



程序设计实践

电商交易平台实现



2018-1-1

[裴子祥 计科七班 学号 2015211921]

[指导老师：闫丹凤 王玉龙]

目录

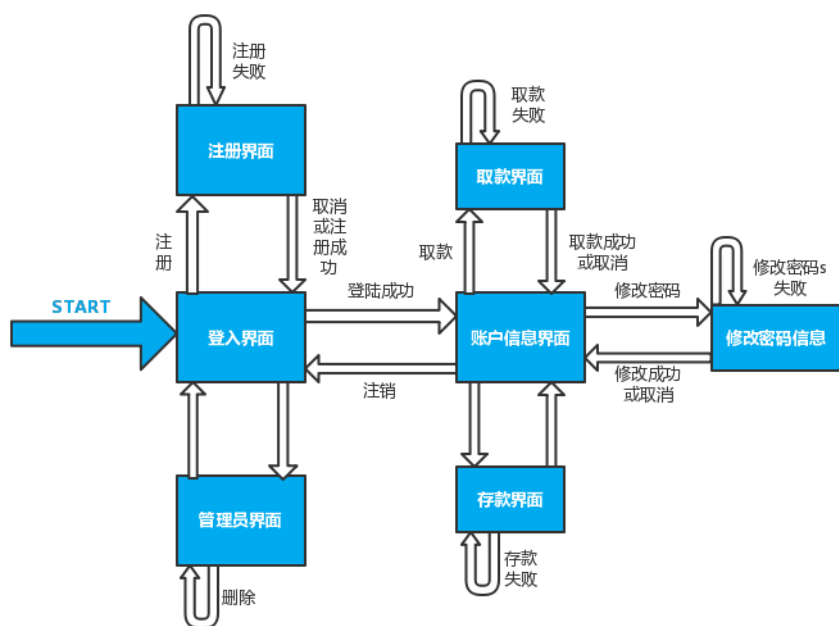
前言	2
一. 银行门户的设计	2
1. 模块划分	2
2. 数据库表结构	2
3. 功能实现	3
4. 数据结构	4
二. 电商平台的设计	5
1. 模块划分	5
2. 数据库表结构	5
3. 功能实现	7
4. 数据结构	9
三. 电商平台网络版	10
1. 模块设计	11
2. 功能实现	11
3. 数据结构	12
四. 设计模式应用	15
五. 程序 UI 设计	18
六. 程序课程设计总结	19
七. 不足之处与优化方向	19
八. 程序运行截图	20
九. 源码附件	23

前言

程序的**亮点**：本程序主要由两个进程组成，银行门户与商城平台。银行门户功能将在正文中展示。商城平台功能将在正文中展示。在完成所要求的功能情况下，合理的使用了单例、简单工厂等设计模式，有美观的用户界面，容错处理也做得较为完善，如溢出等非常规错误都有处理，使得程序的鲁棒性较好，部分错误处理使用了 try-catch 机制，自己定义异常类，并抛出。当然也有一些我自己添加的功能，如排序，模糊搜索，分类查看，用户在购物车自选支付，还有银行管理员的删、查，电商管理员的增、删、改、查；用户银行卡的解绑，一系列完整的功能。

一. 银行门户的设计

1. 模块划分



2. 数据库表结构

名称	类型	架构
表 (2)		
accountInfo		CREATE TABLE accountInfo (num NVARCHAR(20), password NVARCHAR(20), bankname NVARCHAR, idcredit NVARCHAR(20), phonenum NVARCHAR(15), balance REAL)
num	NVARCHAR (20)	'num' NVARCHAR (20)
password	NVARCHAR (20)	'password' NVARCHAR (20)
bankname	NVARCHAR	'bankname' NVARCHAR
idcredit	NVARCHAR (20)	'idcredit' NVARCHAR (20)
phonenum	NVARCHAR (15)	'phonenum' NVARCHAR (15)
balance	REAL	'balance' REAL
bankInfo		CREATE TABLE bankInfo (name VARCHAR PRIMARY KEY)
name	VARCHAR	'name' VARCHAR

账户信息表 accountInfo

```
query.exec("CREATE TABLE accountInfo ( num VARCHAR, password VARCHAR, bankname VARCHAR, idcredit VARCHAR, phonenum VARCHAR, balance REAL )");
```

	name
	过滤
1	中国银行
2	中国工商银行
3	中国建设银行
4	中国农业银行
5	中国邮政储蓄...
6	中国民生银行
7	交通银行
8	招商银行
9	中信银行
10	平安银行
11	华夏银行
12	北京银行
13	广发银行

银行信息表 bankInfo，如右图所示

```
query.exec("CREATE TABLE bankInfo ( name VARCHAR PRIMARY KEY)");
```

