武汉纺织大学

面向对象课程设计

超市商品管理系统

学　　院：计算机与人工智能学院

班　　级：计科 12101 班

姓　　名：刘家麒

学　　号：2107250127

指导老师：崔树芹

成　　绩：

完成日期：2023 年 1 月 1 日

目录

1 需求分析

设计一个超市管理系统，实现“新增商品”、“显示所有商品”、“收银台”、“销售统计”、“商品管理”、“退出系统”功能，具体要求如下：

1.1 用户登陆

当程序运行时，显示“超市商品管理系统”，提示用户依次输入用户名及其密码，当用户名及密码不正确时，显示提示信息“密码不正确，请重新输入”，当用户输入正确的账号密码时才能正常进入系统。若是初次使用该系统则会提示设置初始密码。另实现修改密码功能，当用户点击修改密码时，提示输入旧密码，正确后会提示输入新密码并重新登录，若旧密码输入错误，则不能设置新密码。

1.2 主菜单

当用户名及密码正确时，加载相应的文件数据后进入系统主界面，有如下功能：

1、新增商品
2、显示所有商品
3、收银台
4、销售统计
5、商品管理
6、退出系统

当用户选择对于的按钮时，会跳转到相应的界面。特别的，在点击退出系统时，会先将数据保存到磁盘中，再退出系统。在主菜单中，还会显示当前的时间。

1.3 各项功能

1.3.1 新增商品

当点击新增商品按钮时，显示选择商品类型的选择界面，用户选择新商品类型并确定后，进入信息输入的窗口，在用户依次输入或选择商品的各种信息并且点击确定后，系统开始检测是否与已有商品重复，若无重复则添加成功，若有重复则提示商品已存在。商品重复的具体判定规则为：若一个商品与当前库存中商品的名字和品牌都相同则为同一商品。若用户在选择商品类型或输入窗口中点击取消，则放弃本次添加操作。

1.3.2 显示所有商品

当点击显示所有商品按钮时，系统跳转到显示所有商品的页面。在该界面可选择商品的排序方式，可根据价格、库存数量、名称、销量四中方式进行排序，同时可选择排序的升降序模式。默认为价格升序排序。用户可在下拉选择框内选择需要的排序方式并点击显示即可按照所选择的模式进行排序。

1.3.3 收银台

当在主界面点击收银台按钮时，系统跳转到收银台的页面。系统用户可输入要购买的商品的商品号和数量进行购买商品，点击添加按钮时，会进行检测，若商品不存在或购买数量大于选择商品的库存数量均会进行提示，反正则会进行添加操作，更新总金额，在购物车表中显示所购买商品。用户可多次添加需要购买的商品，直到点击结算完成该次购买。结算后，会显示当前订单的总金额以及交易号。交易号是购买的唯一标识符。

1.3.4 销售统计

当点击销售统计按钮时，系统跳转到销售统计的页面。用户可输入订单号并查询该笔订单的信息。点击查询时，若输入的订单号不合法（不符订单号格式、没有该笔订单、订单已经被退货等），非合法订单号。若输入正确则会将该笔订单所购买的所有商品的基本信息和购买数量以及购买价格显示到表格中，并显示该笔订单的总金额。在输入订单号正确的情况下，可对该订单进行退货操作，即将该笔订单的所有商品全部退去，也可点击修改购买数量修改某一商品的购买数量。若为增加数量，增量大于购买数量时会提示库存不足，无法修改。该模块还会显示当前超市进货的总成本，以及售卖商品的总营业额。

1.3.5 商品管理

当在系统主界面点击销售管理按钮时，系统跳转到商品管理的页面。商品管理页面右下角有一个查询方式选项，可以根据商品号、商品名、品牌、类型查询，也可以选择全部，显示所有商品的属性。查询过后，页面中下方有个删除商品按钮，可以选择相应的商品进行删除操作。在此界面，双击某个商品的商品名，会弹出商品信息修改窗口，可以对该商品的信息进行修改。

