数名：register_botton_click()
功能：注册一个账号并存入account数据表
*/

void Register::register_button_click()
{
    //获取输入框信息
    QString id=this->ui->userNameTextbox->text();
    QString password=this->ui->passwordTextbox->text();
    QString phone=this->ui->phoneTextbox->text();
    QString address=this->ui->addressTextbox->text();

    //输入检查
    if(id.isEmpty() || password.isEmpty() || phone.isEmpty()|| address.isEmpty()){
        QMessageBox::warning(NULL,"error","注册信息不完整");
        return ;
    }

    Account user(id,password,phone,address);
    //查询account表，判定用户名是否已存在
    DbManager* dba=DbManager::get_dba();
    QSqlQuery res(dba->db);
    QString sqlSentence=QString("select id from account where id='%1';").arg(id);
    res.exec(sqlSentence);
    if(res.next()){ //发现重名就弹窗报错
        QMessageBox::warning(NULL,"error","用户名已存在");
        return ;
    }

    //添加账号记录,弹出成功提示后返回登录界面
    sqlSentence=QString("insert into account values('%1','%2','%3','%4');").arg(id).arg(password).arg(phone).arg(address);
    res.exec(sqlSentence);
    QMessageBox::information(NULL,"success","账号已注册!");
    this->hide();
```

```
64      emit show_loginForm();
65  }
66
67  /*
68  函数名：cancel_botton_click()
69  功能：取消注册，回到登录界面
70  */
71  void Register::cancel_button_click()
72  {
73      this->hide();
74      emit show_loginForm();
75  }
76
77
78
79
80  //-----------------------------------------------接收其他窗口信号-----------------
    -------------------------
81
82  /*
83  函数名：receive_from_login()
84  功能：接收login界面点击注册按钮后发出的信号，并显示注册界面
85  */
86  void Register::receive_from_login()
87  {
88      this->show();
89  }
90
```

6.6.3 register.ui

Filter

对象	类
Register	QWidget
addressTextbox	QLineEdit
cancelButton	QPushButton
label	QLabel
label_3	QLabel
label_4	QLabel
label_5	QLabel
label_6	QLabel
passwordTextbox	QLineEdit
phoneTextbox	QLineEdit
registerButton	QPushButton
userNameTextbox	QLineEdit

6.7 BuyerForm(买方界面类)

6.7.1 buyerform.h

```h
#pragma once
#ifndef BUYERFORM_H
#define BUYERFORM_H

#include "all_headers.h"
#include "account.h"
#include "sellerform.h"
#include "dbmanager.h"
#include "passwordeditform.h"
#include "purchasedialog.h"

namespace Ui {
class BuyerForm;
}

class BuyerForm : public QMainWindow
{
    Q_OBJECT

public:
    //公用方法
    explicit BuyerForm(QWidget *parent = nullptr);
    ~BuyerForm();

    //Account& get_account();
    PasswordEditForm& get_passwordEditForm();
    void set_booksTableView();
    void set_ordersTableView();

private slots:
    //自定义控件事件
    void on_action_log_out_triggered();
    void goto_sellerForm_click();
    void on_actionchange_your_password_triggered();
    void search_selling_book_click();
    void search_target_book_click();
    void on_purchase_button_click();
    void on_search_buyerOrder_click();


    //接收其他窗口的信号
    void receive_from_login();
    void receive_from_seller();
    void receive_from_purchaseDialog(Order order);


signals:
    void show_loginForm();
    void show_sellerForm();

private:
    Ui::BuyerForm *ui;
    PasswordEditForm passwordEditForm;
    QSqlTableModel* model;
    QSqlTableModel* orderModel;
```

```
    PurchaseDialog* purchaseDialog;

};

#endif // BUYERFORM_H
```

6.7.2 buyerform.cpp

```
#pragma once
#include "buyerform.h"
#include "ui_buyerform.h"

BuyerForm::BuyerForm(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::BuyerForm)
{
    ui->setupUi(this);

    //预设置model
    DbManager* dba=DbManager::get_dba();
    this->model=new QSqlTableModel(this->ui->booksTableView,dba->db);
    this->orderModel=new QSqlTableModel(this->ui->ordersTableView,dba->db);
    //设置为手动提交数据库信息
    model->setEditStrategy(QSqlTableModel::OnManualSubmit);
    orderModel->setEditStrategy(QSqlTableModel::OnManualSubmit);
}

BuyerForm::~BuyerForm()
{
    delete ui;
}

//获取私有对象
PasswordEditForm &BuyerForm::get_passwordEditForm(){return this->passwordEditForm;}


/*
函数名：set_booksTableView()
功能：设置booksTableView的字段行
*/
void BuyerForm::set_booksTableView()
{
    model->setHeaderData(0,Qt::Horizontal,QObject::tr("ID"));
    model->setHeaderData(1,Qt::Horizontal,QObject::tr("ISBN"));
    model->setHeaderData(2,Qt::Horizontal,QObject::tr("书名"));
    model->setHeaderData(3,Qt::Horizontal,QObject::tr("作者"));
    model->setHeaderData(4,Qt::Horizontal,QObject::tr("出版社"));
    model->setHeaderData(5,Qt::Horizontal,QObject::tr("卖方"));
    model->setHeaderData(6,Qt::Horizontal,QObject::tr("获取方式"));
    model->setHeaderData(7,Qt::Horizontal,QObject::tr("价格"));
    model->setHeaderData(8,Qt::Horizontal,QObject::tr("新旧程度"));
    model->setHeaderData(9,Qt::Horizontal,QObject::tr("售出状态"));
}

```

```
47
48  /*
49  函数名：set_ordersTableView()
50  功能：设置ordersTableView的字段行
51  */
52  void BuyerForm::set_ordersTableView()
53  {
54      orderModel->setHeaderData(0,Qt::Horizontal,QObject::tr("ID"));
55      orderModel->setHeaderData(1,Qt::Horizontal,QObject::tr("买方"));
56      orderModel->setHeaderData(2,Qt::Horizontal,QObject::tr("买方电话"));
57      orderModel->setHeaderData(3,Qt::Horizontal,QObject::tr("买方地址"));
58      orderModel->setHeaderData(4,Qt::Horizontal,QObject::tr("卖方"));
59      orderModel->setHeaderData(5,Qt::Horizontal,QObject::tr("卖方电话"));
60      orderModel->setHeaderData(6,Qt::Horizontal,QObject::tr("卖方地址"));
61      orderModel->setHeaderData(7,Qt::Horizontal,QObject::tr("书名"));
62      orderModel->setHeaderData(8,Qt::Horizontal,QObject::tr("价格"));
63      orderModel->setHeaderData(9,Qt::Horizontal,QObject::tr("购买时间"));
64
65  }
66
67
68
69  //================================================private
    slots=====================================================
70
71  //-----------------------------------------------自定义控件事件--------------
    --------------------------------
72
73  /*
74  函数名：on_action_log_out_triggered()
75  功能：退出当前用户
76  */
77  void BuyerForm::on_action_log_out_triggered()
78  {
79      this->close();
80      emit show_loginForm();
81  }
82
83
84  /*
85  函数名：goto_sellerForm_click()
86  功能：当前用户转到卖方界面
87  */
88  void BuyerForm::goto_sellerForm_click()
89  {
90      this->hide();
91      emit show_sellerForm();
92  }
93
94  /*
95  函数名：on_actionchange_your_password_triggered()
96  功能：弹出修改密码界面
97  */
98  void BuyerForm::on_actionchange_your_password_triggered()
99  {
100     this->passwordEditForm.clear_lineEdit();
101     this->passwordEditForm.show();
102 }
```

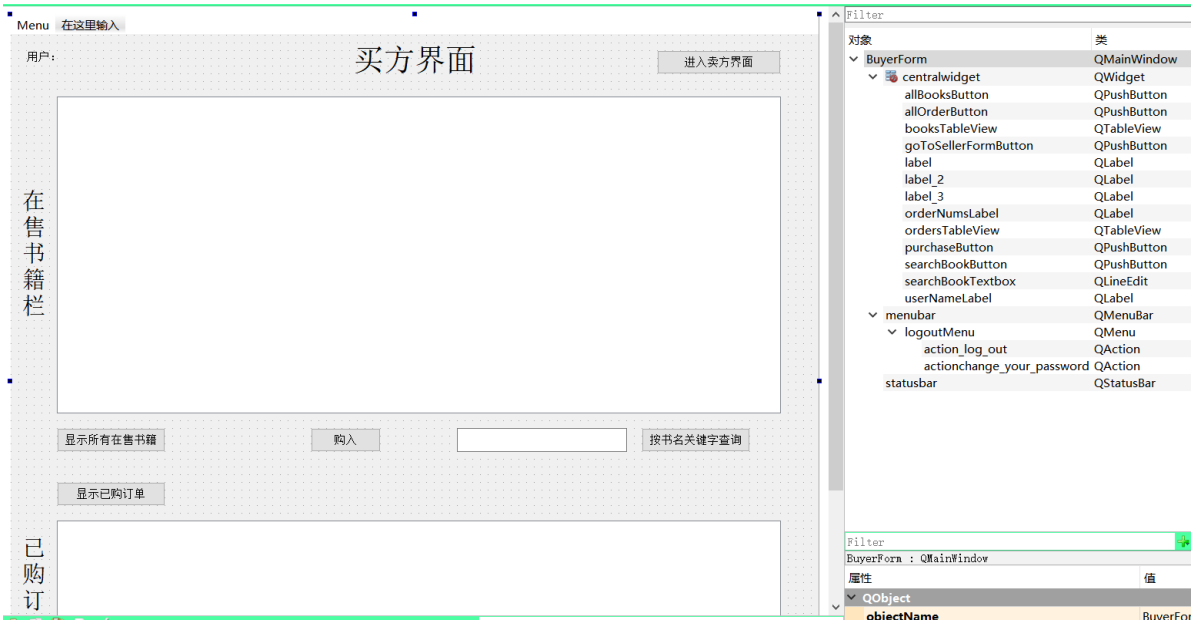
```
103
104
105 /*
106 函数名：search_selling_book_click()
107 功能：查询所有在售书籍
108 */
109 void BuyerForm::search_selling_book_click()
110 {
111     model->setTable("book");
112     Account* account=Account::get_account();
113
114     //筛选条件
115     model->setFilter(QString("owner!='%1' and selled=0").arg(account-
>get_id()));
116     model->select();
117
118     //设置表头
119     this->set_booksTableView();
120
121     this->ui->booksTableView->setModel(model);
122     //table界面设为不可编辑
123     this->ui->booksTableView-
>setEditTriggers(QAbstractItemView::NoEditTriggers);
124     if(model->rowCount()==0){
125         QMessageBox::information(NULL,"tip","暂无在售书籍");
126     }
127 }
128
129
130 /*
131 函数名：search_target_book_click()
132 功能：按关键字搜索指定书籍
133 */
134 void BuyerForm::search_target_book_click()
135 {
136
137     QString target=this->ui->searchBookTextbox->text();
138     Account* account=Account::get_account();
139
140     //筛选条件
141     model->setTable("book");
142     model->setFilter(QString("owner!='%1' and bookName like '%"+target+"%'
;").arg(account->get_id()));//.arg(target)); //筛选条件
143     model->select();
144
145     this->set_booksTableView();
146
147     this->ui->booksTableView->setModel(model);
148     //table界面设为不可编辑
149     this->ui->booksTableView-
>setEditTriggers(QAbstractItemView::NoEditTriggers);
150     if(model->rowCount()==0){
151         QMessageBox::information(NULL,"tip","没有相关在售书籍");
152     }
153 }
154
155
156
```

```
/*
函数名：on_purchase_button_click()
功能：打开买书界面
*/
void BuyerForm::on_purchase_button_click()
{
    int curRow=this->ui->booksTableView->currentIndex().row();
    //判断用户是否有选中一条记录，-1代表未选择记录
    if(curRow==-1){
        QMessageBox::warning(NULL,"error","请选择一条书籍记录");
        return ;
    }

    //与购买对话框建立连接
    this->purchaseDialog=new PurchaseDialog(this);

 QObject::connect(purchaseDialog,SIGNAL(purchase_confirm(Order)),this,SLOT(receive_from_purchaseDialog(Order)));

    //获取选中的记录
    QSqlRecord record=this->model->record(curRow);

    //书籍基本信息
    int id=record.value(0).toInt();
    QString isbn=record.value(1).toString();
    QString bookName=record.value(2).toString();
    QString writer=record.value(3).toString();
    QString publisher=record.value(4).toString();
    QString sellerName=record.value(5).toString();
    QString takeMethod=record.value(6).toString();
    float price=record.value(7).toFloat();
    int status=record.value(8).toInt();
    int selled=record.value(9).toInt();
    qDebug()<<price<<status<<selled;
    Account* account=Account::get_account();

