**实 验 报 告**

**课 程 名 称:** **面向对象程序设计与实践2**

**学院 :计算机学院**

**班级 :计科八班**

**学 生 姓 名:李楠轩**

**学 号:2015211362**

1. **实验目的**

**理解面向对象的思想，实际操作类与对象学习C++的基本知识。**

1. **实验内容**

电商交易平台设计

我们经常会在淘宝、京东等电商平台购买产品，这次编程的任务是做一个类似的电商交易平台。此外，为了实现交易功能，我们还要设计一个银行系统，用于产品的支付。

（注：请将所有的类中的数据置为私有，请将不需要被该类外部访问的函数设置为私有；如有良好的界面设计有额外加分）

题目一：银行门户的设计

该门户是一个单独的程序，程序要求至少支持以下功能：

1. **注册银行卡**，银行可选；
2. **修改银行卡密码**；
3. **存款取款**；

银行卡至少需要有卡号，密码，所属银行名称，持卡人身份证号，卡内金额等内容。

要求1：请把所有的银行名写入文件（或数据库），注册银行卡的时候，要求只能选择已经存在的银行进行注册。

要求2：请做好错误场景的处理，例如读银行文件错误，输入数据不合法等等。

题目二：电商平台的设计

该平台是银行门户之外的一个单独的程序，程序要求至少支持以下功能：

1. **注册&登录**：支持新用户注册平台账号，已注册用户用平台账号登录平台。（要求已注册用户的信息长久保留。）
2. **浏览平台产品信息**。
3. **优惠活动**：支持对同一品类下所有产品打折的活动，支持单笔订单满X减Y的活动。
4. **购买产品**：支持用户添加产品到购物车，查看实际应付的产品价格，提交订单。

在题目二我们暂时不考虑提交订单后支付等后续需求。

要求1：电商平台上至少有三类产品：如食物、服装、图书等，每类产品中至少有三个具体的产品（如图书中可以有《C++ Primer》、《Effertive C++》等），每个具体的产品请至少包含产品描述，产品原价，产品剩余量等数据。所有的产品信息需要存储在数据库或文件中，不能写在代码中，平台管理员通过直接修改数据库或文件，管理本平台上的产品，包括产品的增加和删除，修改数量以及具体产品的属性信息等。

要求2：请至少设计一层继承体系（产品基类-产品子类），设计一个产品基类，然后让图书类、电子产品类和服装类等产品子类类继承它，具体的产品是产品子类的实例对象（<<C++ Primer >> 是图书类的实例对象） 。产品基类请至少具有一个虚函数getPrice()用于计算具体产品的价格。

要求3：请通过为每个产品子类定义“品类折扣系数”来支持对同产品子类下所有产品打折扣的活动（如图书全场5折，则图书类这一产品子类的折扣系数为0.5）。

要求4：请做好错误场景的处理。

题目三：网上支付的设计

实现题目三会让你设计的平台像个真正的运行在网络上的平台。题目三要求在题目一、二的基础上支持通过网上支付在电商平台上购物，请至少实现以下功能：

1. **绑定银行卡**：支持电商平台账号绑定银行卡。同一账号可以绑定多张银行卡，而且银行卡可以属于不同银行。
2. **网上支付**：选择绑定的任一银行卡支付电商平台上的订单。

要求1：绑定银行卡和选择银行卡进行支付时都要求提供对应的银行卡密码。

要求2：当在电商平台上绑定银行卡或者进行网上支付的时候，请不要直接打开某个文件查找银行卡的信息，因为银行卡信息文件是银行门户系统私有的，电商平台系统无法直接访问，而应该由电商平台系统去向银行门户系统对接数据。请用socket通信来传送数据。

要求3：请做好错误场景的处理，如绑定银行卡失败，支付失败等。

1. **实验代码**

**见源码**

**增加的文件：**

**bank :**

**mainwindow.cpp中的有关qsocket的源码**

void MainWindow::**on\_pushButton\_clicked**()//查询数据

{ //登陆

QString m=ui->lineEdit->text();

int m1=m.toInt();

QString n=ui->lineEdit\_2->text();

ui->lineEdit->clear();

ui->lineEdit\_2->clear();

//查询数据库

db=QSqlDatabase::database("bank.db");

query=new QSqlQuery(db);

QString sql\_select;

sql\_select=QString("select \* from banks ");

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

}

else

{

bool flag11=true;

while(query->next())

{

if(m1==query->value(0).toInt()&&n==query->value(1).toString())

{

int a=query->value(0).toInt();

QString b=query->value(1).toString();

QString c=query->value(2).toString();

QString d=query->value(3).toString();

QString e =query->value(4).toString();

double f=query->value(5).toDouble();

sp=new bank(a,b,c,d,e,f);

two k(sp);

//成功登陆界面

k.show();

k.*exec*();

flag11=false;

break;

}

}

if(flag11==true)

{

QMessageBox::information(this,"警告","账号或密码错误");

}

}

}

void MainWindow::**on\_pushButton\_2\_clicked**()//进入支付操作

{

\_2two k;

k.*exec*();