数据库

SQLite 轻量级数据库保存所有账户信息

注册界面

卡号 19 位数字组成(只能输入数字), 已注册报错, 长度报错。

密码 6 位数字（只能输入数字），位数不够报错，两次输入不匹配报错。

身份证号 18 位（前 17 位只能输入数字，最后一位为数字或 x），位数不够报错

联系方式 11 位（只能输入数字）

规范输入用正则表达式实现，数值错误（如过大、过小会有判断）

登录界面

卡号 19 位数字组成(只能输入数字)，未注册报错，长度报错

密码 6 位数字（只能输入数字），密码不匹配报错

账户信息主界面

卡内余额会动态变化,两位小数

其它（卡号、银行名称、电话号、身份证号）是固定基本信息

设置界面（修改密码界面）

原密码输入错误, 新密码, 密码都是只能为 6 位数字

取款界面

只能输入浮点数或整数, 余额不足报错, 非法输入报错: 长度过长也会报错

存款界面

只能输入浮点数或整数, 单笔存款金额大于最大值 100000.0 报错, 非法输入报错:长度过长

注销

直接退出

管理员界面（方便演示）

查看所有银行账号信息（信息不可修改），可以删除账号

4. 数据结构

信息隐藏设计

```
typedef int errorType;//错误类型封装
typedef QString infoType;//信息类型封装
typedef float balanceType;//余额类型封装

//全局变量,保存当前登入账户信息
account *accountPtr = NULL;

//账户类定义
class account
{
private:
    static constexpr balanceType ADDMAX=100000.0;

private:
    infoType num;           //卡号
    infoType password;      //密码
    infoType bankname;      //银行名称
    infoType idcredit;      //身份证号
    infoType phonenum;      //联系电话
    balanceType balance;    //账户余额

public:
    //构造函数
    account();
    account(QString num, QString password, QString bankname, QString idcredit, QString phonenum,
balanceType balance = 0.0);
    ~account();

    //get、set 函数
    infoType getNum();
    void setNum(QString num);
    infoType getPassword();
    void setPassword(QString password);
    infoType getBankName();
    void setBankName(QString bankname);
    infoType getIDcredit();
    void setIDcredit(QString idcredit);
    infoType getPhoneNum();
    void setPhoneNum(QString phonenum);
    balanceType getBalance();
```

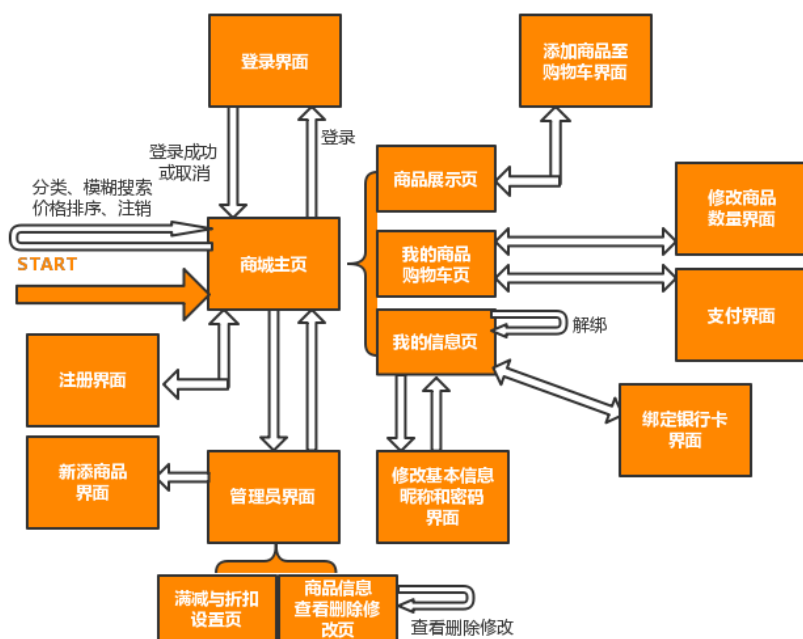
```

//存钱
bool deposit(balanceType add);
//取钱
bool withdraw(balanceType sub);
};

```

二. 电商平台的设计

1. 模块划分



2. 数据库表结构

名称	类型	架构
表 (6)		
cardinfo		CREATE TABLE cardinfo (accountNum VARCHAR,userPhoneNum VARCHAR)
accountNum	VARCHAR	'accountNum' VARCHAR
userPhoneNum	VARCHAR	'userPhoneNum' VARCHAR
cartinfo		CREATE TABLE 'cartinfo' ('id' INT, 'name' VARCHAR, 'type' VARCHAR, 'oriPrice' REAL, 'price' REAL, 'quantity' INT, 'userPhoneNum' VARCHAR)
id	INT	'id' INT
name	VARCHAR	'name' VARCHAR
type	VARCHAR	'type' VARCHAR
oriPrice	REAL	'oriPrice' REAL
price	REAL	'price' REAL
quantity	INT	'quantity' INT
userPhoneNum	VARCHAR	'userPhoneNum' VARCHAR
onSaleInfo		CREATE TABLE onSaleInfo (label VARCHAR,isStart VARCHAR,full INT,reduce INT)
label	VARCHAR	'label' VARCHAR
isStart	VARCHAR	'isStart' VARCHAR
full	INT	'full' INT
reduce	INT	'reduce' INT
productInfo		CREATE TABLE productInfo (id INTEGER PRIMARY KEY AUTOINCREMENT,name VARCHAR,type VARCHAR,originalPrice VARCHAR,inventory INT,rate REAL,other VARCHAR,description TEXT)
id	INTEGER	'id' INTEGER PRIMARY KEY AUTOINCREMENT
name	VARCHAR	'name' VARCHAR
type	VARCHAR	'type' VARCHAR
originalPrice	VARCHAR	'originalPrice' VARCHAR
inventory	INT	'inventory' INT
rate	REAL	'rate' REAL
other	VARCHAR	'other' VARCHAR
description	TEXT	'description' TEXT
sqlite_sequence		CREATE TABLE sqlite_sequence(name,seq)
name	TEXT	'name' TEXT
seq	TEXT	'seq' TEXT
userInfo		CREATE TABLE userInfo (phoneNum VARCHAR PRIMARY KEY,name VARCHAR,password VARCHAR)
phoneNum	VARCHAR	'phoneNum' VARCHAR
name	VARCHAR	'name' VARCHAR
password	VARCHAR	'password' VARCHAR

用户信息表 userInfo

```
query.exec("CREATE TABLE userInfo ("
           "phoneNum VARCHAR PRIMARY KEY,"
           "name VARCHAR,"
           "password VARCHAR );如下图所示
```

表(I): userInfo

	phoneNum	name	password
	过滤	过滤	过滤
1	13879619046	sad	123456
2	18811522176	Pei	123456
3	00000000000	pei	asdfghjkl
4	10000000000	spodjfpasoj	123abc