    Book book(id,isbn,bookName,writer,publisher,account->get_id(),price,takeMethod,status,selled);

    //设置购买对话框的内容
    this->purchaseDialog->set_ui(sellerName,book);
    this->purchaseDialog->show();
}


/*
函数名：on_search_buyerOrder_click()
功能：查询当前账号作为买方的订单
*/
void BuyerForm::on_search_buyerOrder_click()
{
    orderModel->setTable("detailorder");
    Account* account=Account::get_account();
    //筛选条件
    orderModel->setFilter(QString("buyerName='%1'").arg(account->get_id()));
    orderModel->select();

```

```
211     //设置订单栏的表头
212     this->set_ordersTableView();
213 
214     this->ui->ordersTableView->setModel(orderModel);
215     this->ui->ordersTableView-
>setEditTriggers(QAbstractItemView::NoEditTriggers);
216     this->ui->orderNumsLabel->setText(QString("订单总数：
%1").arg(orderModel->rowCount()));
217     if(orderModel->rowCount()==0){
218         QMessageBox::information(NULL,"tip","该用户暂无购入订单");
219     }
220 }
221 
222 
223 
224 //-----------------------------------------接收其他窗口的信号----------------
----------------------------
225 
226 /*
227 函数名：receive_from_login()
228 功能：完成登录事件后进入买方界面，并清空买方界面中tableview的内容
229 */
230 void BuyerForm::receive_from_login()
231 {
232     //清空两个tableview的内容
233     QSqlQueryModel* res=new QSqlQueryModel(this->ui->booksTableView);
234     res->clear();
235     this->ui->booksTableView->setModel(res);
236     this->ui->ordersTableView->setModel(res);
237 
238     Account* account=Account::get_account();
239     this->ui->userNameLabel->setText(QString("用户：%1").arg(account-
>get_id()));
240     this->ui->orderNumsLabel->setText(QString("订单总数："));
241     this->show();
242 }
243 
244 /*
245 函数名：receive_from_seller()
246 功能：从卖方界面切换到买方界面，并清空买方界面中tableview的内容
247 */
248 void BuyerForm::receive_from_seller()
249 {
250     //清空两个tableview的内容
251     QSqlQueryModel* res=new QSqlQueryModel(this->ui->booksTableView);
252     res->clear();
253     this->ui->booksTableView->setModel(res);
254     this->ui->ordersTableView->setModel(res);
255     this->ui->orderNumsLabel->setText(QString("订单总数："));
256     this->show();
257 }
258 
259 
260 /*
261 函数名：receive_from_purchaseDialog()
262 功能：完成购买流程
263 */
264 void BuyerForm::receive_from_purchaseDialog(Order order)
```

```
{
    orderModel->setTable("bookorder");
    orderModel->select();

    //设置订单记录
    QSqlRecord record=orderModel->record();
    Account* account=Account::get_account();
    record.setValue(1,account->get_id());
    record.setValue(2,order.get_seller()->get_id());
    record.setValue(3,order.get_book()->get_bookName());
    record.setValue(4,QString("%1").arg(order.get_book()->get_price()));
    record.setValue(5,order.get_time());

    if(orderModel->insertRecord(orderModel->rowCount(),record)){
        QMessageBox::information(NULL,"success","购买成功!");

        //修改书籍拥有者，并将书籍售出状态设置为1(已售出)
        int curRow=this->ui->booksTableView->currentIndex().row();
        QSqlRecord purchasedBook=model->record(curRow);
        purchasedBook.setValue(5,account->get_id());
        purchasedBook.setValue(9,1);
        model->setRecord(curRow,purchasedBook);

        //提交所有修改信息
        model->submitAll();
        orderModel->submitAll();
        this->purchaseDialog->close();
        this->on_search_buyerOrder_click();
    }
    else{
        QMessageBox::warning(NULL,"error","购买失败。");
        this->purchaseDialog->close();
    }
}

```

6.7.3 buyerform.ui



6.8 SellerForm(卖方界面类)

6.8.1 sellerform.h

```
#pragma once
#ifndef SELLERFORM_H
#define SELLERFORM_H

#include "all_headers.h"
#include "account.h"
#include "dbmanager.h"
#include "passwordeditform.h"
#include "insertdialog.h"
#include "editdialog.h"
#include "book.h"

namespace Ui {
class SellerForm;
}

class SellerForm : public QMainWindow
{
    Q_OBJECT

public:
    explicit SellerForm(QWidget *parent = nullptr);
    ~SellerForm();

    PasswordEditForm &get_passwordEditForm();
    void set_booksTableView();
    void set_ordersTableView();

private slots:
    //自定义控件事件
    void on_action_log_out_triggered();
    void goto_buyerForm_click();
    void on_actionchange_your_password_triggered();
    void get_my_book_click();
    void search_target_book_click();
    void on_edit_button_clicked();
    void on_insert_button_clicked();
    void on_delete_button_clicked();
    void on_search_sellerOrder_click();

    //接收其他窗口的信号
    void receive_from_buyer();
    void receive_from_edit_confirm(Book book);
    void receive_from_insert_confirm(Book book);

signals:
    void show_buyerForm();
    void show_loginForm();

private:
    Ui::SellerForm *ui;
    PasswordEditForm passwordEditForm;
    QSqlTableModel* model;
```

```
    QSqlTableModel* orderModel;
    EditDialog* editDialog;
    InsertDialog* insertDialog;
};

#endif // SELLERFORM_H
```

6.8.2 sellerform.cpp

```
#pragma once
#include "sellerform.h"
#include "ui_sellerform.h"

SellerForm::SellerForm(QWidget *parent) :
    QMainWindow(parent),
    ui(new Ui::SellerForm)
{
    ui->setupUi(this);

    //预设置model
    DbManager* dba=DbManager::get_dba();
    model=new QSqlTableModel(this->ui->booksTableView,dba->db);
    this->orderModel=new QSqlTableModel(this->ui->ordersTableView,dba->db);
    model->setEditStrategy(QSqlTableModel::OnManualSubmit);
    orderModel->setEditStrategy(QSqlTableModel::OnManualSubmit);
}

SellerForm::~SellerForm()
{
    delete ui;
}

//获取私有对象
PasswordEditForm &SellerForm::get_passwordEditForm(){return this->passwordEditForm;}



/*
函数名：set_booksTableView()
功能：设置booksTableView的字段行
*/
void SellerForm::set_booksTableView()
{
    model->setHeaderData(0,Qt::Horizontal,QObject::tr("ID"));
    model->setHeaderData(1,Qt::Horizontal,QObject::tr("ISBN"));
    model->setHeaderData(2,Qt::Horizontal,QObject::tr("书名"));
    model->setHeaderData(3,Qt::Horizontal,QObject::tr("作者"));
    model->setHeaderData(4,Qt::Horizontal,QObject::tr("出版社"));
    model->setHeaderData(5,Qt::Horizontal,QObject::tr("卖方"));
    model->setHeaderData(6,Qt::Horizontal,QObject::tr("获取方式"));
    model->setHeaderData(7,Qt::Horizontal,QObject::tr("价格"));
    model->setHeaderData(8,Qt::Horizontal,QObject::tr("新旧程度"));
    model->setHeaderData(9,Qt::Horizontal,QObject::tr("售出状态"));
}
```

```
46
47  /*
48  函数名：set_ordersTableView()
49  功能：设置ordersTableView的字段行
50  */
51  void SellerForm::set_ordersTableView()
52  {
53      orderModel->setHeaderData(0,Qt::Horizontal,QObject::tr("ID"));
54      orderModel->setHeaderData(1,Qt::Horizontal,QObject::tr("买方"));
55      orderModel->setHeaderData(2,Qt::Horizontal,QObject::tr("买方电话"));
56      orderModel->setHeaderData(3,Qt::Horizontal,QObject::tr("买方地址"));
57      orderModel->setHeaderData(4,Qt::Horizontal,QObject::tr("卖方"));
58      orderModel->setHeaderData(5,Qt::Horizontal,QObject::tr("卖方电话"));
59      orderModel->setHeaderData(6,Qt::Horizontal,QObject::tr("卖方地址"));
60      orderModel->setHeaderData(7,Qt::Horizontal,QObject::tr("书名"));
61      orderModel->setHeaderData(8,Qt::Horizontal,QObject::tr("价格"));
62      orderModel->setHeaderData(9,Qt::Horizontal,QObject::tr("购买时间"));
63  }
64
65
66
67
68  //=========================================private
    slots=====================================================
69  //----------------------------------------自定义控件事件--------------------
    ------------------------------------
70
71  /*
72  函数名：on_action_log_out_triggered()
73  功能：退出当前账号
74  */
75  void SellerForm::on_action_log_out_triggered()
76  {
77      this->close();
78      emit show_loginForm();
79  }
80
81
82  /*
83  函数名：goto_buyerForm_click()
84  功能：切换到买方界面
85  */
86  void SellerForm::goto_buyerForm_click()
87  {
88      this->close();
89      emit show_buyerForm();
90  }
91
92
93  /*
94  函数名：on_actionchange_your_password_triggered()
95  功能：弹出修改密码界面
96  */
97  void SellerForm::on_actionchange_your_password_triggered()
98  {
99      this->passwordEditForm.clear_lineEdit();
100     this->passwordEditForm.show();
101 }
```

```
102
103
104 /*
105 函数名：get_my_book_click()
106 功能：查询自己的书
107 */
108 void SellerForm::get_my_book_click()
109 {
110     model->setTable("book");
111     Account* account=Account::get_account();
112
113     //筛选条件
114     model->setFilter(QString("owner='%1'").arg(account->get_id()));
115     model->select();
116
117     //设置书籍栏的表头
118     this->set_booksTableView();
119
120     this->ui->booksTableView->setModel(model);
121     this->ui->booksTableView->setEditTriggers(QAbstractItemView::NoEditTriggers);   //table界面设为不可编辑
122     if(model->rowCount()==0){
123         QMessageBox::information(NULL,"tip","你的书库为空");
124     }
125 }
126
127
128 /*
129 函数名：search_target_book_click()
130 功能：搜索指定书籍
131 */
132 void SellerForm::search_target_book_click()
133 {
134
135     QString target=this->ui->searchBookTextbox->text();
136     Account* account=Account::get_account();
137     model->setTable("book");
138
139     //筛选条件
140     model->setFilter(QString("owner='%1' and bookName like '%"+target+"%';").arg(account->get_id())); //筛选条件
141     model->select();
142
143     //设置书籍栏表头
144     this->set_booksTableView();
145     this->ui->booksTableView->setModel(model);
146
147     if(model->rowCount()==0){
148         QMessageBox::information(NULL,"tip","你的书库没有相关书籍");
149     }
150 }
151
152
153 /*
154 函数名：on_edit_botton_clicked()
155 功能：修改书籍信息，显示书籍信息编辑界面
156 */
```

```
157 void SellerForm::on_edit_button_clicked()
158 {
159     int curRow=this->ui->booksTableView->currentIndex().row();
160     if(curRow==-1){
161         QMessageBox::warning(NULL,"error","请选择一条书籍记录");
162         return ;
163     }
164     QSqlRecord record=this->model->record(curRow);
165 
166     //建立卖书界面和书籍编辑界面的连接
167     this->editDialog=new EditDialog(this);
168 
    QObject::connect(editDialog,SIGNAL(book_edit_confirm(Book)),this,SLOT(rece
    ive_from_edit_confirm(Book)));
169 
170 
171     //书籍基本信息
172     int id=record.value(0).toInt();
173     QString isbn=record.value(1).toString();
174     QString bookName=record.value(2).toString();
175     QString writer=record.value(3).toString();
176     QString publisher=record.value(4).toString();
177     float price=record.value(7).toFloat();
178     QString takeMethod=record.value(6).toString();
179     int status=record.value(8).toInt();
180     int selled=record.value(9).toInt();
181     qDebug()<<price<<status<<selled;
182     Account* account=Account::get_account();
183 
184 
185     Book book(id,isbn,bookName,writer,publisher,account-
    >get_id(),price,takeMethod,status,selled);
186 
187 
188     //设置书籍编辑界面的基本信息
189     this->editDialog->set_ui(book);
190     this->editDialog->show();
191 }
192 
193 
194 /*
195 函数名：on_insert_botton_clicked()
196 功能：添加一本书，显示书籍信息添加界面
197 */
198 void SellerForm::on_insert_button_clicked()
199 {
200     //建立卖书界面和添加书籍界面的连接
201     this->insertDialog=new InsertDialog(this);
202 
    QObject::connect(insertDialog,SIGNAL(book_insert_confirm(Book)),this,SLOT(
    receive_from_insert_confirm(Book)));
203     this->insertDialog->show();
204 }
205 
206 /*
207 函数名：on_delete_botton_clicked()
208 功能：删除选中的一本书
209 */
```