}

void MainWindow::**server\_New\_Connect**()//连接

{

//获取客户端连接

socket = server->*nextPendingConnection*();

//连接QTcpSocket的信号槽，以读取新数据

QObject::connect(socket, &QTcpSocket::readyRead, this, &MainWindow::socket\_Read\_Data);

QObject::connect(socket, &QTcpSocket::disconnected, this, &MainWindow::socket\_Disconnected);

//发送按键使能

// ui->pushButton\_Send->setEnabled(true);

qDebug() << "A Client connect!";

}

void MainWindow::**socket\_Read\_Data**()//收到信息后操作

{

QByteArray buffer;

//读取缓冲区数据

buffer = socket->readAll();

QString str;

if(!buffer.isEmpty())

{

//str = ui->textEdit->toPlainText();

str=tr(buffer);

//刷新显示

// ui->textEdit->setText(str);

}

else

{

qDebug()<<"服务端为空";

}

qDebug()<<"服务端"<<str;

//qDebug()<<"服务端"<<str.at(0);

if(str.at(0)=='\*')//绑定银行卡

{

QString receivestr=str.mid(1);

QStringList K=receivestr.split("#");

QString receNAME=K[0];

QString recePASS=K[1];

qDebug()<<"绑定银行"<<receNAME<<" "<<recePASS;

db=QSqlDatabase::database("bank.db");

query=new QSqlQuery(db);

QString sql\_select;

sql\_select=QString("select \* from banks where id='%1' and password='%2' ").arg(receNAME).arg(recePASS);

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

str="";

}

else

{

if(!query->next())

{

socket->write("false");//成功

socket->flush();

str="";

}

else

{

socket->write("true");//失败

socket->flush();

str="";

}

}

}

else

{

if(str.at(0)=='@')//支付时

{

QString receivestr=str.mid(1);

QStringList K=receivestr.split("#");

QString receNO=K[0];

QString recePAY=K[1];

QString recePA=K[2];

qDebug()<<"需要支付"<<receNO<<" "<<recePAY;

db=QSqlDatabase::database("bank.db");

query=new QSqlQuery(db);

int receNO1=receNO.toInt();

qDebug()<<"账号"<<receNO1;

qDebug()<<"密码"<<recePA;

QString sql\_select;

sql\_select=QString("select \* from banks where id='%1' and password='%2'").arg(receNO1).arg(recePA);

//与银行存款比较

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

}

else

{

if(query->next())

{

double balance= query->value(5).toDouble();

qDebug()<<"余钱"<<balance;

double pay=recePAY.toDouble();//接收到的数据

//比较付款与余钱

if(balance<pay)

{

qDebug()<<"余钱不够";

socket->write("0");

socket->flush();

receNO="";

recePAY="";

recePA="";

str="";

}

else

{

sql\_select=QString("update banks set money='%1' where id='%2' ").arg(balance-pay).arg(receNO1);

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

}

socket->write("1");

socket->flush();

receNO="";

recePAY="";

recePA="";

str="";

}

}

else

{

//发回错误

qDebug()<<"没打开数据库";

socket->write("2");

socket->flush();

receNO="";

recePAY="";

recePA="";

str="";

}

}

}

}

}

void MainWindow::**socket\_Disconnected**()//取消链接

{

//发送按键失能

// ui->pushButton\_Send->setEnabled(false);

qDebug() << "Disconnected!";

}

**platform：**

void two::**on\_pushButton\_5\_clicked**()//支付

{

db=QSqlDatabase::database("PlatformUsers.db");

query=new QSqlQuery(db);

query->setForwardOnly(true);

QString sql\_select;

sql\_select=QString("select \* from banklist where userid ='%1'").arg(sp->returnid());

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

}

else

{

if(query->next())

{

int row = ui->tableView\_3->currentIndex().row ();

QAbstractItemModel \*\_2model = ui->tableView\_3->model();

QModelIndex index = \_2model->*index*(row,0);//选中行第一列的内容

QVariant data = \_2model->*data*(index);

QString k=data.toString();

qDebug()<<"k"<<k;

if(k.isEmpty())

{

QMessageBox::information(NULL, "警告", "您未选择银行卡，请先选择银行卡");

}

else

{

qDebug()<<"待付款"<<sp->returnwaitmoney();

this ->close();

pay z(k,sp->returnwaitmoney(),ui->lineEdit->text(),sp);

z.show();

z.*exec*();

}

}

else

{

QMessageBox::information(NULL, "警告", "您未绑定银行卡，请先绑定银行卡");

}

}

}

void two::**on\_pushButton\_12\_clicked**()

{

this->close();

}

void two::**on\_tableView\_clicked**(const QModelIndex &index)

{

qDebug()<<"index";

qDebug()<<index.row();

QAbstractItemModel \*\_3model = ui->tableView->model();

QModelIndex index1 = \_3model->*index*(index.row(),0);//选中行第一列的内容

QVariant data = \_3model->*data*(index1);

QString k=data.toString();

qDebug()<<"k"<<k;

db=QSqlDatabase::database("PlatformUsers.db");

query=new QSqlQuery(db);

query->setForwardOnly(true);