商品信息表 productInfo

```
query.exec("CREATE TABLE productInfo ("
           "id INTEGER PRIMARY KEY AUTOINCREMENT,"
           "name VARCHAR,"
           "type VARCHAR,"
           "originalPrice VARVHAR,"
           "inventory INT,"
           "rate REAL,"
           "other VARCHAR,"
           "description TEXT );
```

如下图所示：

表(I): productInfo

	id	name	type	originalPrice	inventory	rate	other	description
	过滤	过滤	过滤	过滤	过滤	过滤	过滤	过滤
1	1	C++Primer	BOOK	60	49	0.9	SomeBody	一本C++专业...
2	2	羽绒服	CLOTH	899.5	99	0.98	男士上装	来自NIKE出...
3	3	妙脆角	FOOD	6.9	999	0.1	休闲零食	闲暇时刻，尽...
4	4	MATE10	PHONE	4999.9	999	1.0	华为	华为mate10...
5	5	高效C++	BOOK	39.8	994	0.9	Smith	一本C++权威...
6	6	鸡腿	FOOD	5	78	0.1	肉类	新鲜鸡腿肉，...
7	7	iphoneX	PHONE	8999.9	999	1.0	苹果公司	全球最具科技...
8	9	数学之美	BOOK	26.8	99	0.9	吴军	计算机与数学...
9	10	java编程	BOOK	45.5	999	0.9	StephonC	计算机语言，...

满减信息与值表 onSaleInfo

```
query.exec("CREATE TABLE onSaleInfo ("
           "label VARCHAR,"
           "isStart VARCHAR,"
           "full INT,"
           "reduce INT);
```

	label	isStart	full	reduce
	过滤	过滤	过滤	过滤
1	FullThenRed...	YES	100	5

购物车信息表 cartInfo，用 userPhoneNum 索引用户的预购商品


```
query.exec("CREATE TABLE cartInfo ("
```

```

"id INT,"
"name VARCHAR,"
"type VARCHAR,"
"oriPrice REAL,"
"price REAL,"
"quantity INT,"
"userPhoneNum VARCHAR)");

```

如下图所示：

表(I):  cartInfo

	id	name	type	oriPrice	price	quantity	userPhoneNum
	过滤	过滤	过滤	过滤	过滤	过滤	过滤
1	13	花生油	FOOD	65.0	6.5	1	18811522176
2	7	iphoneX	PHONE	8999.9	8999.9	1	13879619046
3	3	妙脆角	FOOD	6.9	0.69	1	18811522176


银行卡信息表 cardInfo，用 userPhoneNum 索引用户已绑定的银行卡

```

query.exec("CREATE TABLE cardInfo ("
            "accountNum VARCHAR,"
            "userPhoneNum VARCHAR)");

```

如下图所示：

表(I):  cardInfo

	accountNum	userPhoneNum
	过滤	过滤
1	11111111111111111111	13879619046
2	11111111111111111112	13879619046
3	00000000000000000002	18811522176
4	11111111111111111111	18811522176

3. 功能实现

数据库

SQLITE 轻量级数据库保存所有用户信息与商品信息，和其他减免信息

商城界面页

展示全部商品 程序打开即首页

按品类查看 一共图书、服装、食物、手机四种分类，可分类查看

价格排序 按价格从高到低、从低到高、默认，三种排序方式。

模糊搜索 搜索产品文本信息即名称，描述，其它信息，部分匹配即可（验收时是精确匹配，将数据库过滤器“=”改为“LIKE”语句，就能实现模糊搜索，部分匹配，这一步是简单的）

未登入时，用户信息受保护，不会显示，并且相关按钮操作不会被使用（会有提示信息，提醒登入）

登入后，选中物品（若没选中会提示选中），点击添加购物车按钮，会弹出添加购物车对话框

添加购物车对话框界面

显示选中商品所有信息

选择添加数量，下方有 SpinBox，能够输入 1~1000（也可以设置得更大）的数量（不在该范围内的数字不会被

允许)，点击添加成功后，可在我的购物车界面看到。细节：如果，该物品已经在购物车中，则仍能添加成功，并且只增加物品数量。

登入界面

按下主界面中登入按钮则弹出，输入手机号（只能输入 11 位数字），密码（输入不限制），若密码匹配不成功，则会报错，登入失败，登入成功回到主界面，相应信息会被显示出来。

登入按钮与注销按钮是一个按钮，未登入时显示“登录”，已登录时显示“注销”。

注册界面

11 位手机号（只能输入 11 位数字），密码（不受限制，可以是数字和字母的组合），两次输入密码需要匹配，不匹配也会报错，昵称（不受限制，字符串），手机号若已经注册，也会提示已被注册。

我的购物车界面

未登入时，用户信息受保护，不会显示用户购物车内商品信息，并且相关按钮操作不会被使用（会有提示信息，提醒登入），但满减信息会在此页显示，若满减开始，则有 eg. 满 100 减 20 的显示，若满减没有开始，则会显示 XXXX 表示满减活动未开。

登入后，用户购物车内商品信息被读取并显示，下方，是购物车内所有物品的原价综合，经过折扣后的现价综合，满减信息，应付金额显示。旋转一件物品（没选中则会提示需选中），能够按下修改物品数量按钮，弹出修改物品数量对话框；也可按下删除按钮，则删除选中物品（没选中会有提示信息）。信息更改后，会及时在界面中动态更新。可以选中多个物品，点击提交订单后，弹出支付页。

修改物品数量对话框

表现与添加购物车对话框相同，但数量若选择为 0，则代表删除操作，会将此商品从购物车中移除，其它情况，则改变物品在购物车数量。此时下方的相关价值总额会同步改变。

我的信息界面

展示我的基本信息，昵称与注册手机号，还有银行卡栏，显示所有已经绑定的银行卡。有修改信息按钮，能够修改用户基本信息（昵称，密码），昵称修改没有限制，密码需要输入正确原密码，新密码输入两遍匹配成功，密码可以是字母或数字或其它符号。

管理员界面

展示所有商品信息，就像商城首页一样，展示所有商品信息。

新添商品，弹出添加商品对话框。

删除商品，选中商品（没选中会有提示），询问是否删除，确认，则将商品从数据库移除，并且其它界面都会同步显示。

修改商品信息，因为使用了 model，商品显示表格与数据库绑定，可以点击商品展示表格，直接编辑修改，自动同步到数据库中，库存量，简介，单件折扣率，原价都能实时更新!!!