```cpp
void SellerForm::on_delete_button_clicked()
{
    int curRow=this->ui->booksTableView->currentIndex().row();
    if(curRow==-1){
        QMessageBox::warning(NULL,"error","请选择一条书籍记录");
        return ;
    }

    this->model->removeRow(curRow);

    //确认删除
    int ok=QMessageBox::warning(this,"删除当前行","确定删除这本书的记录吗？",
                                QMessageBox::Yes,QMessageBox::No);
    if(ok==QMessageBox::No){
        this->model->revertAll();
        QMessageBox::information(NULL,"revert","已取消");
        return ;
    }
    else{
        this->model->submitAll();
        QMessageBox::information(NULL,"success","相关书本已删除");
    }
}


/*
函数名：on_search_sellerOrder_click()
功能：查询当前账号作为卖方的订单
*/
void SellerForm::on_search_sellerOrder_click()
{
    orderModel->setTable("detailorder");
    Account* account=Account::get_account();

    //筛选条件
    orderModel->setFilter(QString("sellerName='%1'").arg(account->get_id()));
    orderModel->select();

    //设置订单栏表头
    this->set_ordersTableView();

    this->ui->ordersTableView->setModel(orderModel);
    this->ui->ordersTableView->setEditTriggers(QAbstractItemView::NoEditTriggers);
    this->ui->orderNumsLabel->setText(QString("订单总数：%1").arg(orderModel->rowCount()));
    if(orderModel->rowCount()==0){
        QMessageBox::information(NULL,"tip","该用户暂无售出订单");
    }
}



//----------------------------------------------接收其他窗口的信号----------------------------

/*
```

```
函数名：receive_from_buyer()
功能：接收从买方界面切换到卖方界面发出的信号，并清空卖方界面中tableview的所有内容
*/
void SellerForm::receive_from_buyer()
{
    //清空两个tableview的内容
    QSqlQueryModel* res=new QSqlQueryModel(this->ui->booksTableView);
    res->clear();
    this->ui->booksTableView->setModel(res);
    this->ui->ordersTableView->setModel(res);

    Account* account=Account::get_account();
    this->ui->userNameLabel->setText(QString("用户：%1").arg(account->get_id()));
    this->ui->orderNumsLabel->setText(QString("订单总数："));
    this->show();
}


/*
函数名：receive_from_edit_confirm(Book book)
功能：接收从修改书籍界面传来的书籍对象信号，并更新数据库
*/
void SellerForm::receive_from_edit_confirm(Book book)
{
    //设置待更新的书籍记录
    int curRow=this->ui->booksTableView->currentIndex().row();
    QSqlRecord record=this->model->record(curRow);
    record.setValue(1,book.get_isbn());
    record.setValue(2,book.get_bookName());
    record.setValue(3,book.get_writer());
    record.setValue(4,book.get_publisher());
    record.setValue(6,book.get_takeMethod());
    record.setValue(7,QString("%1").arg(book.get_price()));
    record.setValue(8,book.get_status());
    record.setValue(9,book.get_selled());


    //提交要更新的记录
    if(model->setRecord(curRow,record)){
        model->submitAll();
        QMessageBox::information(NULL,"success","记录修改成功");
        this->editDialog->close();
    }
    else{
        QMessageBox::warning(NULL,"error","书籍信息修改失败");
    }
}


/*
函数名：receive_from_insert_confirm(Book book)
功能：接收从添加书籍界面传来的书籍对象信号，并插入数据库
*/
void SellerForm::receive_from_insert_confirm(Book book)
{
    //设置要插入的记录
    QSqlRecord record=model->record();
```

```
321
322     record.setValue(1,book.get_isbn());
323     record.setValue(2,book.get_bookName());
324     record.setValue(3,book.get_writer());
325     record.setValue(4,book.get_publisher());
326     record.setValue(5,book.get_owner());
327     record.setValue(6,book.get_takeMethod());
328     record.setValue(7,QString("%1").arg(book.get_price()));
329     record.setValue(8,book.get_status());
330     record.setValue(9,book.get_selled());
331
332     //插入记录
333     if(model->insertRecord(model->rowCount(),record)){
334         QMessageBox::information(NULL,"success","书本添加成功");
335         model->submitAll();
336         this->insertDialog->close();
337     }
338     else{
339         QMessageBox::warning(NULL,"error","书本添加失败");
340     }
341
342 }
343
```

6.8.3 sellerform.ui

6.9 PasswordEditForm(修改密码界面类)

6.9.1 passwordeditdorm.h

```
1 #pragma once
2 #ifndef PASSWORDEDITFORM_H
3 #define PASSWORDEDITFORM_H
4
5 #include "account.h"
6 #include "dbmanager.h"
```

```
#include "order.h"
#include <QWidget>

namespace Ui {
class PasswordEditForm;
}

class PasswordEditForm : public QWidget
{
    Q_OBJECT

public:
    explicit PasswordEditForm(QWidget *parent = nullptr);
    ~PasswordEditForm();
    void clear_lineEdit();

private slots:
    void password_edit_click();
    void cancel_click();

signals:
    void renew_account();

private:
    Ui::PasswordEditForm *ui;
};

#endif // PASSWORDEDITFORM_H

```

6.9.2 passwordeditform.cpp

```
#include "passwordeditform.h"
#include "ui_passwordeditform.h"

PasswordEditForm::PasswordEditForm(QWidget *parent) :
    QWidget(parent),
    ui(new Ui::PasswordEditForm)
{
    ui->setupUi(this);

    //限制输入长度
    this->ui->originalTextbox->setMaxLength(20);
    this->ui->newTextbox->setMaxLength(20);
    this->ui->newConfirmTextbox->setMaxLength(20);
}

PasswordEditForm::~PasswordEditForm()
{
    delete ui;
}


/*
函数名：clear_lineEdit()
```

```
功能：清空界面中的编辑框内容
*/
void PasswordEditForm::clear_lineEdit()
{
    this->ui->originalTextbox->clear();
    this->ui->newTextbox->clear();
    this->ui->newConfirmTextbox->clear();
}


//=============================================private slots=============================================
//-----------------------------------------------自定义控件事件-------------------------


/*
函数名：password_edit_click()
功能：确认修改密码
*/
void PasswordEditForm::password_edit_click()
{
    QString original=this->ui->originalTextbox->text();
    QString newPasswd=this->ui->newTextbox->text();
    QString newConfirmPasswd=this->ui->newConfirmTextbox->text();

    Account* account=Account::get_account();

    //信息栏是否完善
    if(original.isEmpty() || newPasswd.isEmpty() || newConfirmPasswd.isEmpty()){
        QMessageBox::warning(NULL,"error","修改信息不完善");
        return ;
    }
    //原密码是否正确
    if(original!=account->get_password()){
        QMessageBox::warning(NULL,"error","原密码不正确");
        return ;
    }
    //新密码与原密码是否相同
    if(newPasswd==account->get_password()){
        QMessageBox::warning(NULL,"error","新密码与原密码相同");
        return ;
    }
    //新密码与确认密码是否相同
    if(newPasswd!=newConfirmPasswd){
        QMessageBox::warning(NULL,"error","新密码与确认密码不吻合");
        return ;
    }


    //修改信息
    DbManager* dba=DbManager::get_dba();
    QSqlQuery res(dba->db);

    QString sqlSentence=QString("update account set password='%1' where id='%2'").arg(newPasswd).arg(account->get_id());
    res.exec(sqlSentence);
```

```
78      QMessageBox::information(NULL,"success","密码修改成功!");
79      this->close();
80      account->set_password(newPasswd);
81      //emit renew_account();
82  }
83
84
85  /*
86  函数名：cancel_click()
87  功能：取消修改密码
88  */
89  void PasswordEditForm::cancel_click()
90  {
91      this->close();
92  }
```

6.9.3 passwordeditform.ui

6.10 PurchaseDialog(购买订单对话框类)

6.10.1 purchasedialog.h

```
1   #ifndef PURCHASEDIALOG_H
2   #define PURCHASEDIALOG_H
3
4   #include "all_headers.h"
5   #include "order.h"
6   #include "account.h"
7   #include <QDialog>
8
9   namespace Ui {
10  class PurchaseDialog;
11  }
12
13  class PurchaseDialog : public QDialog
14  {
15      Q_OBJECT
```

```
16
17  public:
18      explicit PurchaseDialog(QWidget *parent = nullptr);
19      ~PurchaseDialog();
20
21      void set_ui(QString sellerName,Book book);
22
23  private slots:
24      //自定义控件事件
25      void confirmButton_click();
26      void cancelButton_click();
27
28  signals:
29      void purchase_confirm(Order order);
30
31
32  private:
33      Ui::PurchaseDialog *ui;
34      Order* order;
35  };
36
37  #endif // PURCHASEDIALOG_H
```

6.10.2 purchasedialog.cpp

```
1   #include "purchasedialog.h"
2   #include "ui_purchasedialog.h"
3
4   PurchaseDialog::PurchaseDialog(QWidget *parent) :
5       QDialog(parent),
6       ui(new Ui::PurchaseDialog)
7   {
8       ui->setupUi(this);
9   }
10
11  PurchaseDialog::~PurchaseDialog()
12  {
13      delete ui;
14  }
15
16  /*
17  函数名：set_ui(QString sellerName, Book book)
18  功能：设置订单界面内容
19  */
20  void PurchaseDialog::set_ui(QString sellerName, Book book)
21  {
22      this->order=new Order(sellerName,book);
23      Account* buyer=Account::get_account();
24      Account* seller=this->order->get_seller();
25      //设置订单界面内容
26      this->ui->buyerNameLabel->setText(buyer->get_id());
27      this->ui->buyerPhoneLabel->setText(buyer->get_phone());
28      this->ui->buyerAddressLabel->setText(buyer->get_address());
29      this->ui->sellerNameLabel->setText(seller->get_id());
30      this->ui->sellerPhoneLabel->setText(seller->get_phone());
```

```
    this->ui->sellerAddressLabel->setText(seller->get_address());

    this->ui->bookNameLabel->setText(book.get_bookName());
    this->ui->takeMethodLabel->setText(book.get_takeMethod());
    this->ui->priceLabel->setText(QString("%1").arg(book.get_price()));
    this->ui->statusLabel->setText(QString("%1").arg(book.get_status()));
}

//=============================================private slots=============================================
//-----------------------------------------------自定义控件事件-----------------------------------------


/*
函数名：confirmButton_click()
功能：确认购买
*/
void PurchaseDialog::confirmButton_click()
{
    emit purchase_confirm(*this->order);
}

/*
函数名：cancelButton_click()
功能：取消购买
*/
void PurchaseDialog::cancelButton_click()
{
    QMessageBox::information(NULL,"revert","订单已取消");
    this->close();
}
```

6.10.3 purchasedialog.ui

6.11 InsertDialog(插入书籍信息对话框类)

6.11.1 insertdialog.h

```
#pragma once
#ifndef INSERTDIALOG_H
#define INSERTDIALOG_H

#include "all_headers.h"
#include "book.h"
#include "account.h"
#include <QDialog>


namespace Ui {
class InsertDialog;
}

class InsertDialog : public QDialog
{
    Q_OBJECT

public:
    explicit InsertDialog(QWidget *parent = nullptr);
    ~InsertDialog();

private slots:
    //自定义控件事件
    void cancelButton_click();
    void confirmButton_click();


signals:
    void book_insert_confirm(Book book);

```

```
32
33  private:
34      Ui::InsertDialog *ui;
35  };
36
37  #endif // INSERTDIALOG_H
38
```

6.11.2 insertdialog.cpp

```
1   #pragma once
2   #include "insertdialog.h"
3   #include "ui_insertdialog.h"
4
5   InsertDialog::InsertDialog(QWidget *parent) :
6       QDialog(parent),
7       ui(new Ui::InsertDialog)
8   {
9       ui->setupUi(this);
10
11      //限制输入长度
12      this->ui->isbnEdit->setMaxLength(60);
13      this->ui->bookNameEdit->setMaxLength(250);
14      this->ui->writerEdit->setMaxLength(40);
15      this->ui->publisherEdit->setMaxLength(40);
16      this->ui->priceEdit->setMaxLength(7);
17      this->ui->takeMethodEdit->setMaxLength(250);
18  }
19
20  InsertDialog::~InsertDialog()
21  {
22      delete ui;
23  }
24
25  //==========================================private
    slots=====================================================
26  //----------------------------------------自定义控件事件--------------------
    ------------------------------------
27
28
29  /*
30  函数名：cancelButton_click()
31  功能：取消添加记录流程
32  */
33
34  void InsertDialog::cancelButton_click()
35  {
36      this->close();
37  }
38
39
40  /*
41  函数名：confirmButton_click()
42  功能：完成编辑，并将编辑完成的对象发送给卖方界面
43  */
```