QString sql\_select;

sql\_select=QString("select \* from picture where id ='%1'").arg(k);

QString yy;

if(!query->exec(sql\_select))

{

query->lastError();

}

else

{

query->next();

yy=query->value(2).toString();

qDebug()<<"yy"<<yy;

}

QPixmap myPix(yy);

ui->label\_6->setPixmap(myPix);

ui->label\_6->setScaledContents(true);

ui->label\_6->show();

}

#include "connectbank.h"

#include "ui\_connectbank.h"

#include<QSqlError>

#include<QMessageBox>

#include"ui/two.h"

#include"QThread"

connectbank::**connectbank**(platformuser \*s,QWidget \*parent) :

QDialog(parent),

ui(new Ui::connectbank)

{

ui->setupUi(this);

sp=new platformuser(s->returnid(),s->returnlogname(),s->returnpassword(),s->returnwaitmoney());

socket = new QTcpSocket();

ui->pushButton->setText("绑定");

//连接信号槽

QObject::connect(socket, &QTcpSocket::readyRead, this, &connectbank::socket\_Read\_Data);

QObject::connect(socket, &QTcpSocket::disconnected, this, &connectbank::socket\_Disconnected);

}

connectbank::~***connectbank***()

{

delete ui;

}

void connectbank::**socket\_Read\_Data**()//判断是不是如果收到成功则加入银行

{

QByteArray buffer;

//读取缓冲区数据

buffer = socket->readAll();

QString str;

if(!buffer.isEmpty())

{

// str = ui->textEdit->toPlainText();

str=tr(buffer);

//刷新显示

// ui->textEdit->setText(str);

}

else

{

qDebug()<<"客户端读的是空的";

}

if(str=="true")//比较发过来的数据

{

db=QSqlDatabase::database("PlatformUsers.db");

query=new QSqlQuery(db);

QString sql\_select;

QString receNAME=ui->lineEdit->text();

QString recePASS=ui->lineEdit\_2->text();

sql\_select=QString("insert into banklist values(null,'%1','%2') ").arg(sp->returnid()).arg(receNAME);

query->exec(sql\_select);

// ui->pushButton->setText("成功");

ui->label\_3->setText("成功");

// ui->pushButton->setEnabled(true);

// QThread::sleep(100);

on\_pushButton\_2\_clicked();

}

else{

// ui->pushButton->setText("失败");

ui->label\_3->setText("失败");

ui->pushButton->setEnabled(true);

}

}

void connectbank::**socket\_Disconnected**()

{

//发送按键失能

//ui->pushButton\_Send->setEnabled(false);

//修改按键文字

//ui->pushButton\_Connect->setText("连接");

qDebug() << "Disconnected!";

}

void connectbank::**on\_pushButton\_clicked**()//绑定银行卡

{

QString IP;

int port;

QString logname=ui->lineEdit->text();

QString password=ui->lineEdit\_2->text();

if(!logname.isEmpty()&&!password.isEmpty())

{

db=QSqlDatabase::database("bank.db");

query=new QSqlQuery(db);

QString sql\_select;

sql\_select=QString("select \* from banklist where userid='%1' and cardno='%2'").arg(sp->returnid()).arg(logname);

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

}

else

{

if(query->next())

{

QMessageBox::about(NULL, "About", "已存在");

}

else

{

//获取IP地址

IP = "127.0.0.1";

//获取端口号

port = 8086;

//取消已有的连接

socket->abort();

//连接服务器

socket->*connectToHost*(IP, port);

//等待连接成功

if(!socket->*waitForConnected*(30000))

{

qDebug() << "Connection failed!";

return;

}

else

{

qDebug() << "Connect successfully!";

QString flagdate="\*";

flagdate+=logname;

flagdate+="#";

flagdate+=password;

socket->write(flagdate.toLatin1());

socket->flush();

// ui->label\_3->setText("请等待");

//ui->pushButton->setText("请等待");

ui->pushButton->setEnabled(false);

}

}

}

}

else

{

ui->label\_3->setText("未输入");

}

}

void connectbank::**on\_pushButton\_2\_clicked**()//回到原始界面

{

platformuser z(sp->returnid(),sp->returnlogname(),sp->returnpassword(),sp->returnwaitmoney());

two y(z);

this->close();

y.show();

y.*exec*();

}

#include "pay.h"

#include "ui\_pay.h"

#include<QSqlError>

#include<QDebug>

#include"ui/two.h"

pay::**pay**(QString a, double b,QString e, platformuser\*s, QWidget \*parent) :

QDialog(parent),

ui(new Ui::pay)

{

ui->setupUi(this);

db3=QSqlDatabase::database("PlatformUsers.db");

query3=new QSqlQuery(db);

db4=QSqlDatabase::database("PlatformUsers.db");

query4=new QSqlQuery(db);

sp=new platformuser(s->returnid(),s->returnlogname(),s->returnpassword(),s->returnwaitmoney());

//获得账号金额密码

c=a;

d=b;

f=e;

ui->pushButton->setText("请等待");

ui->pushButton->setEnabled(false);

ui->pushButton\_2->setEnabled(false);

socket = new QTcpSocket();