1.3.6 退出

在主界面选择退出界面后，系统先会将经过用户添加和修改的数据保存到对应的文件中，最后才退出程序。

2 系统设计

2.1 用户用例图

系统由管理员管理，用户用例图如下图 2-1 所示。

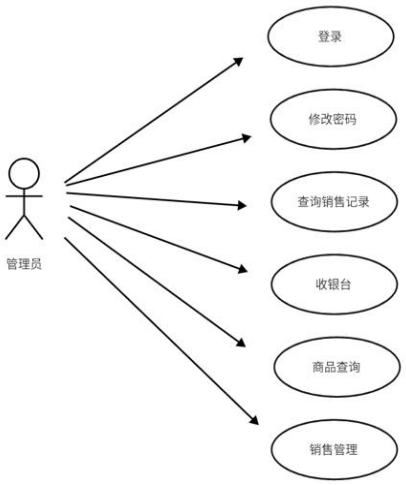


图 2-1 用户用例图

2.2UML 类图（Class Diagram）

对于商品，共有如下 5 个类。

- 商品基类 Commodity：有所有商品的共性。
- 饮料类 Beverage：继承商品基类，并多了属性容量和饮料种类。
- 化妆品类 Cosmetic：继承商品基类，并多了属性重量和化妆品种类。
- 日常用品类 Daily：继承商品基类，并多了属性重量和日常用品种类。
- 食品类 Food：继承商品基类，并多了属性重量和食品种类。

对于管理员，共有如下四个类。

- 管理基类 Manager：有所有管理员的属性，用静态的 vector<commodity *>保存所有商品信息，并便于管理。
- 仓库管理类 RepManager：继承自管理基类，拥有添加三种商品基类的子类方法，有加载库存及保存库存的方法。并有一个静态 map 成员，可以根据商品名存储商品信息地址，因此可以根据这个 map 成员以商品编号搜索商品。仓库管理类还可以通过商品号删除此商品的库存。可以
- 销售类 Saler：继承自管理基类 Manager，负责商品的销售记录，查询订单号，可以读取、显示和保存交易记录。
- 排序类 Sorter：继承自管理基类 Manager，负责商品信息的排序。