```cpp
void InsertDialog::confirmButton_click()
{
    //获取初始化book对象所需的内容
    QString isbn=this->ui->isbnEdit->text();
    QString bookName=this->ui->bookNameEdit->text();
    QString writer=this->ui->writerEdit->text();
    QString publisher=this->ui->publisherEdit->text();
    QString takeMethod=this->ui->takeMethodEdit->text();
    QString textPrice=this->ui->priceEdit->text();
    int status=this->ui->newStatusComboBox->currentText().toInt();
    int selled=this->ui->sellingStatuscomboBox->currentIndex();


    //判断输入信息是否完整
    if(isbn.isEmpty() || bookName.isEmpty() || writer.isEmpty() ||
            publisher.isEmpty() || takeMethod.isEmpty() ||
QString("%1").arg(textPrice)<=0){
        QMessageBox::warning(NULL,"error","存在未填信息.(也可能是书本价格为0)");
        return ;
    }


    int pos=0;
    QValidator::State res;

    //正则匹配ISBN信息
    QRegExp isbnFormula("^978-7-[0-9]{2,6}-[0-9]{2,6}-[0-9]$");
    QRegExpValidator isbnCheck(isbnFormula,0);
    pos=0;
    res=isbnCheck.validate(isbn,pos);
    if(QValidator::Acceptable==res){
    }
    else{
        QMessageBox::warning(NULL,"error","isbn值必须形如“978-7-04-051102-
4”\n"
                                          "中国的ISBN编号共13位,978为固定,其他
十位,"
                                          "第一位为中国的标识7,之后是标识出版社的
2-6位数字,"
                                          "之后是出版社内刊物的流水,最后一位是校
验位0-9或X");
        return ;
    }

    //正则匹配书籍价格信息
    QRegExp priceFormula("\\d+\\.\\d+|(\\d+)");
    QRegExpValidator priceCheck(priceFormula,0);
    res=priceCheck.validate(textPrice,pos);
    if(QValidator::Acceptable==res){
        qDebug()<<1;
    }
    else{
        QMessageBox::warning(NULL,"error","书本价格必须是大于0的数值");
        return ;
    }

    float price=this->ui->priceEdit->text().toFloat();

```

```
    Account* account=Account::get_account();

    Book book(0,isbn,bookName,writer,publisher,account->get_id(),price,takeMethod,status,selled);
    emit book_insert_confirm(book);
}

```

6.11.3 insertdialog.ui

Filter

对象	类
InsertDialog	QDialog
bookNameEdit	QLineEdit
cancelButton	QPushButton
confirmButton	QPushButton
isbnEdit	QLineEdit
isbnLabel	QLabel
label	QLabel
label_2	QLabel
label_3	QLabel
label_4	QLabel
label_5	QLabel
label_6	QLabel
label_7	QLabel
label_8	QLabel
label_9	QLabel
newStatusComboBox	QComboBox
priceEdit	QLineEdit
publisherEdit	QLineEdit
sellingStatuscomboBox	QComboBox
takeMethodEdit	QLineEdit
titleLabel	QLabel
writerEdit	QLineEdit

6.12 EditDialog(修改书籍信息对话框类)

6.12.1 editdialog.h

```
#pragma once
#ifndef EDITDIALOG_H
#define EDITDIALOG_H

#include "all_headers.h"
#include "book.h"
#include "account.h"
#include <QDialog>

namespace Ui {
class EditDialog;
}

class EditDialog : public QDialog
{
    Q_OBJECT

public:
    explicit EditDialog(QWidget *parent = nullptr);
    ~EditDialog();

```

```
22      void set_ui(Book book);
23
24  private slots:
25      //自定义控件事件
26      void cancelButton_click();
27      void confirmButton_click();
28
29
30  signals:
31      void book_edit_confirm(Book book);
32
33  private:
34      Ui::EditDialog *ui;
35  };
36
37  #endif // EDITDIALOG_H
```

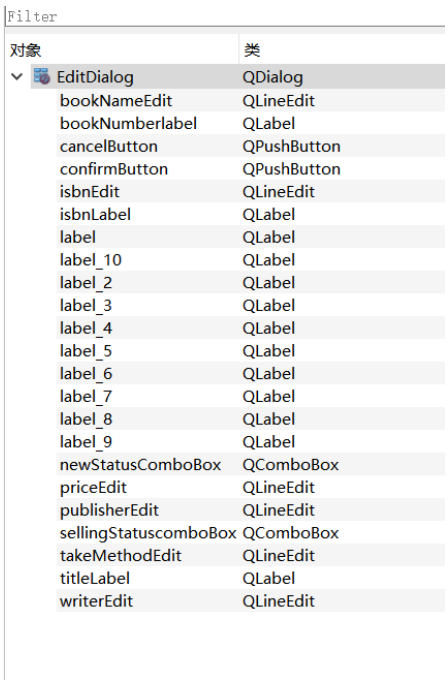
6.12.2 editdialog.cpp

```
1   #pragma once
2   #include "editdialog.h"
3   #include "ui_editdialog.h"
4
5   EditDialog::EditDialog(QWidget *parent) :
6       QDialog(parent),
7       ui(new Ui::EditDialog)
8   {
9       ui->setupUi(this);
10
11      //限制输入长度
12      this->ui->isbnEdit->setMaxLength(60);
13      this->ui->bookNameEdit->setMaxLength(250);
14      this->ui->writerEdit->setMaxLength(40);
15      this->ui->publisherEdit->setMaxLength(40);
16      this->ui->priceEdit->setMaxLength(7);
17      this->ui->takeMethodEdit->setMaxLength(250);
18  }
19
20  EditDialog::~EditDialog()
21  {
22      delete ui;
23  }
24
25  /*
26  函数名：set_ui(Book book)
27  功能：设置书本记录编辑界面的内容
28  */
29  void EditDialog::set_ui(Book book)
30  {
31      this->ui->bookNumberlabel->setText(QString("%1").arg(book.get_id()));
32      this->ui->isbnEdit->setText(book.get_isbn());
33      this->ui->bookNameEdit->setText(book.get_bookName());
34      this->ui->writerEdit->setText(book.get_writer());
35      this->ui->publisherEdit->setText(book.get_publisher());
36      this->ui->priceEdit->setText(QString("%1").arg(book.get_price()));
```

```
    this->ui->takeMethodEdit->setText(book.get_takeMethod());
    this->ui->newStatusComboBox->setCurrentIndex(book.get_status()-1);
    this->ui->sellingStatuscomboBox->setCurrentIndex(book.get_selled());
}

//=========================================private slots=================================================
//----------------------------------------自定义控件事件-------------------------------------

/*
函数名：cancelButton_click()
功能：取消编辑流程
*/
void EditDialog::cancelButton_click()
{
    this->close();
}



/*
函数名：confirmButton_click()
功能：完成编辑，并将编辑完成的对象发送给卖方界面
*/
void EditDialog::confirmButton_click()
{
    //获取初始化book对象所需的内容
    int id=this->ui->bookNumberlabel->text().toInt();
    QString isbn=this->ui->isbnEdit->text();
    QString bookName=this->ui->bookNameEdit->text();
    QString writer=this->ui->writerEdit->text();
    QString publisher=this->ui->publisherEdit->text();
    QString textPrice=this->ui->priceEdit->text();
    QString takeMethod=this->ui->takeMethodEdit->text();
    int status=this->ui->newStatusComboBox->currentText().toInt();
    int selled=this->ui->sellingStatuscomboBox->currentIndex();
    Account* account=Account::get_account();



    //判断输入信息是否完整
    if(isbn.isEmpty() || bookName.isEmpty() || writer.isEmpty() ||
            publisher.isEmpty() || takeMethod.isEmpty() || QString("%1").arg(textPrice)<=0){
        QMessageBox::warning(NULL,"error","存在未填信息.(也可能是书本价格为0)");
        return ;
    }


    int pos=0;
    QValidator::State res;

    //正则匹配ISBN信息
    QRegExp isbnFormula("^978-7-[0-9]{2,6}-[0-9]{2,6}-[0-9]$");
    QRegExpValidator isbnCheck(isbnFormula,0);
    pos=0;
    res=isbnCheck.validate(isbn,pos);
```

```
92      if(QValidator::Acceptable==res){
93      }
94      else{
95          QMessageBox::warning(NULL,"error","isbn值必须形如“978-7-04-051102-
4”\n"
96                                            "中国的ISBN编号共13位,978为固定,其他
十位,"
97                                            "第一位为中国的标识7,之后是标识出版社的
2-6位数字,"
98                                            "之后是出版社内刊物的流水,最后一位是校
验位0-9或X");
99          return ;
100     }
101
102     //正则匹配书籍信息
103     QRegExp priceFormula("\\d+\\.\\d+|(\\d+)");
104     QRegExpValidator priceCheck(priceFormula,0);
105     res=priceCheck.validate(textPrice,pos);
106     if(QValidator::Acceptable==res){
107         qDebug()<<1;
108     }
109     else{
110         QMessageBox::warning(NULL,"error","书本价格必须是大于0的数值");
111         return ;
112     }
113     float price=this->ui->priceEdit->text().toFloat();
114
115     Book book(id,isbn,bookName,writer,publisher,account-
>get_id(),price,takeMethod,status,selled);
116     emit book_edit_confirm(book);
117 }
118
```

6.12.3 editdialog.ui



6.13 Main函数

```
#pragma once

#include "login.h"
#include "dbmanager.h"
#include "register.h"


DbManager* DbManager::dba=NULL;
Account* Account::account=NULL;

int main(int argc, char *argv[])
{
    QApplication a(argc, argv);


    qDebug()<<"1111";
    //界面声明
    Login loginForm;
    Register registerForm;
    BuyerForm buyerForm;
    SellerForm sellerForm;

    loginForm.show();
    qDebug()<<"2222";


    //连接登录界面和注册界面

 QObject::connect(&loginForm,SIGNAL(show_registerForm()),&registerForm,SLOT(receive_from_login()));

 QObject::connect(&registerForm,SIGNAL(show_loginForm()),&loginForm,SLOT(receive_from_register()));

    //连接登录界面和买(卖)方界面

 QObject::connect(&loginForm,SIGNAL(show_buyerForm()),&buyerForm,SLOT(receive_from_login()));

 QObject::connect(&buyerForm,SIGNAL(show_loginForm()),&loginForm,SLOT(receive_from_buyerOrseller()));

 QObject::connect(&sellerForm,SIGNAL(show_loginForm()),&loginForm,SLOT(receive_from_buyerOrseller()));

    //连接买方界面和买方界面

 QObject::connect(&buyerForm,SIGNAL(show_sellerForm()),&sellerForm,SLOT(receive_from_buyer()));

 QObject::connect(&sellerForm,SIGNAL(show_buyerForm()),&buyerForm,SLOT(receive_from_seller()));

```