//连接信号槽

socket\_pay();

QObject::connect(socket, &QTcpSocket::readyRead, this, &pay::socket\_Read\_Data);

QObject::connect(socket, &QTcpSocket::disconnected, this, &pay::socket\_Disconnected);

}

pay::~***pay***()

{

delete ui;

}

void pay::**socket\_Read\_Data**()

{

QByteArray buffer;

//读取缓冲区数据

buffer = socket->readAll();

QString str;

if(!buffer.isEmpty())

{

// str = ui->textEdit->toPlainText();

str=tr(buffer);

//刷新显示

//ui->textEdit->setText(str);

}

else

{

qDebug()<<"客户端读的是空的";

}

if(str=="1")//成功

{

db=QSqlDatabase::database("PlatformUsers.db");

query=new QSqlQuery(db);

QString sql\_select;

sql\_select=QString("update student set waitmoney= 0 where id = '%1'").arg(sp->returnid());

qDebug()<<"支付成功？";

qDebug()<<"id？"<<sp->returnid();

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

str="";

}

else

{

QString y;

ui->label->setText("成功支付");

sql\_select=QString("select \* from shoplist where userid='%1'").arg(sp->returnid());

db1=QSqlDatabase::database("PlatformUsers.db");

query1=new QSqlQuery(db);

db2=QSqlDatabase::database("PlatformUsers.db");

query2=new QSqlQuery(db);

QString sql\_select1;

int c11;

if(!query->exec(sql\_select))

{

qDebug()<<"1"<<query->lastError();

}

else

{

while(query->next())

{

int a=query->value(4).toInt();

QString b=query->value(3).toString();

qDebug()<<"A"<<a<<"b"<<b;

sql\_select1=QString(" select \* from goods where name = '%1' ").arg(b);

if(!query1->exec(sql\_select1))

{

qDebug()<<"2"<<query1->lastError();

}

else

{

if(query1->next())

{

c11=query1->value(3).toInt();

}

else

{

qDebug()<<"mei zhaodao";

}

}

qDebug()<<"到这里了";

y=QString("update goods set num= '%1' where name ='%2'").arg(c11-a).arg(b);

query2->exec(y);

}

}

QString sql\_select2;

QString sql\_select3;

sql\_select2=QString("select \*from shoplist where userid = '%1'").arg(sp->returnid());

if(!query3->exec(sql\_select2))

{

qDebug()<<query3->lastError();

}

else

{

while (query3->next())

{

qDebug()<<"holyshit";

sql\_select3=QString("insert into paylist values(null,'%2','%3','%4','%5','%6','%7','%8')").arg(sp->returnid()).arg(query3->value(2).toString()).arg(query3->value(3).toString()).arg(query3->value(4).toInt()).arg(query3->value(5).toString()).arg(query3->value(6).toDouble()).arg(query3->value(7).toDouble());

if(!query4->exec(sql\_select3))

{

qDebug()<<query4->lastError();

}

else

{

qDebug()<<"成功放入paylist";

}

}

}

sql\_select=QString("delete from shoplist where userid= '%1'").arg(sp->returnid());

if(!query->exec(sql\_select))

{

qDebug()<<query->lastError();

}

{

ui->pushButton->setText("返回");

ui->pushButton->setEnabled(true);

// this->close();

str="";

}

}

}

else

{

if(str=="0")//钱不够

{

ui->label->setText("余钱不够");

ui->pushButton\_2->setText("返回");

ui->pushButton\_2->setEnabled(true);

//this->close();

str="";

}

if(str=="2")//密码不对

{

ui->label->setText("密码不对");

ui->pushButton\_2->setText("返回");

ui->pushButton\_2->setEnabled(true);

//this->close();

str="";

}

}

}

void pay::**socket\_pay**()

{

QString IP;

int port;

//获取IP地址

IP = "127.0.0.1";

//获取端口号

port = 8086;

//取消已有的连接

socket->abort();

//连接服务器

socket->*connectToHost*(IP, port);

//等待连接成功

if(!socket->*waitForConnected*(30000))

{

qDebug() << "Connection failed!";

return;

}

else

{

qDebug() << "Connect successfully!";

QString flagdate="@";

flagdate+=c;

flagdate+="#";

QString e;

e.setNum(d);

flagdate+=e;

flagdate+="#";

flagdate+=f;

flagdate+="#";

socket->write(flagdate.toLatin1());

socket->flush();

}

}

void pay::**socket\_Disconnected**()

{

//发送按键失能

//ui->pushButton\_Send->setEnabled(false);

//修改按键文字

//ui->pushButton\_Connect->setText("连接");

qDebug() << "Disconnected!";

}

void pay::**on\_pushButton\_clicked**()

{

this->close();

platformuser y(sp->returnid(),sp->returnlogname(),sp->returnpassword(),0.0);

two z(y);

z.show();

z.*exec*();

}

void pay::**on\_pushButton\_2\_clicked**()

{

this->close();

platformuser y(sp->returnid(),sp->returnlogname(),sp->returnpassword(),sp->returnwaitmoney());

two z(y);

z.show();