各类的结构及类之间的关系如图 2-2 所示：

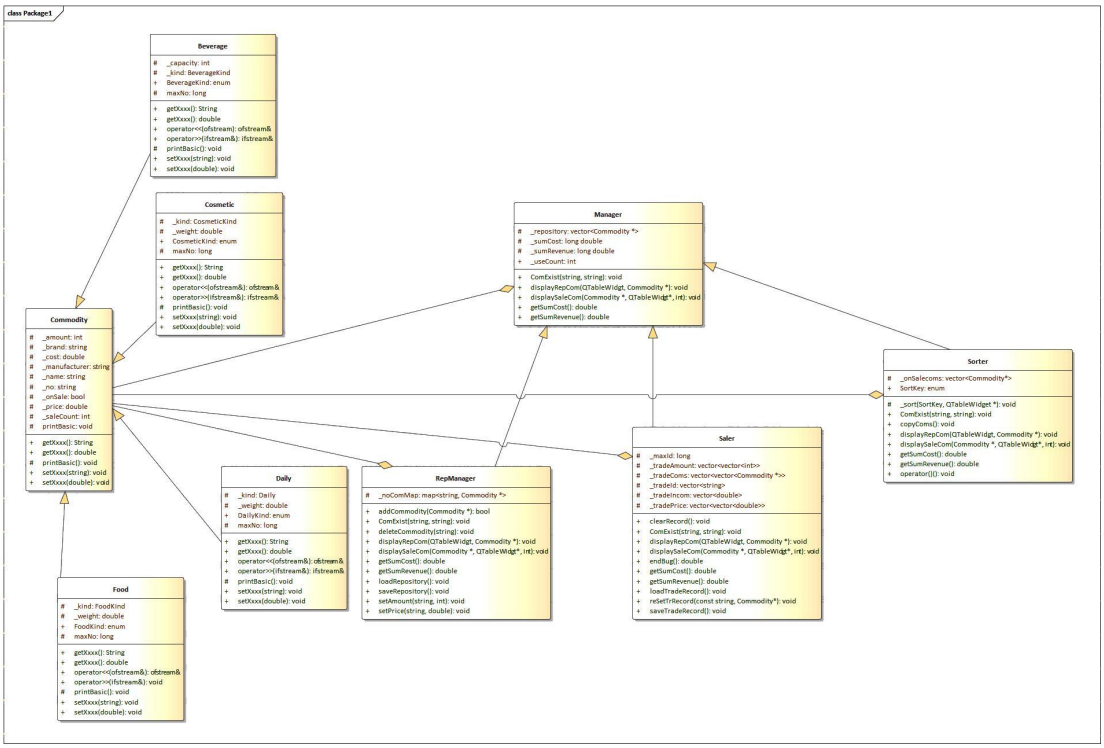


图 2-2 用户功能类图

2.3 UML 活动图（Activity Diagram）

2.3.1 登陆

第一次登录操作会提示用户初始化密码，初始化密码成功后提示用户登录，用户名为 admin，点击登录需要验证用户名，密码，只有当用户名和密码同时存在即用户存在的情况下才能成功登录系统，否则会提示错误信息，具体流程如图 2-2 所示。

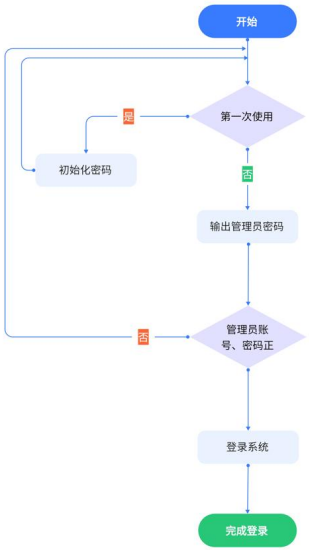


图 2-2 登录流程图

2.3.2 收银

当用户在窗口选择了收银台后，会进入收银台界面，首先，用户在对应的输入框中输入商品号，如果商品号存在，则会获取相应商品的信息，如果不存在，则提醒用户输入有误，并重新输入。获取完商品信息后，再输入购买数量，如果购买数量大于库存，则会提示用户商品数量不足，请再次输入。在没有点击结算按钮之前，是可以多次查询商品并输出商品信息的，即一次购买可以购买多种商品，若点击了结算，则会计算总价，并将数据保存在 Saler 类中，在程序结束后会将订单数据存入文件当中。收银功能的活动图如图 2-3 所示。

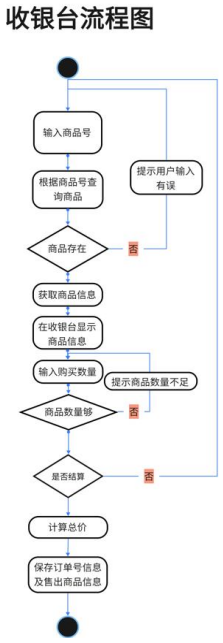


图 2-3 收银功能活动图

3 系统实现

3.1 项目结构

项目文件结构如图 3-1 所示。

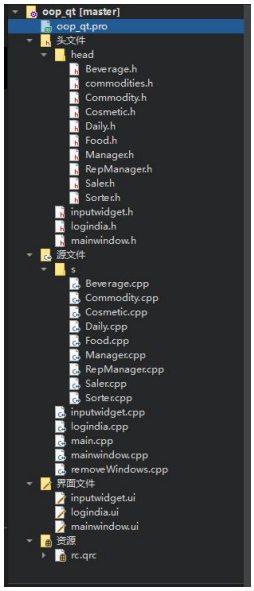


图 3-1 项目文件结构

3.2 窗体类

主要包含窗体类的头文件、源文件以及描述图形化界面设计的界面文件。

3.2.1 登录对话框类

主要包含登录对话框类 LoginDia 的定义，槽函数和成员变量的声明以及实现。该窗口首次打开程序会提示用户初始化密码，将密码写入文件“password.pwd”中，然后每次登录会弹出登陆框，如图 3-2，提示用户登录，对用户的输入的账号密码进行判定，并处理用户修改密码的请求。