```
40
41
42      return a.exec();
43  }
44
```

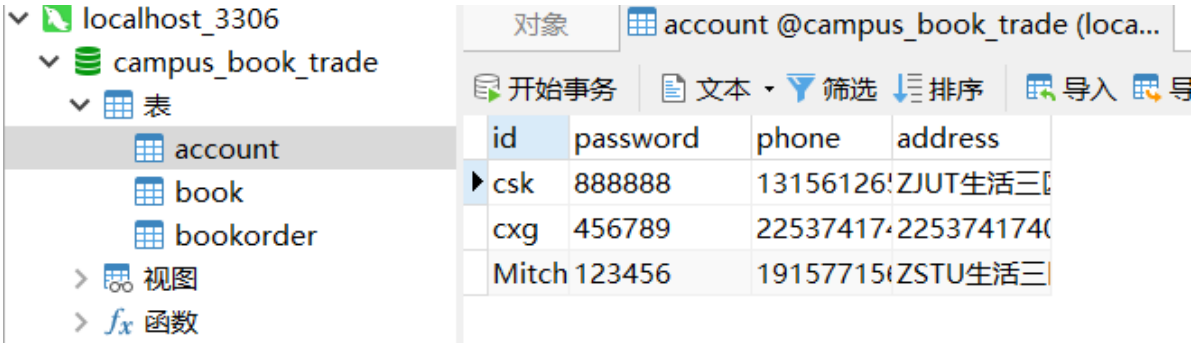
七、运行结果与分析

7.0 输入长度限定

绝大部分的文本编辑框的内容都加上了最大输入长度限定。如图，以登录界面为例

7.1 登录流程

已有的账号如图所示



在登录界面输入正确的用户姓名和密码即可登录，并跳转到买方界面

错误输入测试

1.用户名或密码未输入的情况


Login
校园二手书交易平台
用户名:
Mitchell
密码:
登录
没有账号?点此注册
注册
error
用户名或密码未输入
OK

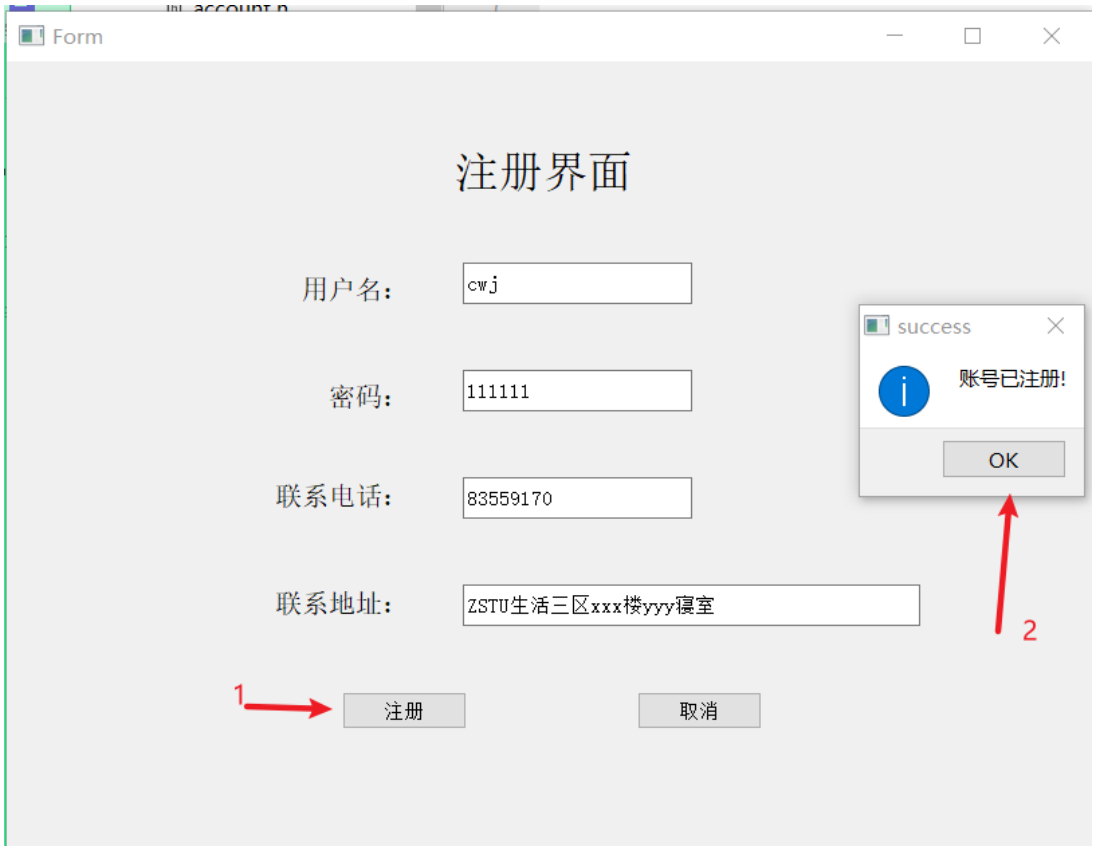
2.密码错误


Login
校园二手书交易平台
用户名:
Mitchell
密码:
12345
正确密码是123456
登录
没有账号?点此注册
注册
error
账号或密码错误
OK

7.2注册流程

进入注册界面：

完善信息并注册：



错误输入测试

1.信息不完善

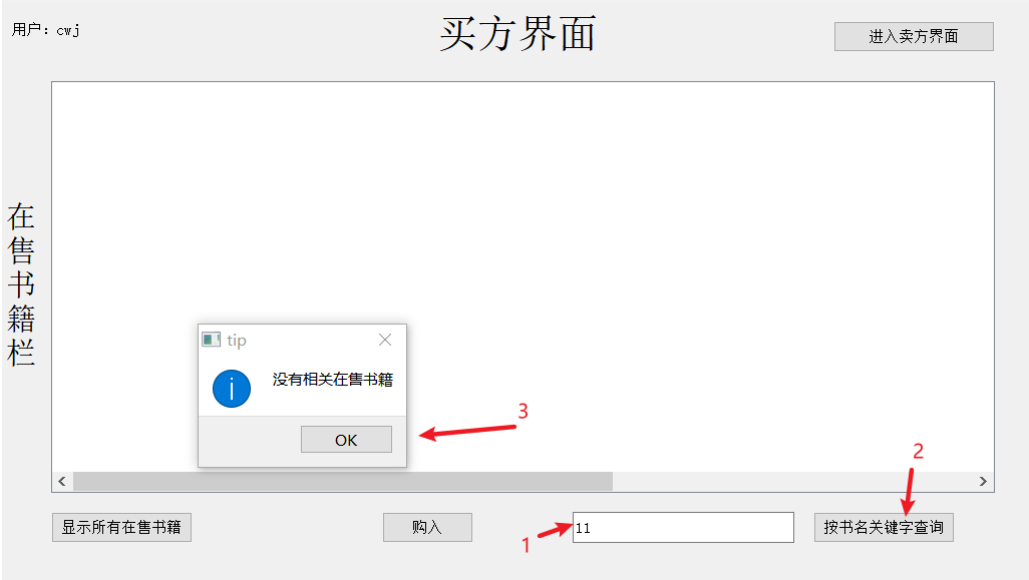
2.用户名冲突

7.3 买方界面

7.3.1 查看正在售卖的书籍

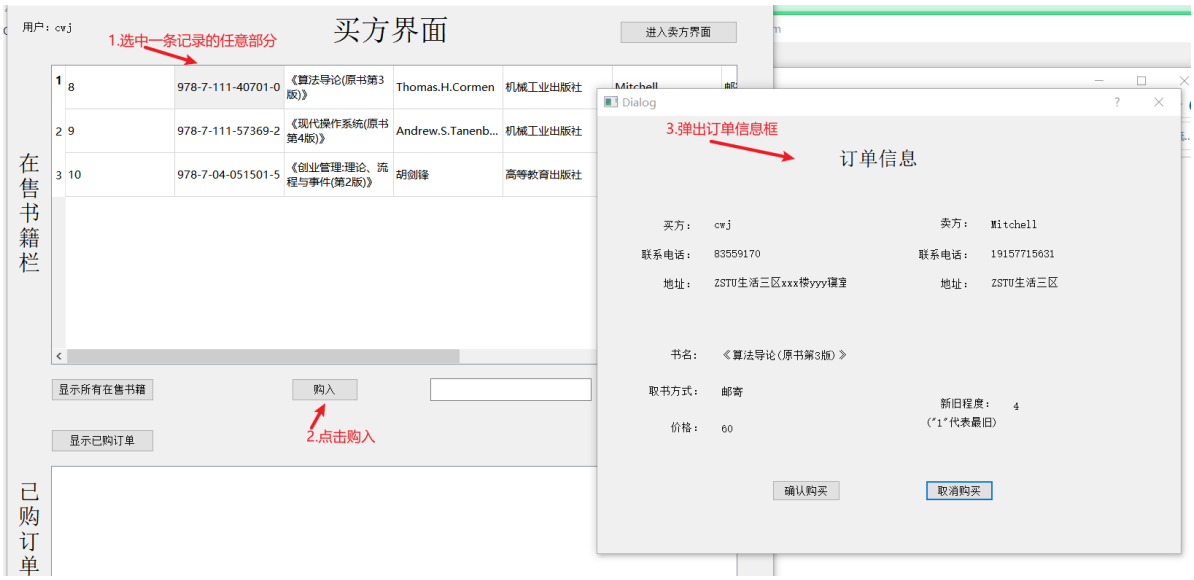
7.3.2 按关键字查看正在售卖的书籍

搜索结果为空则会弹出提示框



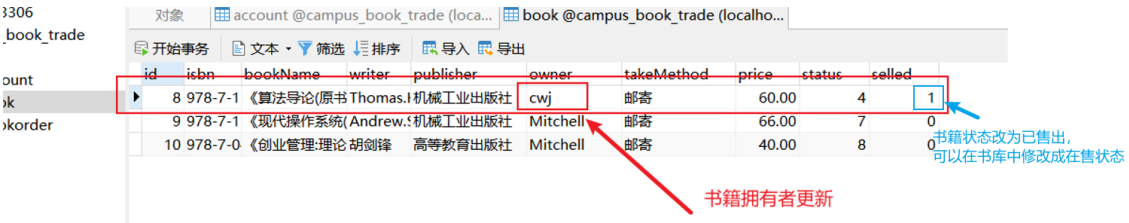
7.3.3 买书流程

1.选中记录

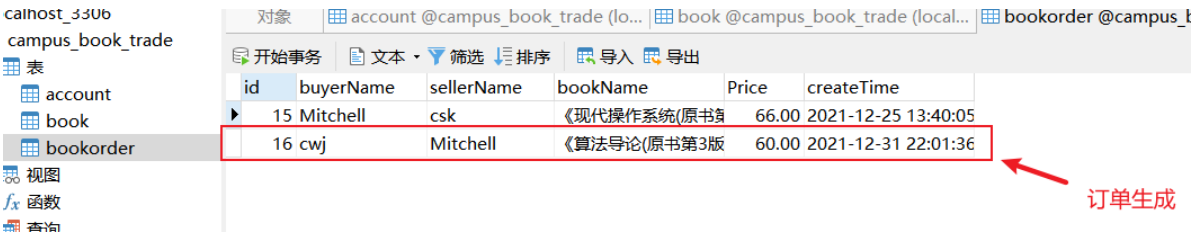


2.确认购买

3.数据库更新



4.订单生成

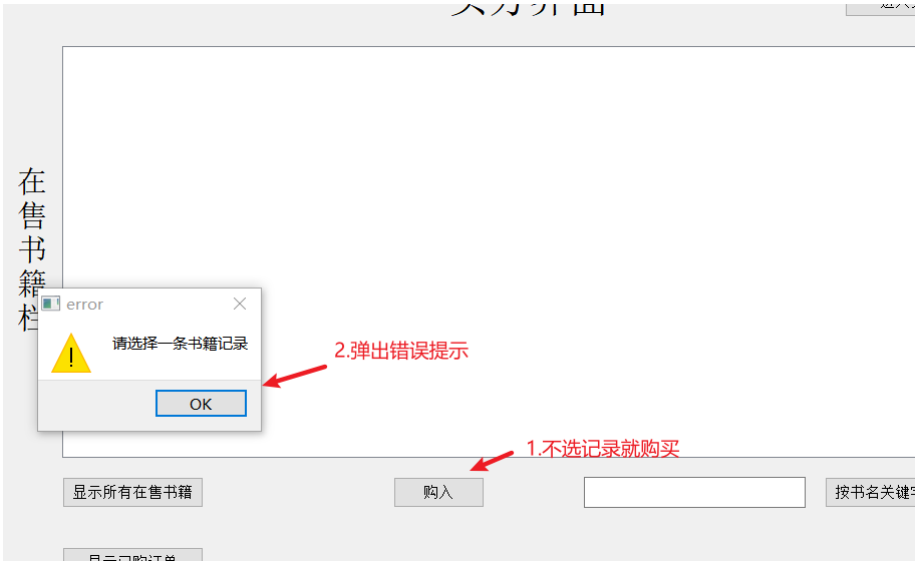


5.订单栏更新

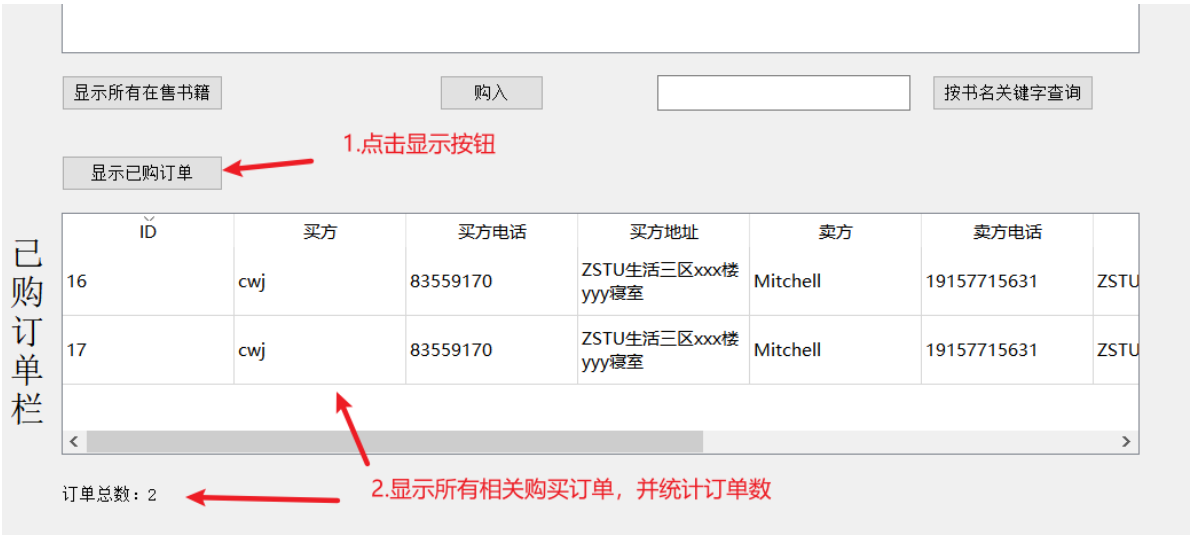
其他输入

1.取消购买

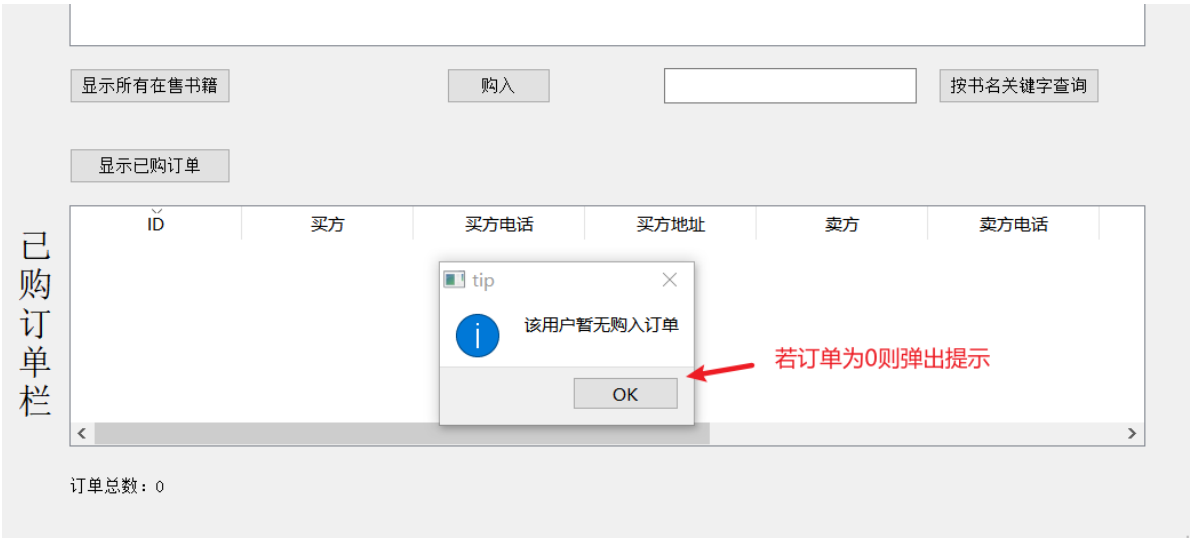
2.未选中记录就点击购买按钮



7.3.4 查看购买订单



若当前用户没有购买订单，则会弹出提示框

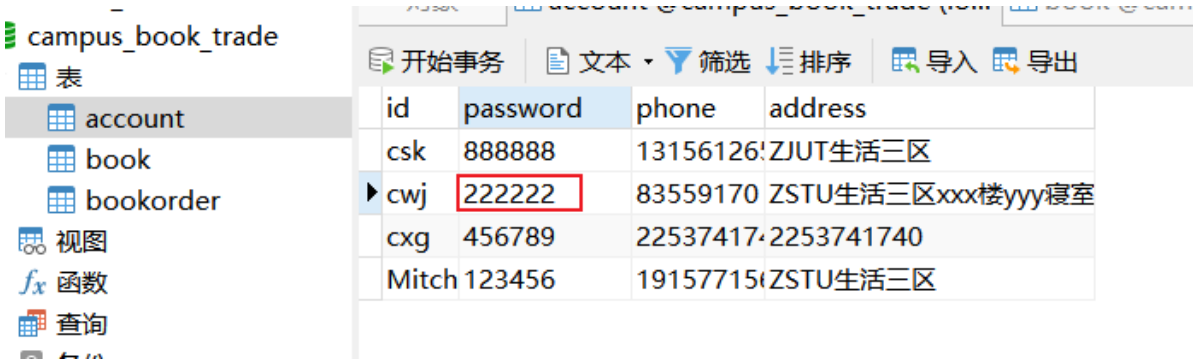


7.3.5 修改密码

1.打开修改密码窗口

2.输入修改密码信息

3.数据库更新



错误输入测试

1.信息不完善

2.原密码不正确