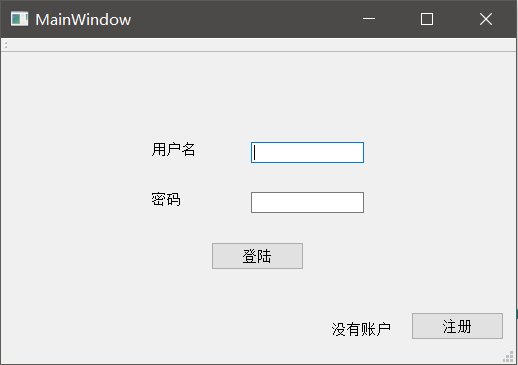
z.*exec*();

}

1. **结果分析、用户手册**

**实验一：**

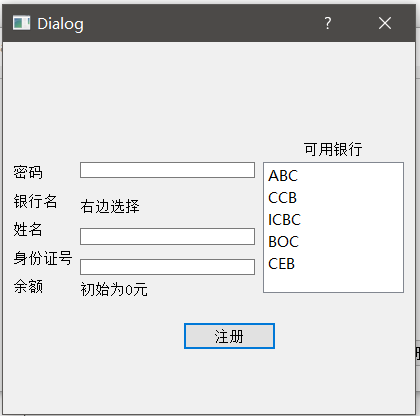
**登陆界面**



**可以注册和登陆**

**----------------------------**

**注册界面**



**身份证18位密码6位数字选择银行点击注册**

**成功框**



**记住账号**

**登陆成功框**



**可以进行存款取款修改密码的操作**

**存500元**



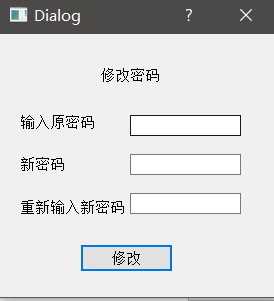