满减信息

进入满减信息界面，可以看到当前的满减信息状态，如果开始则 checkBox 被勾选中，并且把满多少，减多少显示在 LineEdit 中。若没开始则 checkBox 不被勾选。修改满减信息，并点击修改满减状态，可对满减内容进行调整，可以停止满减，也可以修改满值与减值，两者都只能输入（0~100000）的整数（正则表达式实现），满值必须大于减值，否则会报错。

满减活动开始，则在购物车页和支付页面会有满减信息的显示。若没开始，则没有满减信息显示。

折扣信息

与满减信息同页，可以设置四种商品类的品类折扣信息，填写必须是（0.0~1.0）的浮点数（正则表达式），若输入超出范围，或非法长度（超过10位），都会报错，并不修改当前此类折扣信息，四类的折扣信息判别是分开的，并且为空的时候是默认不修改。若设置某类折扣成功，则会在商城主界面，购物车界面同步折扣信息，并且现价也将被更新。

管理员新添商品对话框界面

管理员填写商品的名称，品类、原价，库存，折扣率，产品描述，其它信息。其中，商品名称不能为空、原价（正则表达式，浮点数、输入过长>10位，被视为非法输入，报错）、库存（spinBox 1~10000）、折扣率（spinBox 0~1.0），其它信息和产品描述可以为空。确认添加后，插入数据库，其它界面中也会及时更新。

4. 数据结构

```
//全局变量
bool isLogin = false;           //是否登录
QString userPhoneNum = "";      //用户账号(手机号)
int Full = 0;                   //满减信息，满值
int Reduce = 0;                 //满减信息，减值
bool isFRstart = false;        //满减信息，满减是否开始
bool isPayed = false;          //是否完成付款

// 产品基类 product
class Product
{
public:
    //商品种类
    static const int BOOK = 1;
    static const int CLOTH = 2;
    static const int FOOD = 3;
    static const int PHONE = 4;
private:
    QString name;               //产品名称
    QString description;        //产品描述
    float originalPrice;        //原价
    int inventory;              //库存量

public:
    //构造函数
    Product();
    Product(QString s1, QString s2, float f, int i);

    //get、set 函数
    QString getName();
```

```
void setName(QString name);

QString getDescription();
void setDescription(QString description);

float getOriginalPrice();
void setOriginalPrice(float originalPrice);

int getInventory();
void setInventory(int inventory);

virtual float getPrice();//虚函数，用于计算具体产品的价格
virtual QString getType();//获得产品的品类信息
virtual QString toString();//获得不同品类商品的其它信息
};

//图书基类、继承 product, 其它类似, 就不贴了
class Book : public Product
{
private:
    static const int type = BOOK;    //BOOK 类
    float rate;                      //折扣率
    QString author;                  //作者名

public:
    //构造函数
    Book();
    Book(QString name, QString description, float originalPrice, int inventory,
          float rate, QString author);

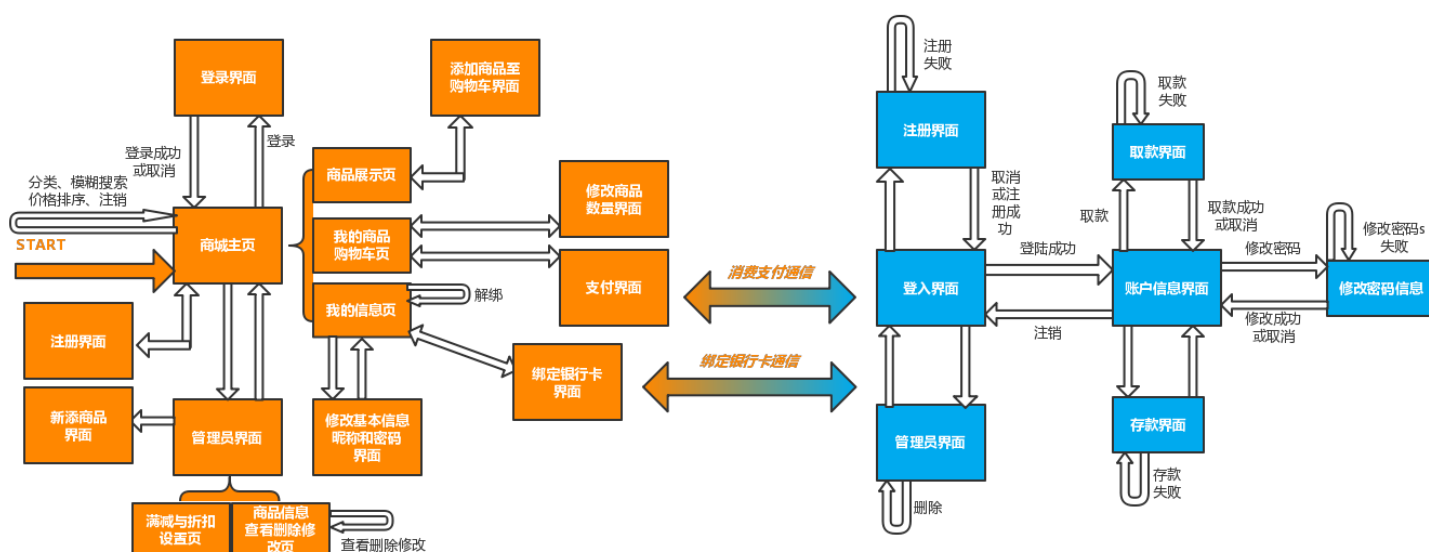
    //get、set
    void setRate(float rate);
    virtual QString getType();
    virtual float getPrice();
    virtual QString toString();
};
```