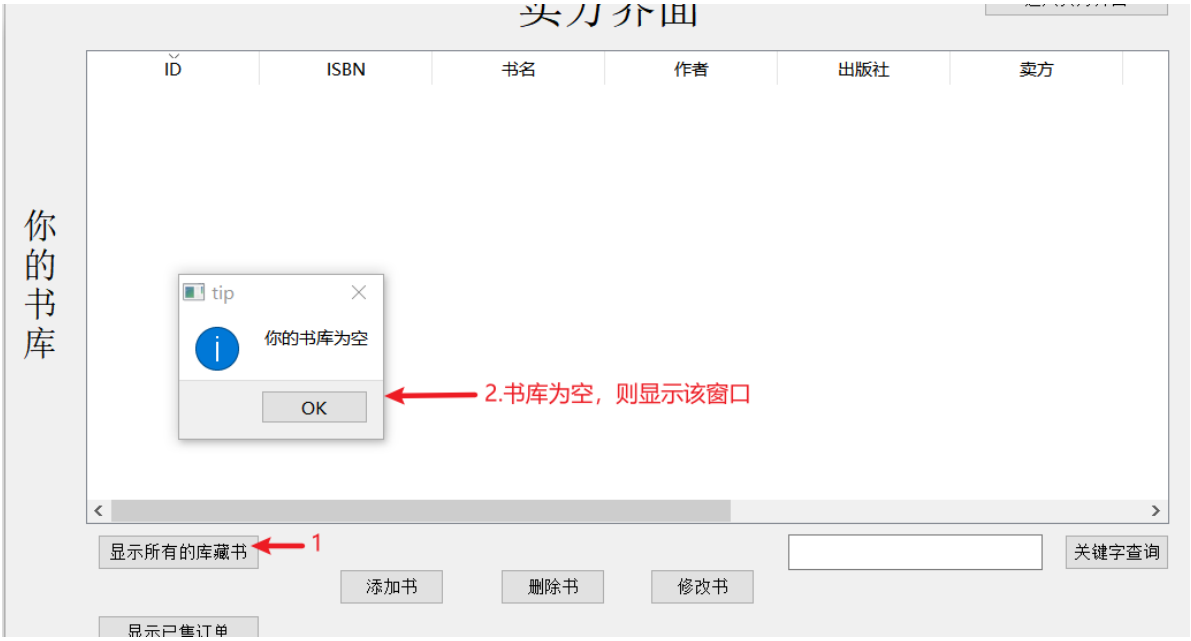
3.新密码与确认密码不匹配

4.新旧密码相同

7.4 卖方界面

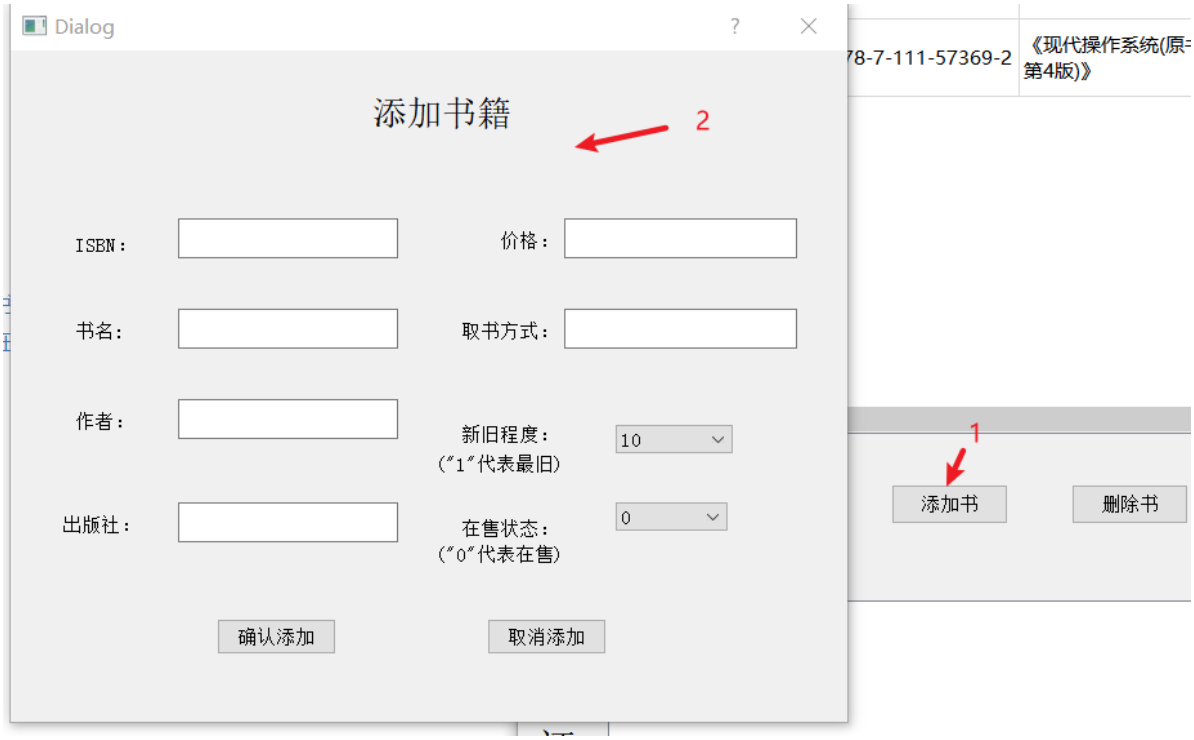
7.4.1 查看自己的书库

书库为空的情况下会给出提示框

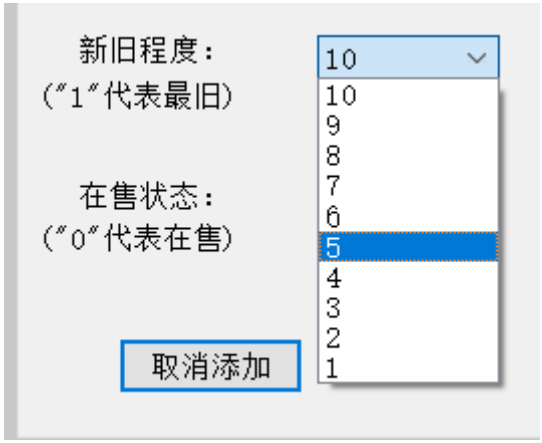


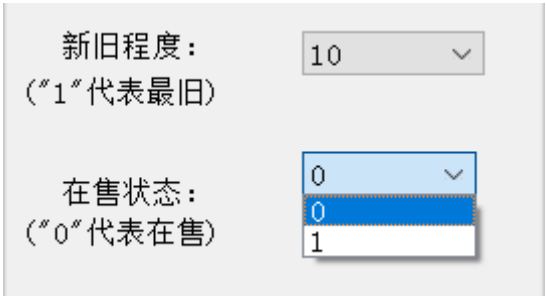
7.4.2 增加一本书

1.打开添加记录窗口



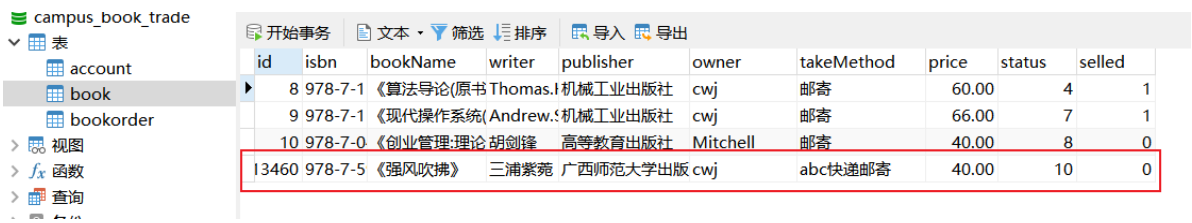
2.多选框展示





3.完善信息并添加记录

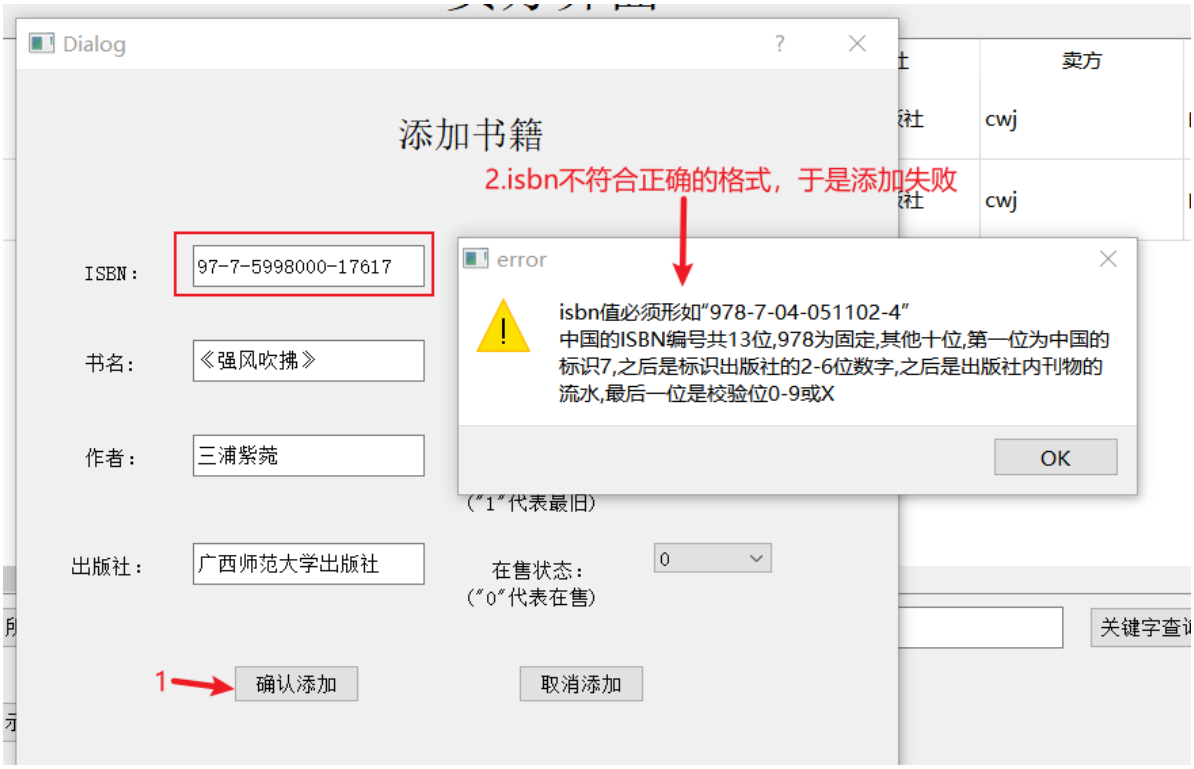
4.数据库更新



错误输入测试

1.存在未输入信息

2.isbn格式错误

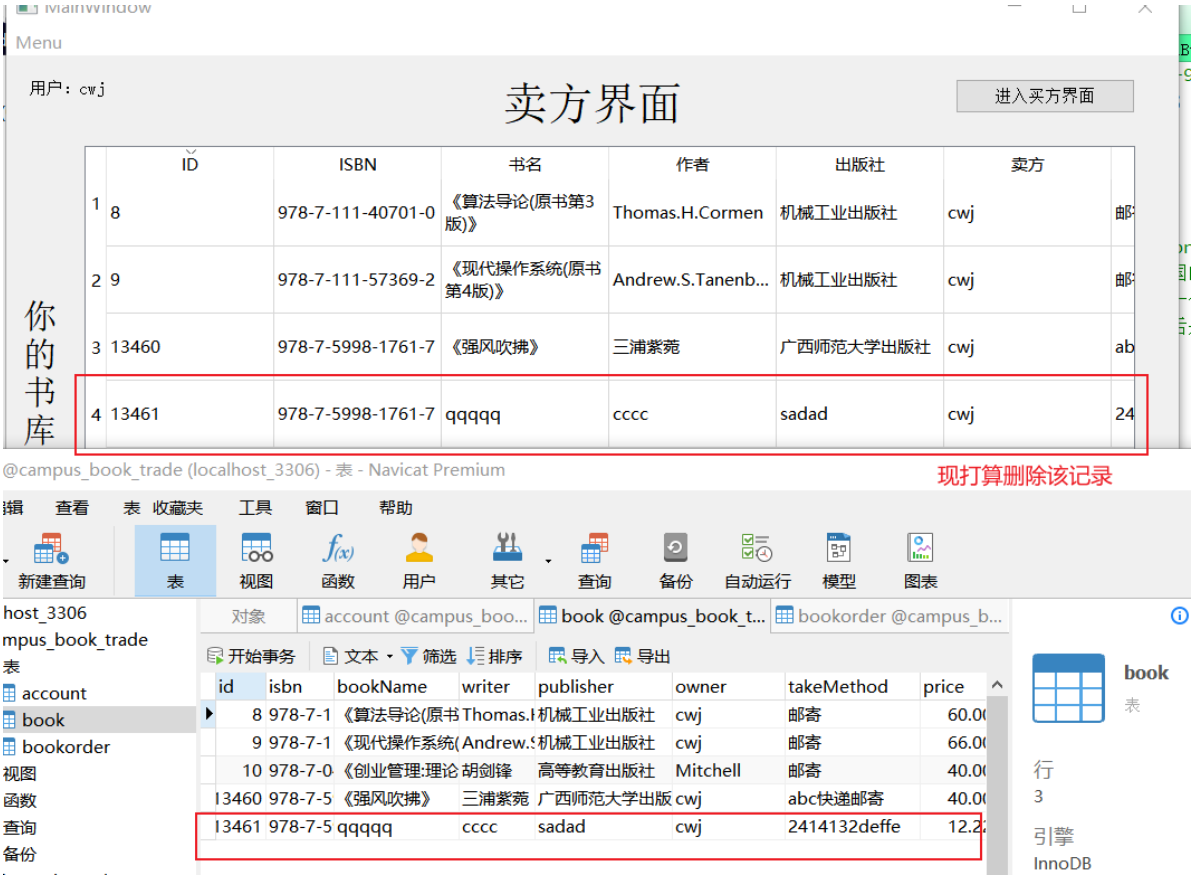


3.价格错误


Dialog
添加书籍
error
书本价格必须是大于0的数值
OK
价格：
40a
取书方式：
邮寄
作者：
三浦紫苑
新旧程度：
（"1"代表最旧）
10
出版社：
广西师范大学出版社
在售状态：
（"0"代表在售）
0
确认添加
取消添加

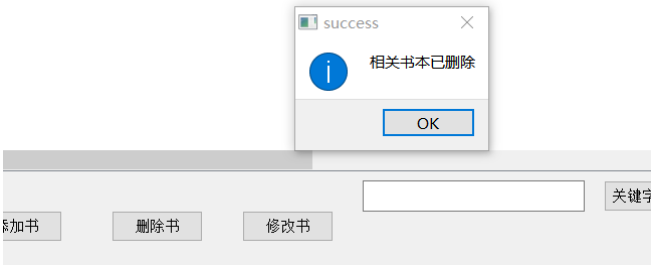
7.4.3 删除一本书

1.书库展示

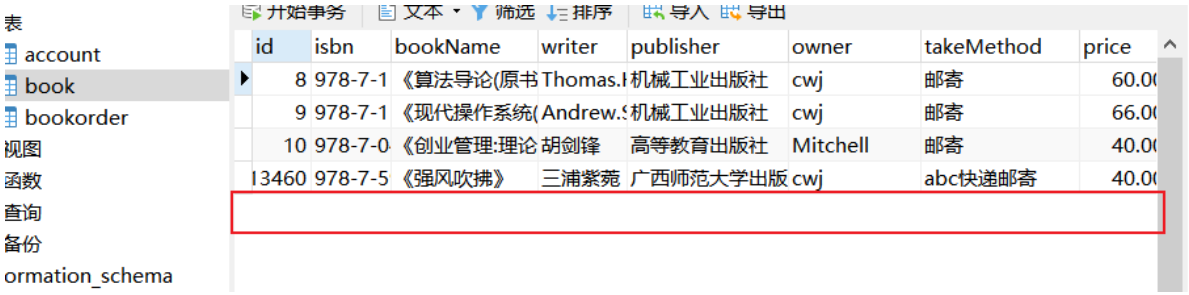


2.选中记录点击删除按钮

3.确认删除

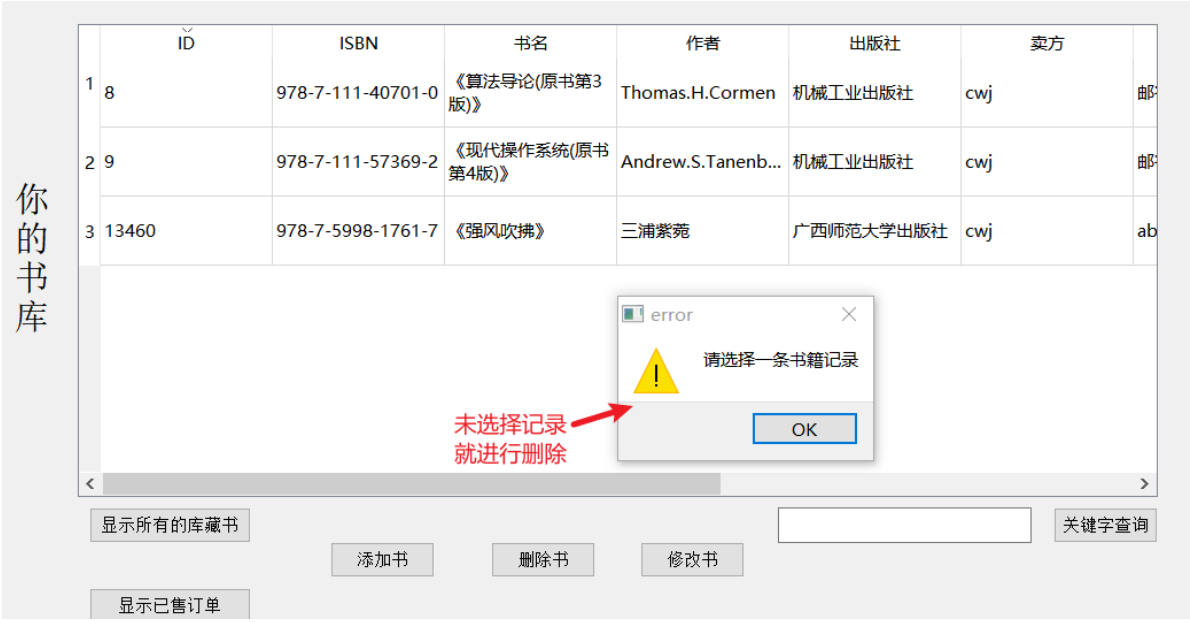


4.数据库更新



其他输入

1.未选择记录就点击删除按钮



7.4.4 修改一本书

大体内容与添加一本书相似。下面以修改书籍的在售状态为例进行测试。

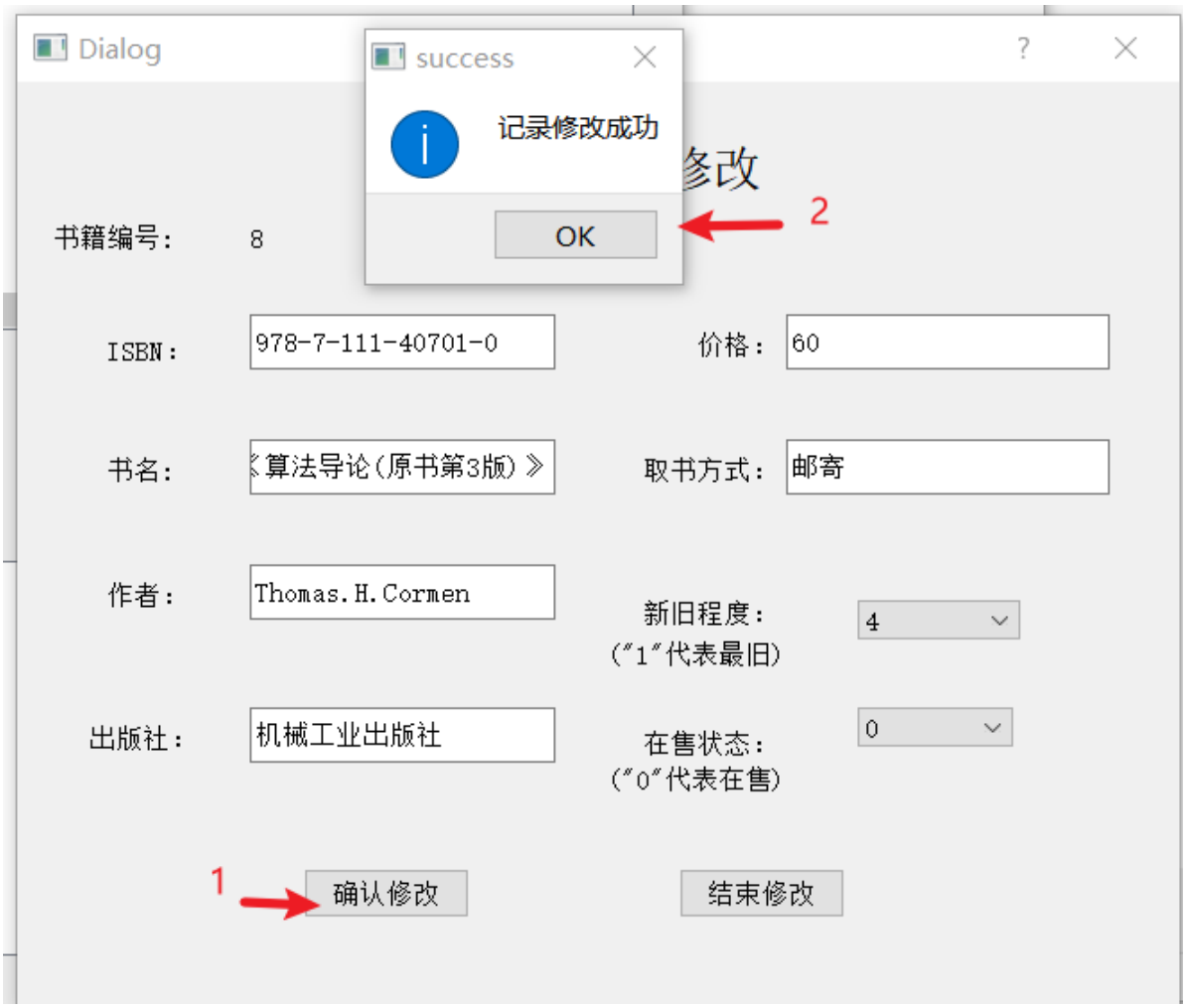
1.书籍在售状态


Menu
用户：Mitchell
买方界面
进入卖方界面
在售书籍栏
1
13460
978-7-5998-1761-7
《强风吹拂》
三浦紫苑
广西师范大学出版社
cwj
ab
cwj的《算法导论》和《现代操作系统》现在不是在售状态
显示所有在售书籍
购入
按书名关键字查询
显示已购订单

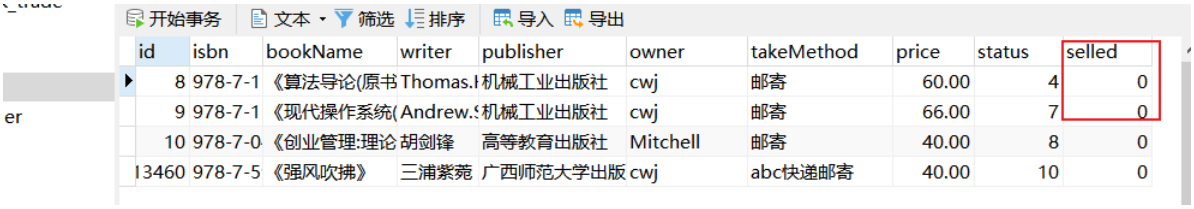
2.修改书籍的在售状态


Menu