**-----------------------------------------------**

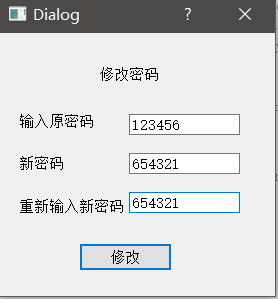
**取500**



**修改密码**



**直接修改**





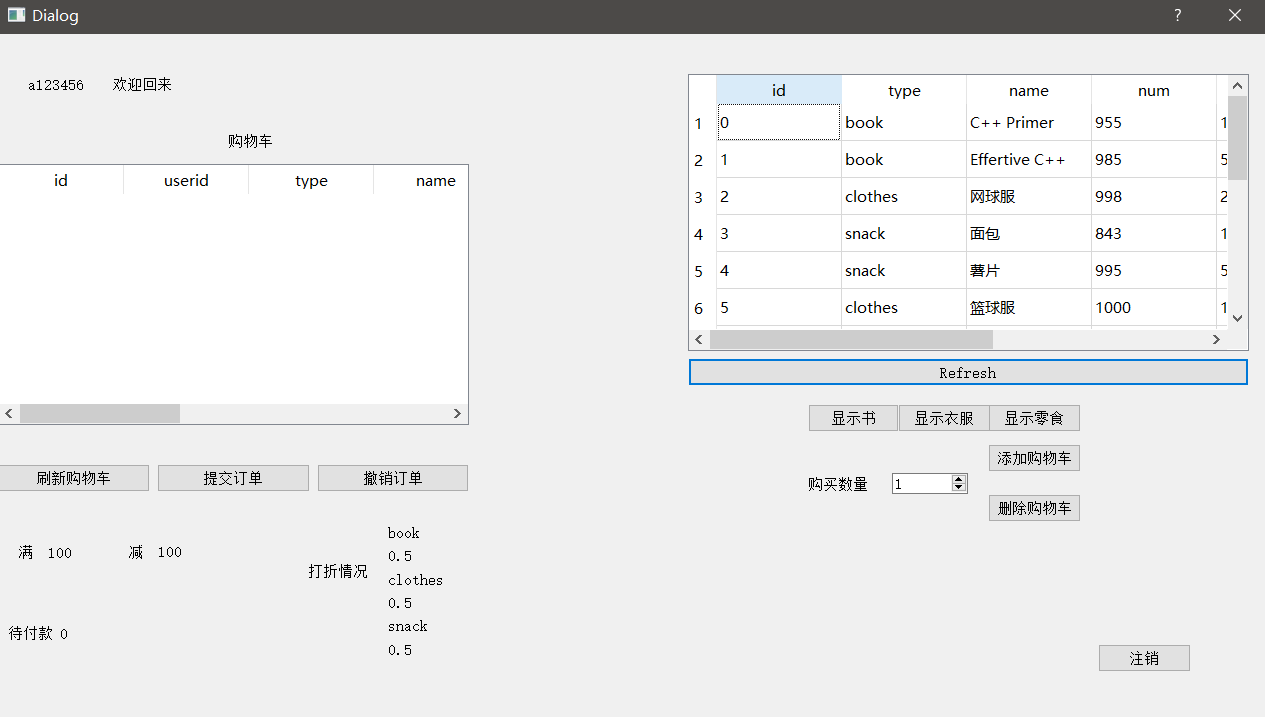
**实验二：**

**初始界面**



**可以登陆和注册，需要先填写在注册**

**登陆成功框，**



**注册：**





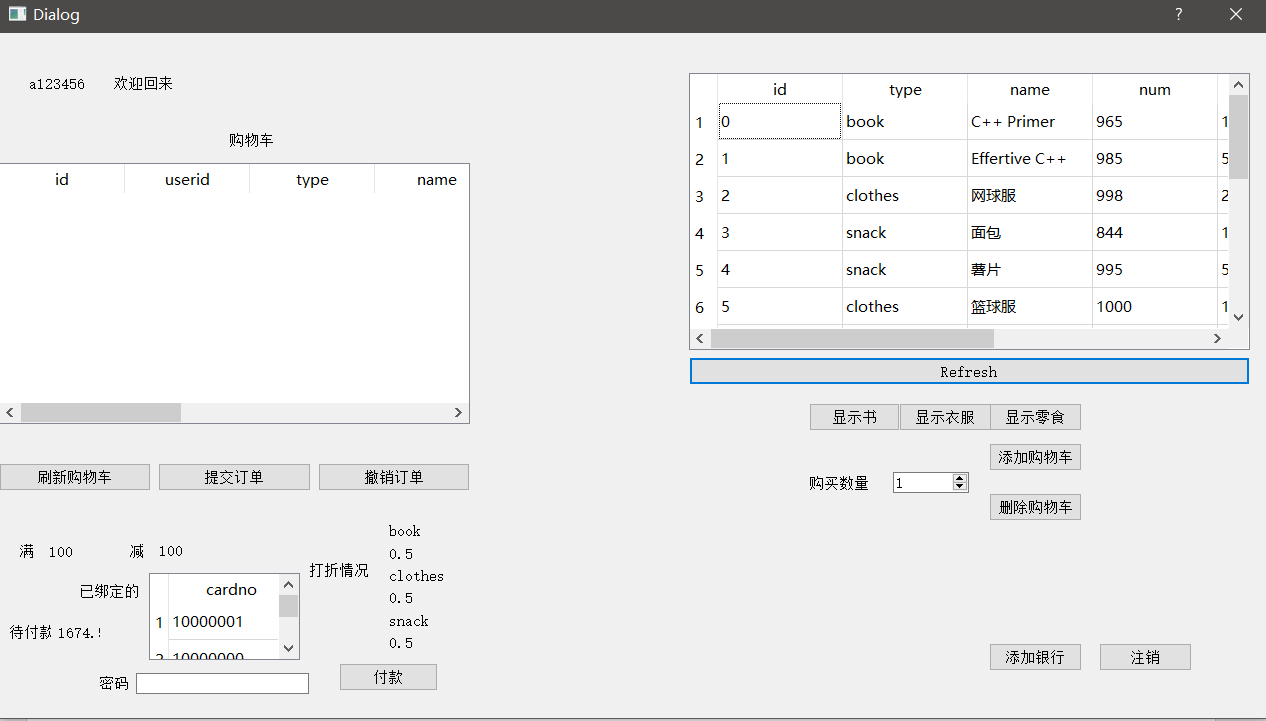