三. 电商平台网络版

在之前的基础上添加了，支付窗口，银行卡绑定窗口，银行服务器，实现了网络通信，采用 TCP/SOCKET 机

制，实现进程之间的通信。新增类结构在数据结构子目录中呈现：

1. 模块设计



2. 功能实现

Store 与 Bank 连接

错误可能:未找到服务器错误，与服务器连接断开错误

绑定银行卡界面

输入银行卡号（只能输入 19 位数字，正则表达式），与对应密码（只能输入 6 位数字，正则表达式），发送请求，若银行卡号被找到，并且相应密码匹配，则绑定成功。

有错误发生情况：银行卡号长度不够，银行卡号未被找到，密码匹配错误，都会有相应的错误提示，并且绑定失败。若卡号已被绑定，会有提示信息，不会再次重复绑定。

绑定成功，则数据库更新，我的信息中银行卡栏更新信息，能够同步。

支付界面

在购物车中选择一个或多个商品（若没有选中，会提示您需要选择支付商品），点击“提交订单”

此时会有商品库存判断，如果库存小于所需购买数量，或商品已经下架，都会提交订单失败。这里是个重要的判断，而不在加购物车的时候判，因为不知道管理员何时添加库存。

提交成功，进入商品结算界面，上方是所选商品的详细信息，包括商品基本信息，还有数量，现价。下方是当前物品的所有价值总和，与满减信息，还有应付信息。

右下方是银行卡选项，选择已经绑定的银行卡，若此时您没有绑定银行卡，会有提示信息，提示您需要绑定一张银行卡，并且下方按钮操作此时无效。

选择一张一行卡，填写密码（只能 6 位数字（正则表达式）），发送支付请求

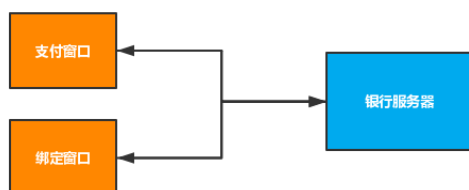
若密码错误，会返回密码错误信息。

若余额不足，会返回余额不足信息。

若成功，则完成支付操作，则银行会更新余额信息，相关界面也会更新，此时用户购物车中已经支付了的物品将被移除，相关信息也会及时更新。

3. 数据结构

电商平台设计，应继承于第二题中购物平台的设计，并添加了，支付窗口，银行卡绑定窗口，银行服务器，采用 TCP/SOCKET 机制，实现进程之间的通信。新增类结构在下面呈现：



```
//全局变量
bool isPayed = false;           //是否完成付款

//支付窗口类
class PayDialog : public QDialog
{
    Q_OBJECT
private:
    QVector<Product *> productList;           //待付款商品指针容器，能够存储所有品类商品指针
    QVector<int> quantities;                 //待付款商品数量容器，记录商品购买数量

public:
    explicit PayDialog(QWidget *parent = 0, QVector<Product *> productItems = QVector<Product
*>(), QVector<int> quant = QVector<int>());
    ~PayDialog();

    static const int SUCCESS = 1;           //支付成功
    static const int NOSEVER = 2;           //为找到服务器
    static const int DISCONNECT = 3;        //失去连接
    static const int WRONGNUM = 4;          //卡号错误
    static const int WRONGPW = 5;           //密码错误
    static const int LACKMONEY = 6;         //余额不足

private:
    Ui::PayDialog *ui;

    void productItemShow();
```

```
static int askBank(QString sendStr);

signals:
    void paySignal(bool isPay);
private slots:
    void payConfirmButton_clicked();
    void returnButton_clicked();
};

/* 定义了 TCP 的服务器线程 bankServer*
 * 通过继承 QTcpServer，在有客户端要建立连接时，
 * 建立 TcpSocket 线程与客户端建立 TCP 连接 */
class BankServer : public QTcpServer
{
    Q_OBJECT
private:
    QTcpSocket *clientConnection;
public:
    BankServer(QObject *parent=0);

signals:
    void paySignal();                //支付成功信号，传给主函数，以更新首页余额信息

private slots:
    void acceptConnection();        //当有新客户端请求建立连接时（即 newConnection() 被唤醒时，建立连接
    void readClient();              /* 建立连接后，如果有信息传入（即 readyRead() 信号被唤醒）
                                     * 开始读取信息并进行处理
                                     * 处理完信息后将处理结果再发回客户端 */
};

//支付窗口类
namespace Ui {
class PayDialog;
}

class PayDialog : public QDialog
{
    Q_OBJECT
private:
    QVector<Product *> productList;    //待付款商品指针容器，能够存储所有品类商品指针
    QVector<int> quantities;          //待付款商品数量容器，记录商品购买数量

public:
    //构造函数
```

```
explicit PayDialog(QWidget *parent = 0, QVector<Product *> productItems = QVector<Product
*>(), QVector<int> quant = QVector<int>());
~PayDialog();

static const int SUCCESS = 1;           //支付成功
static const int NOSEVER = 2;           //为找到服务器
static const int DISCONNECT = 3;        //失去连接
static const int WRONGNUM = 4;          //卡号错误
static const int WRONGPW = 5;           //密码错误
static const int LACKMONEY = 6;         //余额不足

private:
    Ui::PayDialog *ui;
    void productItemShow();              //待支付物品全部信息展示
    static int askBank(QString sendStr);  //向银行发送连接请求，并传递支付信息

private slots:
    void payConfirmButton_clicked();      //确认支付按钮被按下
    void returnButton_clicked();          //取消支付
};

//银行卡绑定窗口类
namespace Ui {
class accountManageDialog;
}

class accountManageDialog : public QDialog
{
    Q_OBJECT

public:
    explicit accountManageDialog(QWidget *parent = 0);
    ~accountManageDialog();

    static const int SUCCESS = 1;          //绑定成功
    static const int NOSEVER = 2;          //为找到服务器
    static const int DISCONNECT = 3;       //失去连接
    static const int WRONGNUM = 4;         //卡号错误
    static const int WRONGPW = 5;          //密码错误

private:
    Ui::accountManageDialog *ui;
    static int askBank(QString sendStr);    //向银行发送连接请求，并银行卡绑定信息

private slots:
```

```
void bindConfirmButton_clicked();           //绑定确认按钮
void returnButton_clicked();               //取消返回按钮
};
```