用户：cwj
卖方界面
进入买方界面
1.选择一条记录
你的书库
书名
作者
出版社
卖方
取方
价格
新旧程度
售出状
1 《算法导论(原书第3版)》 Thomas.H.Cor... 机械工业出版社 cwj 邮寄 60 4 1
2 《现代操作系统(原书第4版)》 Andrew.S.Tan... 机械工业出版社 cwj 邮寄 66 7 1
3 《强风吹拂》 三浦紫苑 广西师范大学出版社 cwj abc快递邮... 40 10 0
显示所有的库藏书
添加书
删除书
修改书
显示已售订单
2.点击
已售订单栏
Dialog
3.弹出修改栏
书籍信息修改
书籍编号： 8
ISBN： 978-7-111-40701-0
价格： 60
书名： 《算法导论(原书第3版)》
取书方式： 邮寄
作者： Thomas.H.Cormen
新旧程度：
("1"代表最旧)
4
出版社： 机械工业出版社
在售状态：
("0"代表在售)
0
4.改为在售状态
确认修改
结束修改

3.确认修改



4.数据库更新



开始事务 文本 筛选 排序 导入 导出

id	isbn	bookName	writer	publisher	owner	takeMethod	price	status	selled
8	978-7-1	《算法导论(原书	Thomas.H	机械工业出版社	cwj	邮寄	60.00	4	0
9	978-7-1	《现代操作系统(	Andrew.S	机械工业出版社	cwj	邮寄	66.00	7	0
10	978-7-0	《创业管理:理论	胡剑锋	高等教育出版社	Mitchell	邮寄	40.00	8	0
13460	978-7-5	《强风吹拂》	三浦紫苑	广西师范大学出版	cwj	abc快递邮寄	40.00	10	0

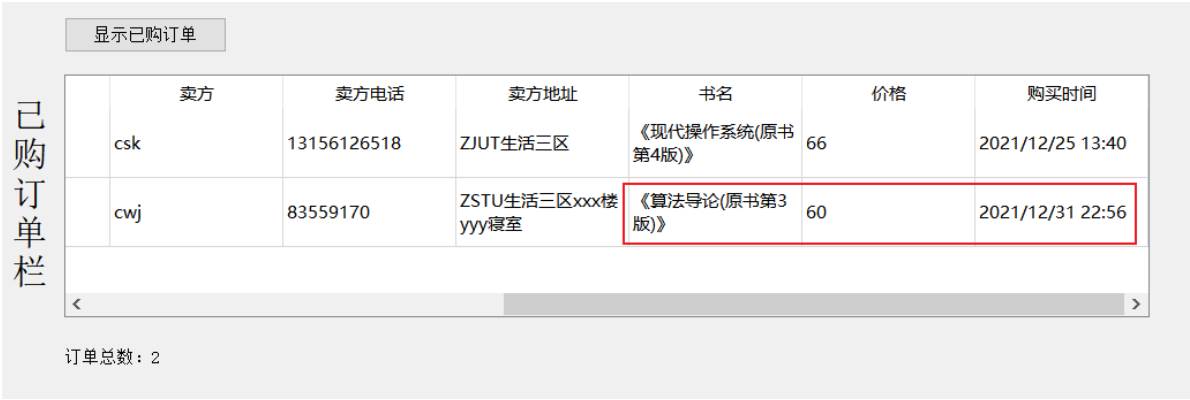
5.修改后其他账号可以看见刚刚改为在售状态的书籍

7.4.5 查看售出订单

紧接7.4.4的例子，现在Mitchell要购买cwj的《算法导论》

1.选中书籍并购买

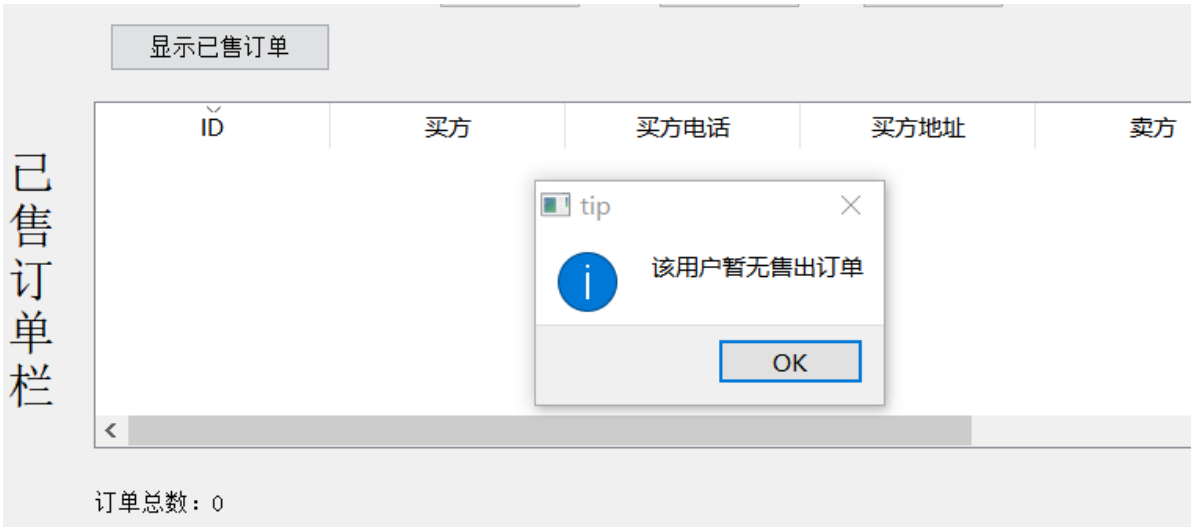
2.买方的购买订单



3.卖方的售出订单

其他输入

1.没有售出订单



八、心得与体会

8.1 缺陷与遗憾

8.1.1 交易逻辑比较简单

一开始在处理购买业务时，思考了比较多的东西，最后得出一个买方一次可以从多个卖方处购买不定数量不定种类的书籍的业务逻辑。不过由于时间有限，上述业务逻辑难以实现，只能添加一些特殊条件简化购买业务。