**其他操作与界面三相同**

**实验三**



**1初始界面，输入内容点击登陆或注册**

**登陆：登陆成功框**

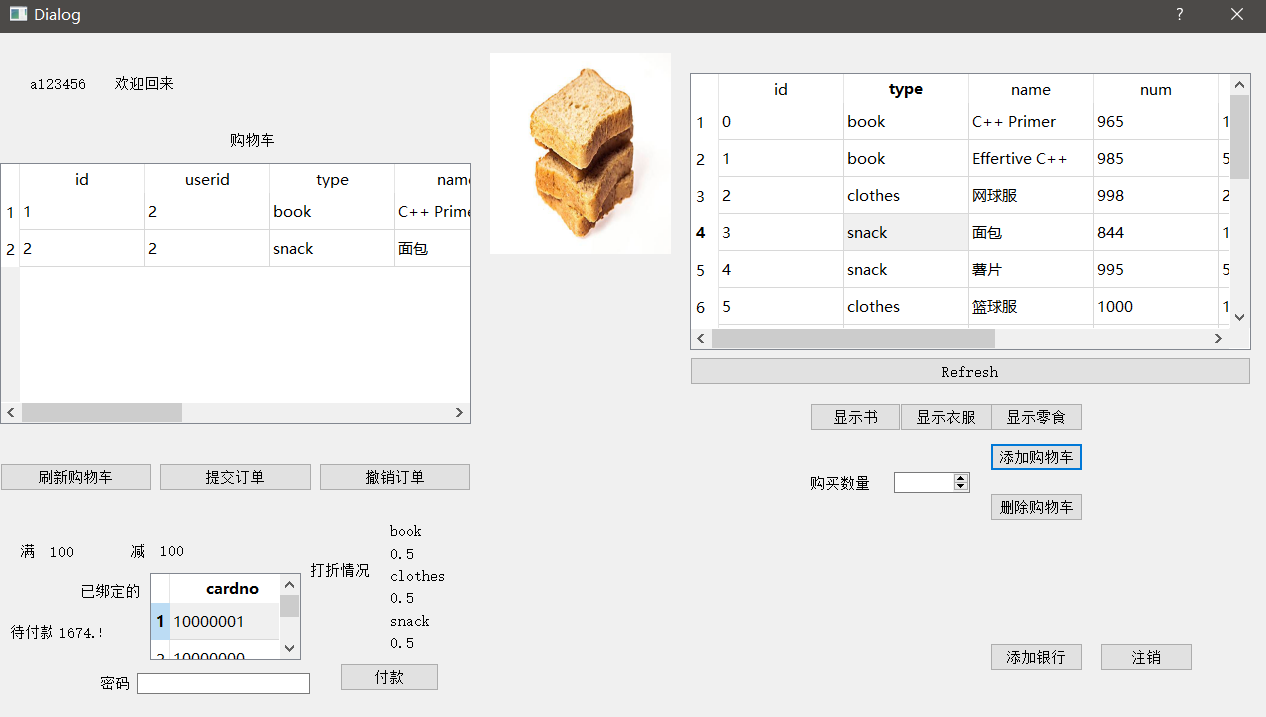


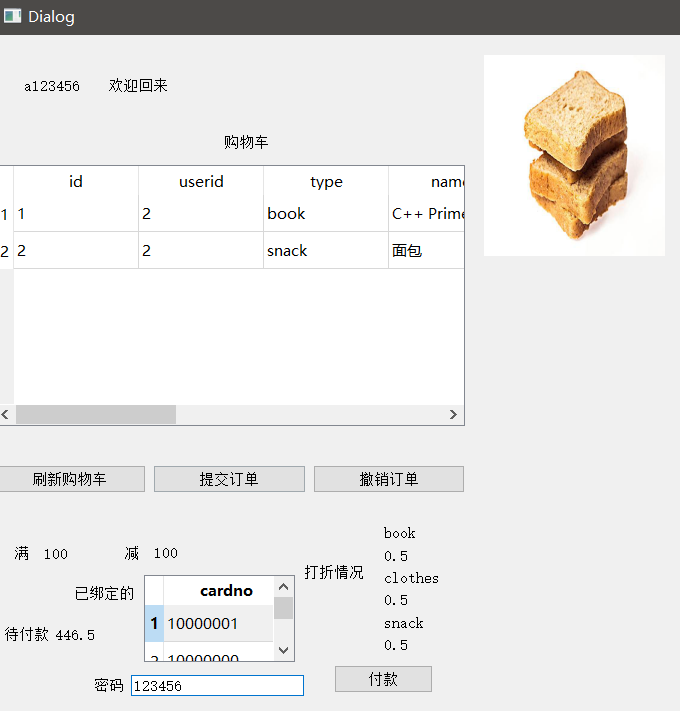
左边是您的购物车，你可以看你已购买的商品，提交订单则可支付，但是要选择用哪一个银行支付。

待付款会显示在左边。撤销订单则就不会支付了，购物车不变。

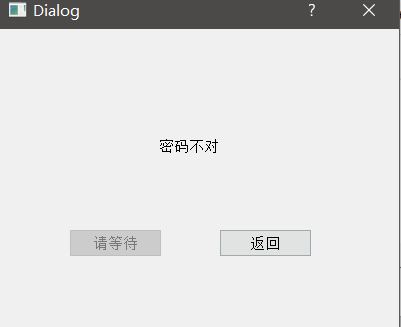
上面点击商品并填写数量点击添加购物车即可。删除购物车则需要点击购物车的商品再点击删除购物车