四. 设计模式应用

商城进程相对复杂，银行进程相对简单，下面三个设计模式：单例模式、简单工厂模式、观察者模式将在电商平台中使用到。

单例模式 singleton

对于全局控制信息，它们对于一个独立进程来说，是只能被唯一构造一次的，并且要求需要易扩展性，所以我们这里构造了一个 GlobalInfo 类，以便于对全局控制得掌握，具体构造如下：

```
//singleton pattern 单例模式
//存储商城的全局信息，如是否登录，用户账号（手机号），满减信息，折扣信息

#include<QString>

class GlobalInfo
{
private:
    //全局变量
    static GlobalInfo* s_instance;
    bool isLogin;           //是否登录
    QString userPhoneNum;   //用户账号(手机号)
    int Full;               //满减信息，满值
    int Reduce;             //满减信息，减值
    bool isFRstart;         //满减信息，满减是否开始
    bool isPayed;           //是否完成付款

    GlobalInfo(bool login = false, QString num = "", int full = 0,
               int reduce = 0, bool fr = false, bool isPay = false)
    {
        isLogin = login;
        userPhoneNum = num;
        Full = full;
        Reduce = reduce;
        isFRstart = fr;
        isPayed = isPay;
    }

public:
    static GlobalInfo* instance(); //获取实例
```



```
//get、set 函数

bool getIsLogin();
void setIsLogin(bool login);

QString getUserNum();
void setUserNum(QString num);

int getFull();
void setFull(int full);

int getReduce();
void setReduce(int reduce);

bool getIsFRstart();
void setIsFRstart(bool fr);

bool getIsPayed();
void setIsPayed(bool isPay);
};
```

简单工厂模式 Simple Factory

对于我们电商平台，最重要的对象就是商品类，这里我们平台主要有四个基本类，Book、Cloth、Food、Phone，它们都继承于抽象类 Product，而它们之间主要依靠 Type 属性区分，所有通过简单工厂模式能很简易的实现它，具体构造如下：

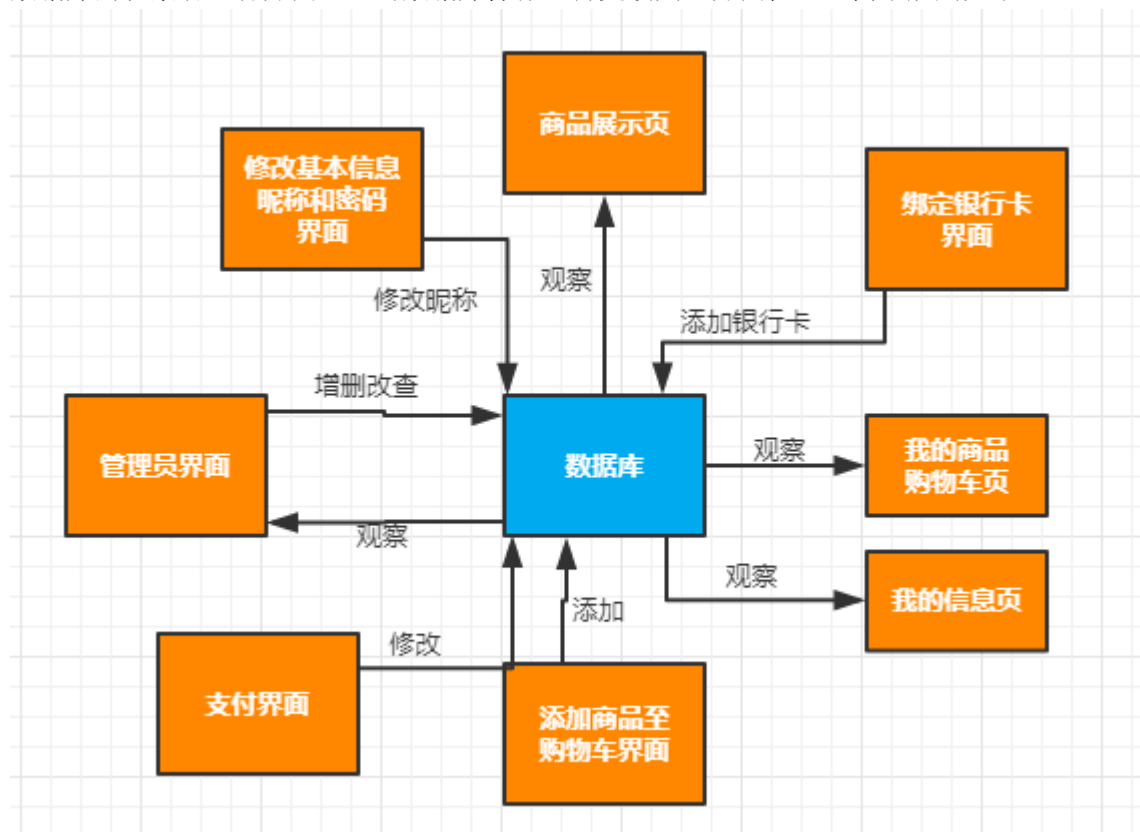
```
//商品构造工厂
class ProductFactory
{
public:
    ProductFactory();
public:
    static Product *CreateProduct(int type);
};

//根据类型不同，建造相应商品实例
Product* ProductFactory::CreateProduct(int type)
{
    Product *p;
    switch(type)
    {
    case Product::BOOK:
        p = new Book();
        break;
    case Product::CLOTH:
        p = new Cloth();
    }
```

```
        break;
    case Product::FOOD:
        p = new Food();
        break;
    case Product::PHONE:
        p = new Phone();
        break;
    default:
        p = NULL;
        break;
}
return p;
}
```