图 3-2 登录窗口

类内包含对外接口 canLog，在窗口被关闭时，可使用该接口确定是否能够进入系统。窗口的构造函数如图 3-3 所示，若文件中密码存在先读入文件，否则提示输入初始密码。修改密码按钮释放信号的槽函数如图 3-4 所示，先要求输入旧密码。若输入正确则提示输入新密码完成修改，若输入错误则进行提示。

```
LoginDia::LoginDia(QWidget *parent) :
QDialog(parent),
    ui(new Ui::LoginDia)
{
    _canLog=false;
    ui->setupUi(this);
    this->setWindowIcon(
        QIcon(":/new/prefix1/rc/n133f35jvek.jpg"));
    std::ifstream ifs("passWord.pwd");
    if(ifs.is_open())
    {
        std::string buff;
        ifs>>buff;
        _passWord = buff.c_str();
        ifs.close();
    }
    else  //不存在密码文件
    {
        bool  ok=false;
        _passWord = QInputDialog::getText(this,"设置密码","请输入新密码:"
                                          ,QLineEdit::Password,"",&ok);
        if(!ok)  this->close();
        std::ofstream ofs("passWord.pwd");
        ofs<<_passWord.toStdString();
        ofs.close();
    }
}
```

图 3-3 登录窗口的构造函数

```
void LoginDia::on_chPwdEdit_released() //改密码
{
    bool ok;
    auto oriPwd = QInputDialog::getText(this,"输入","请输入旧密码:",
                                        QLineEdit::Password,"",&ok);
    if(ok)
    {
        if(oriPwd==_passWord)
        {
            auto newPwd = QInputDialog::getText(
                this,"输入","请输入新密码:",QLineEdit::Password,
                "",&ok);
            if(ok)
            {
                _passWord = newPwd;
                QMessageBox::information(this,
                                         tr("提示"),tr("修改成功! "));
            }
        }
        else
        {
            QMessageBox::critical(this,
                                  tr("错误"),tr("原密码错误"));
        }
    }
}
```

图 3-4 登录窗口的修改密码实现

3.2.2 商品信息输入窗口类

主要被包含对该窗口类 InputWidget 的定义以及槽函数的声明和实现。该窗口用途为输入商品的信息，并根据输入的信息完成对商品的添加和修改。通过构造函数的参数判断需要输入的商品类型和操作类型（新增、修改），提高了代码的复用性。

其构造函数需要提供要输入的商品的类型，窗口的父对象，以及要修改的商品的指针。商品指针提供默认参数 nullptr，若不提供参数则为新增商品，提供参数则为修改商品信息，具体实现如图 3-5 所示。确定按钮释放信号的槽函数如图 3-6 所示。通过对商品指针的判定确认当前操作为新增或修改。新增时，要对商品进行查重操作，确认无重复后方可添加。

```
InputWidget::InputWidget(char classCh,QWidget *parent,Commodity* com) :
    QWidget(parent),
    ui(new Ui::InputWidget),rm(),_classCh(classCh),_com(com)
{
    ui->setupUi(this);
    QStringList kList ;
    switch(_classCh)
    {
    case'F':
        kList = {tr("熟食"), tr("生食"),tr("袋装食品")};
        break;
    case'C':
        kList={tr("美容"),tr("清洁"),tr("护肤")};
        break;
    case 'B':
        kList={tr("普通饮料"),tr("酒"),tr("碳酸饮料")};
        ui->weiOrCapaLab->setText(tr("商品容量"));
        break;
    case 'D':
        kList={tr("床上用品"),tr("厨房用品"),tr("电器"),tr("常用品")};
    }
    ui->kindBox->clear();
    ui->kindBox->addItems(kList); //设置下拉选择框内容
    if(com)
    {
        ui->kindBox->setCurrentIndex(com->getKind());
        ui->nameEdit->setText(tr(com->getName().c_str()));
        ui->brandEdit->setText(tr(com->getBrand().c_str()));
        ui->manuEdit->setText(tr(com->getManufacturer().c_str()));
        ui->priceBox->setValue(com->getPrice());
        ui->costBox->setValue(com->getCost());
        ui->weightOrCapaBox->setValue((com->getWeiOrCapa()));
        ui->amountBox->setValue(com->getAmount());
    }
}
```