8.1.2 多账号登录

目前的二手书交易平台一次只能登录一个用户，并没有实现多用户操作。若要实现多个用户同时进入平台并进行操作，应当会涉及并发、多线程相关的知识。

8.1.3 面向对象设计的成果不大好

主要是不大熟悉类的组合、聚合、依赖等各种关系分别在什么情况下使用才是合理的。这一方面的能力仍需多去见识一些学习案例来加深印象。

8.2 收获

8.2.1 初步熟悉了QT

学习QT大概花了一两周的时间。由于QT是以C++为基础的，因此语法上几乎不用花时间注意。整个学习过程以问题为导向，比如：如何实现页面跳转？如何连接数据库？如何实现不同窗口之间的擦书传递？如何使用QT提供的各种控件？…… 带着问题学习会发现学习的目的性会更强，加之不断的实践以及记下踩坑笔记，学过来的知识也更为扎实，QT上手得也更快。

总结出来一个比较好的自学经验：一开始可以先看一些博客并动手实践以达到快速入门的目的，然后针对性的学习可以直接参考官方文档，还可以上网下载相关的项目阅读源码进行学习。

8.2.2 体会到了图形化界面编程的优势

相比于命令行编程，图形化界面将功能分布到特定的界面内，每个界面又各自封装成一个类。实现界面的切换只需要通过在类中定义相关的槽函数并关联相关的控件事件(如点击按钮跳转到指定界面)。整个可视化编程的过程我认为像是一个儿时搭积木的过程，丰富的控件本身就像积木，通过代码搭在一个界面中

QT图形化界面的编程方式个人认为有两个好处：

1. main函数可以简化更多的内容，不用像命令行一样在main函数中写一长串(还可能嵌套)的switch语句。
2. 图形化界面运行后更直观，输入的信息一般都会封装在editLine等已经完善的控件中，并且输入顺序可以不同，比如先输入密码、再输入用户名、最后点击登录按钮执行登录流程，这种所见即所得的效果用命令行难以实现。
3. QT集成了很多特别实用的头文件。比如QDebug可以进行控制台输出，结合QSqlError等异常捕获相关头文件可以提高Debug的效率。
4. QTCreator这个IDE本身比较好用，比如可以将变量名改为驼峰命名，还可以对相同作用域内的同名变量同时修改变量名等。

8.2.3 版本控制

在打算实现一个新功能前，最好是把当前工程先拷贝一份再写代码，如果写出难以解决的问题的话就可以回到原先的备份中重新规划。上面的方法虽然整个过程中没有用上git进行版本控制，但是大体上有一点版本控制的思想。好处就是可以少做很多无用功，写项目的目的更明确，排查错误的范围也会相对更小。