观察者模式 Observer

电商平台构造一共有多个界面切换，商品主页，购物车界面，用户基本信息界面，管理员界面，添加商品至购物车界面，修改购物车内商品界面，绑定银行卡界面，支付商品界面等。它们之间互相影响，总得来说，它们都是数据库的观察者。各界面通过对数据库操作，再更改其它界面信息，简单规则如下：



//观察者模式

```
class Observer
{
public:
    Observer();
public:
    virtual void Update() = 0;
}
```

```
};  
  
class Subject  
{  
public:  
    Subject();  
public:  
    virtual void registerObserver(Observer *) = 0;  
    virtual void removeObserver(Observer *) = 0;  
    virtual void NotifyObservers() = 0;  
};
```

五. 程序 UI 设计

在完成所有基本功能的基础上，我还添加了良好的图形界面设计（因私人原因，致敬了一下科比退役），银行主要用蓝色系，电商则采用橙色系，我认为我的程序一大亮点就是很好的用户界面，功能分布也按照惯例需求，也遵循 GUI 设计的基本原则，能够使初次使用此程序的人员就能轻松通过图形界面来使用。程序 UI 可以初步通过下面几张截图显示，后面有更为详尽的程序运行截图，请前往查看，或者直接运行我的程序来使用即可。



六. 程序课程设计总结

本次程序课程设计为：一个完整的网络电商平台（银行门户+商城）。

本人对这次课程设计的成果也比较满意，虽然距离实际应用还是有很大差距，但我还是能够把我所想到的尽可能的完成了实践。按照课程所学内容，规范自身代码，写出易读易懂易扩展的代码。并且自己学习了设计模式，不过使用 C++ 去完成设计模式，这还是第一次，所以并不是很熟练，只是完成了几个较为常用简单的设计模式，但也对设计模式有了更加深刻的了解。

本次自选课程设计，是按照三部完成的，并且徐徐渐进：首先，完成银行门户的设计，主要学习了数据库 SQLITE，与 QT 图形化界面设计，并且将银行卡视为一个类以实现功能。其次，完成单独电商平台设计，着重于类的继承与多态的性质，对类的私有继承，公有继承，保护继承都有了更深入的理解，还有更加熟练了数据库的各个操作，什么时候使用数据库操作，什么时候使用类实现，都有较为合理的判断，让信息传递与处理更为流畅易懂。最后完成网络通信，实现电商平台网络版功能，自学了 TCP/SOCKET 相关操作后，实现电商与银行间的通信，模拟了商品的购买与支付。

七. 不足之处与优化方向

程序虽然极大地追求与实际情况相结合，大部分已经做到，但仍是存在些许缺憾，如电商物品信息主要以文本形式展现，而实际淘宝则是图文形式呈现，更为直观，还有银行是不是需要有利息的功能？还有电商是不是应该有购买记录？等，这是今后可以改进的地方，因为时间有限，所有不能完成，也是一个小缺憾吧。

进程互斥问题

首先由于一个进程对应与一个账号用户的操作，由于其中界面操作不可能同时进行（比如同时按下两个按钮），所以可以视为线程安全的。但，我们考虑到多个电商平台进程对应多个用户账户，假定这多个用户账户共用一张银行卡（现实里也是可能出现的），这样他们可能同时使用一张银行卡付款，就需要同时对此银行卡进行更改操作，而这是绝对不被允许的，就相当于“读写者问题”中的“写-写”互斥，因为多个用户去同时绑定一张银行卡，实际上只是做只读操作不会有排斥作，所以相当于“读-读”允许。所以这里需要考虑到怎么解决“写-写”问题，这里给出我的思路：

对于银行服务器来说，每当接收到一个支付请求，则需要对当前支付请求中使用的银行卡加锁，表明当前某个银行卡正在被写操作，其它进程将不允许对此银行卡进行读或写，直到此银行卡解锁。

这里可以使用 QT 中的 `QMutexLocker` 实现相关内容操作，首先对每个已有的银行卡分配一个互斥变量，可以使用 `unordered_map` 产生每个互斥变量 `Mutex` 与一个银行账户号码的一一对应。然后根据当前支付请求，在请求时调用 `mutex.lock()`，锁住，在完成修改操作后，再将 `mutex.unlock()`，使得其他进程能够访问。借此完成“写-写”，“读-写”的互斥性，保证进程的安全性。