图 3-5 商品信息输入窗口类构造函数实现

```cpp
void InputWidget::on_okBtn_released() { //录入信息
    int kind = ui->kindBox->currentIndex();
    auto name = ui->nameEdit->text().toStdString();
    auto brand = ui->brandEdit->text().toStdString();
    auto manufacturer=ui->manuEdit->text().toStdString();
    double price = ui->priceBox->value();
    double cost = ui->costBox->value();
    double weiOrCapa=ui->weightOrCapaBox->value();
    int amount = ui->amountBox->value();
    if(_com){ //修改
        std::string no = _com->getNo();
        rm.setPricre(no,price);
        rm.setAmount(no,amount);
        _com->setName(name);
        _com->setCost(cost);
        _com->setManufacturer(manufacturer);
        _com->setBrand(brand);
        _com->setWeiOrCapa(weiOrCapa);
        _com->setKind(kind);
    }
    else { //新增
        if(rm.comExist(name,brand)) {  //查重 品牌+商品名 相同为重复
            QMessageBox::information(this,tr("提示"),tr("该商品已存在! "));
            this->close();  return;
        }
        switch (_classCh) {
        case 'F': //food
            rm.addFood(std::move(name),std::move(brand),std::move(manufacturer),
                       price,cost,amount,weiOrCapa,(Food::FoodKind)kind);
            break;
        case 'C':
            rm.addCosmetic(std::move(name),std::move(brand),std::move(manufacturer),
                           price,cost,amount,weiOrCapa,(Cosmetic::CosmeticKind)kind);
            break;
        case 'B':
            rm.addBeverage(std::move(name),std::move(brand),std::move(manufacturer),
                           price,cost,amount,weiOrCapa,(Beverage::BeverageKind)kind);
            break;
        case 'D':
            rm.addDaily(std::move(name),std::move(brand),std::move(manufacturer),
                           price,cost,amount,weiOrCapa,(Daily::DailyKind)kind);
            break;
        }
    }
    QMessageBox::information(this,tr("提示"),tr("成功! "));
    this->close();
}
```

图 3-6 确定添加按钮槽函数实现

3.2.3 主窗口文件

主要被包含对主窗口类 MainWindow 的定义以及槽函数的声明和实现。该类基于 qt 自动生成的文件，并加入了 RepManger 和 Saler 两个类的子对象，使主窗口操控数据。主窗口拥有的控件及关系如图 3-7 所示。其含有一个堆叠窗口类 QStackedWidget 的对象，包含了主菜单页面以及各个功能的页面。通过各个按钮的释放信号的槽函数实现各个窗口之间的跳转与切换，其实现在 removeWindows.cpp 中。

其构造函数会完成窗口的初始化操作（设置应用程序图标、初始化时间显示、设置表格

参数等），并通过子对象 rm 和 saler 读取磁盘中的商品库存信息以及交易记录信息。主窗口析构时，析构函数会通过子对象将当前内存中的数据保存至磁盘中。即数据的读入和保存为全自动操作，跟随主窗口的构造与析构自动完成，无需用户手动操作。