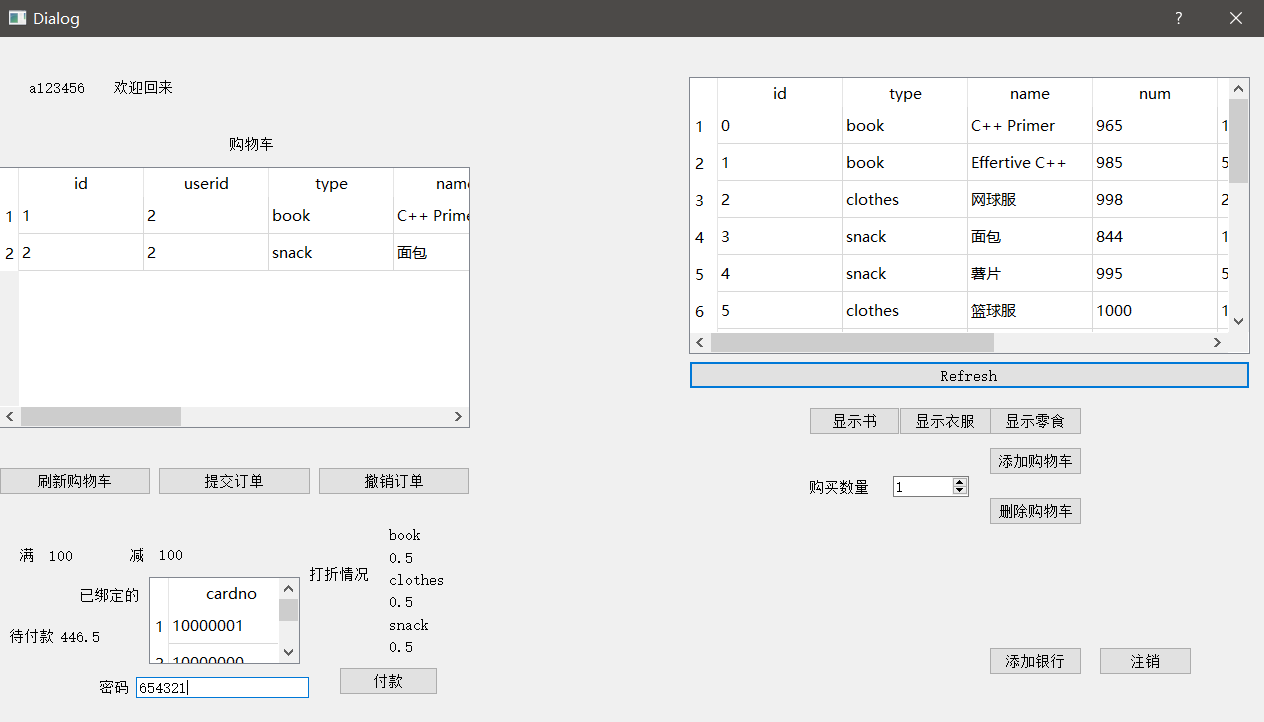
支付页面：





**密码输错会提示**

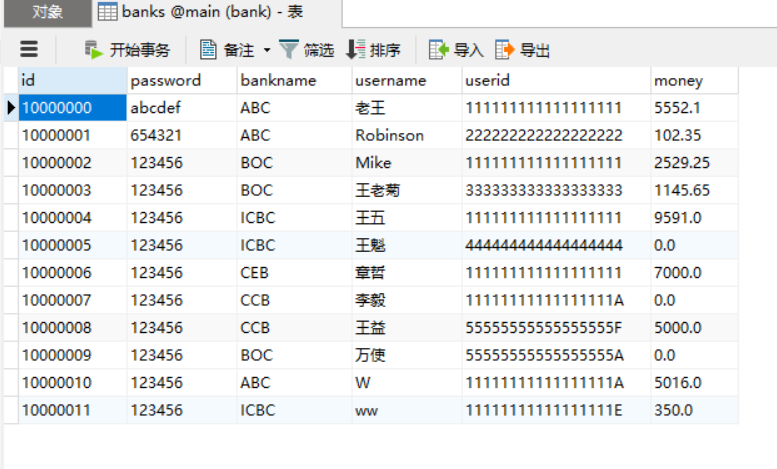


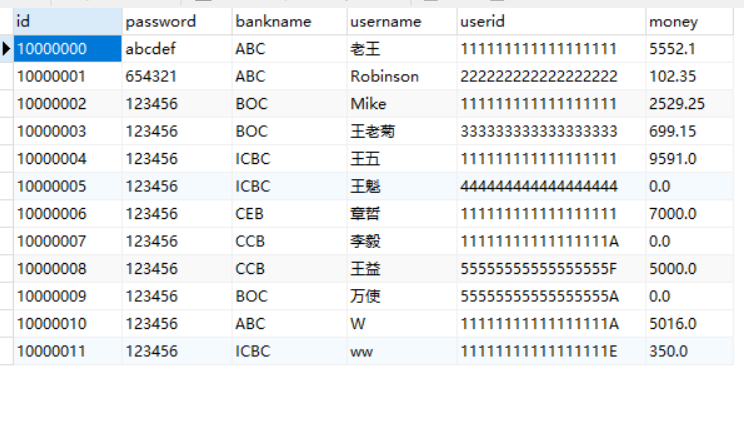


**成功支付**

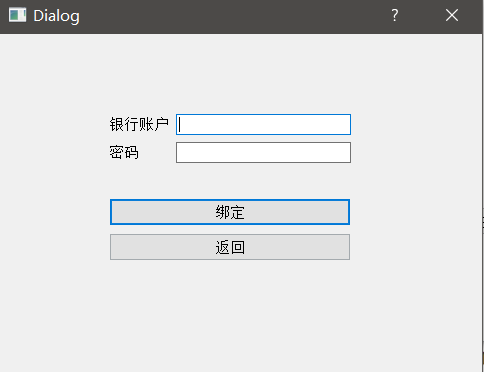


数据库改变





**1.2注册银行卡**



**按提示操作即可**

1. **总结分析**

**第三题是在第二题和第一题上的添加功能，难点在与socket的使用和C++面向对象的运用，感觉本次实验对于C++的理解更深入，也对与面向对象的过程有了一个明显的体验。** **我从对象的角度来理解一些东西，才更有体会，而不再像C一样面向过程，我发现了一些新的东西，比如说用类封装之后安全性能提高很多，比如说用类可以使结构更加的简化，只要给外部API即可。**