八. 程序运行截图



登入界面 1

登入界面 2

账户界面



取款界面 1

取款界面 2

存款界面 1



存款界面 2

设置，修改密码界面



管理员界面



商城主界面



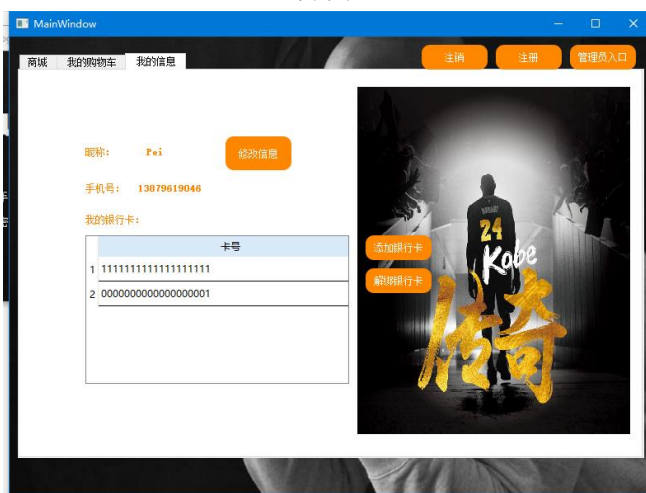
注册界面



登入界面



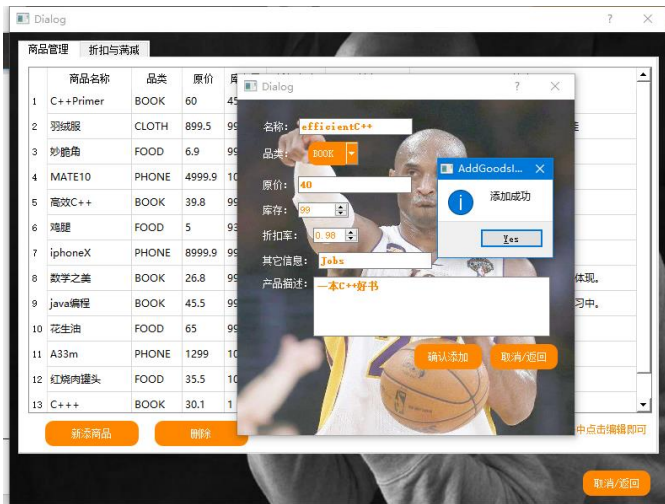
我的购物车界面



我的信息界面



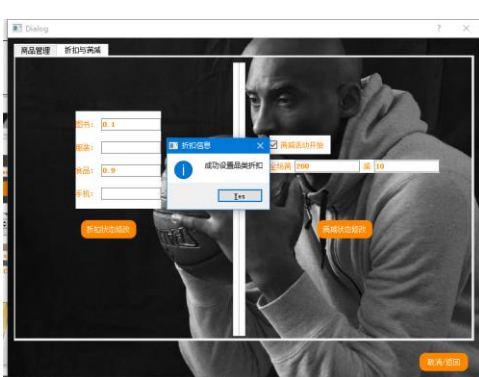
添加至购物车界面



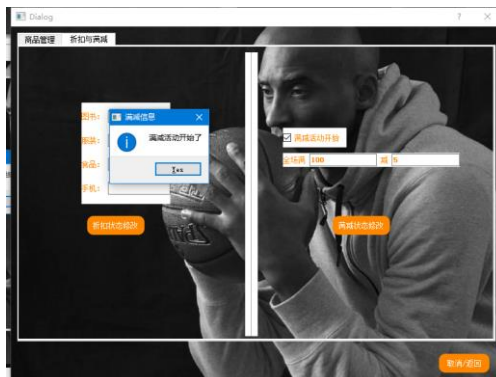
管理员新添商品



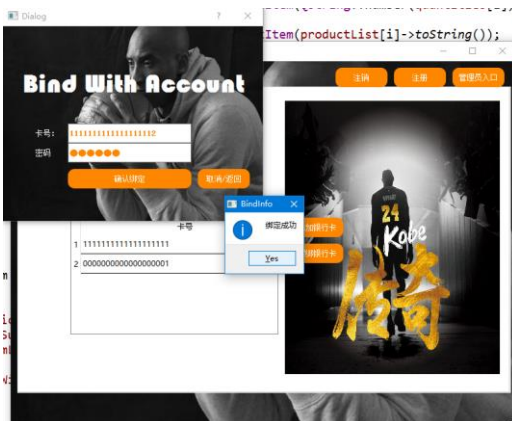
管理员界面 2



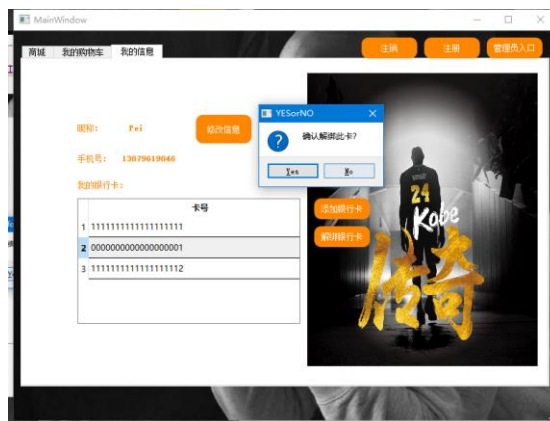
管理员设置品类折扣



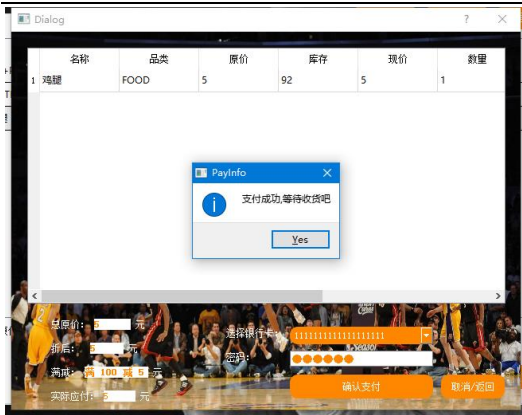
管理员设置满减信息



用户绑定银行卡



用户解绑银行卡



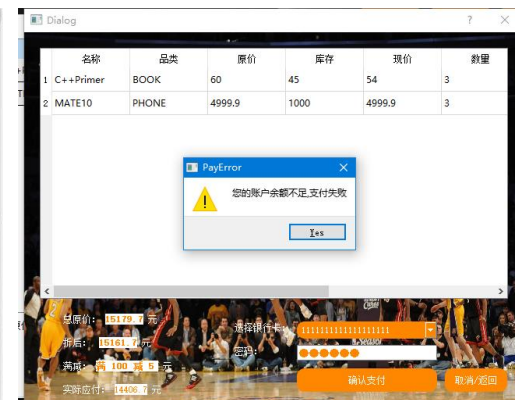
用户购买支付成功界面



用户提交订单失败



修改购物车内商品数量



余额不足，支付失败

九. 源码附件

详见，电子源码压缩包附件。

详见源代码文件夹中 QT 工程文件，可执行文件夹下是可运行程序。

感谢阅读。