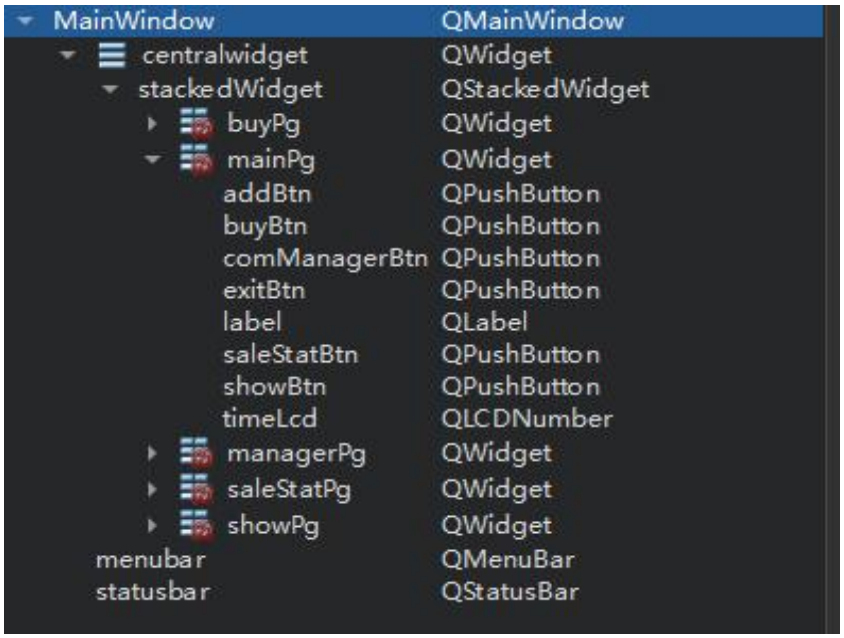


图 3-7 主窗口控件

3.3 商品类

主要包含抽象接口基类Commodity、食品子类Food、化妆品子类Cosmetic、饮料子类Beverage和日用品子类 Daily 的定义与实现。

3.3.1 商品基类文件

主要包含商品编号、商品名、价格成本等所有种类的商品共有属性的成员变量，以及对上述属性的构造函数。其中，商品编号是商品的自增主键,作为商品唯一表示。编号的首位为商品类型的首字母,可以通过商品编号的首位在运行时确商品的类型。由类的封装性需要，所有数据成员均为保护成员，故在类外无法访问。故提供了对于的 get 方法与 set 方法对变量进行获得与设置。特别的对于商品的类型，由于不同的商品子类的区别性，故提供了纯虚函数接口如图 3-8 所示。

```
1  virtual int getKind(void)=0; //返回种类
2  virtual double getWeiOrCapa(void)=0;//获得重量或容量
3  virtual void setWeiOrCapa(double wc)=0;
4  virtual std::string getKindStr(void)=0;
5  virtual void setKind(int k)=0;   //重设商品类型
```

图 3-8 商品基类的纯虚函数接口

3.3.2 商品子类文件

主要包含食品子类 Food、化妆品子类 Cosmetic、饮料子类 Beverage 和日用品子类 Daily 的定义与实现。

各子类均提供了两个重载版本的构造函数。默认构造函数只初始化商品编号，用于从文件读取时的无参构造，带参数构造则用于新增商品时的构造。食品类的构造函数如图 3-9 所示。各子类均重载了 operator<<以及 operator>>用于将数据写入文件中。化妆品类的重载如图 3-10 所示。由于基类中存在纯虚函数，故子类必须完成对虚函数的重写，饮料类的重写如图 3-11 所示。每个子类中均有一个枚举型变量，表示商品的种类。日用品类的枚举类的定义如图 3-12 所示。

```
Food::Food()
{
    _no = "F" + to_string(++maxNo);
}

Food::Food(string name, string brand, string manufacturer,
    double price, double cost, int amount, bool onSale, int saleCount,
    double weight, FoodKind kind)
    :Commodity(to_string(++maxNo), name, brand,
        manufacturer, price, cost, amount,onSale,saleCount),
    _weight(weight), _kind(kind)
{
    _no = "F" + _no;
}
```

图 3-9 食品类的构造函数

```
std::ifstream &operator>>(std::ifstream &ifs, Cosmetic &c)
{
    int k;
    ifs >> c._no >> c._name >> c._brand >>
        c._manufacturer >> c._price >> c._cost >>
        c._amount >> c._weight >> k >> c._onSale>>c._saleCount;
    c._kind = (Cosmetic::CosmeticKind)k;
    return ifs;
}

std::ofstream &operator<<(std::ofstream &ofs, Cosmetic &c)
{
    ofs << c._no << " " << c._name << " " << c._brand << " " <<
        c._manufacturer << " " << c._price << " " << c._cost
        << " " << c._amount << " " << c._weight << " " << (int)c._kind << " " << c._onSale<<" " <<
c._saleCount;
    return ofs;
}
```

图 3-10 化妆品类的左、右移运算符重载

```cpp
int Beverage::getKind(){
    return (int)_kind;
}

double Beverage::getWeiOrCapa(){
    return _capacity;
}

void Beverage::setWeiOrCapa(double wc){
    _capacity=(int)wc;
}

string Beverage::getKindStr(){
    return kindStrs[(int)_kind];
}

void Beverage::setKind(int k){
    _kind=(BeverageKind)k;
}
```

图 3-11 饮料类的重写

```cpp
enum class DailyKind {
        BEDDING,KITCHENWARE,ELECTRIC,USUAL
        //床上用品 厨房用品     电器    常用品
    };
```

图 3-12 日常用品类种类的枚举类

3.4 管理类

3.4.1 管理基类

主要包含管理基类的定义与实现。管理基类的类定义如图 3-13 所示。其中主要包含静态的统计变量总成本、总营业额，来记录超市进货所产生的成本和销售获得的收入。此外，该类还使用商品基类指针的向量存储库存的商品对象。其作为一个静态成员变量，被继承该基类的所有子类对象共享。为了正确的管理堆区资源，使用引用计数的方式并设置如图 3-14 所示的构造函数和虚析构函数使得拥有该资源的所有对象均被析构时才将商品对象析构。此外，该类中还设置了判断商品信息是否重复的接口如图 3-15 所示，用于新增商品时的查重

操作。

```
class Manager
{
protected:
    using Commodities = vector<Commodity*>;
protected:
    static Commodities _repository;  //当前库存
    static double _sumRevenue; //营业额
    static double _sumCost;    //成本
    static double _useCount;  //资源的使用计数
public:
    Manager();
    virtual ~Manager();
    bool comExist(std::string name,std::string brand);  //检验商品是否存在
    double getSumRevenue();
    double getSumCost();
    static void displayRepCom(Commodity* com, QTableWidget* bigTable);  //大表格展示库存中的商品
    static void displaySaledCom(Commodity* saledCom,QTableWidget* samllTable,int amount);  //小表格展示已经购买的商品

};
```

图 3-13　管理基类的定义

```
Manager::Manager(){
    ++_useCount;
}

Manager::~Manager(){
    --_useCount;
    if (_useCount == 0){ //资源引用计数为0时释放
        for (auto& pCommodity : _repository){
            if (pCommodity){
                delete pCommodity;
                pCommodity = nullptr;
            }
        }
    }
}
```

图 3-14 构造函数和虚析构函数的实现

```
bool Manager::comExist(string name, string brand){  //商品查重
    bool b=false,c=false;
    auto check = [](auto begin,auto end,const string& n,const string& b,bool* exist) {
        for(auto it=begin;it!=end;++it) {
            if((*it)->getName()==n && (*it)->getBrand()==b && (*it)->onSale()){
                *exist=true;
                return;
            }
        }
    };
    int half = _repository.size()/2;
     thread th(check,_repository.begin()+half
             ,_repository.end(),name,brand,&c);   //多线程查询
    check(_repository.begin(),_repository.begin()+half,name,brand,&b);
    th.join();
    return b||c;
}
```

图 3-15 基于商品名与品牌判断商品是否重复的接口方法的实现

3.4.2 库存管理子类

该子类主要实现对于当前库存的管理工作，对外提供了对于商品的增删查改以及将商品信息的文件存取操作的接口。

对于商品的查找，主要包含一对一查找（主键查找）与一对多查询（名称、品牌、类型查找）两种方式。在一对多查询时，将会返回一个符合查询条件的基类商品指针的 std::list 链表。其中按照商品名查询的实现如图 3-16 所示，其会返回所有以参数字符串为字串的商品名所对应的基类指针。此外，由于根据商品类的主键商品编号查找在售商品的场景较多，为提高查找速度，使用 std::map 建立商品编号到对应商品的查找的索引。

对于商品的新增与修改，提供了新增四个商品类的接口，接口如图 3-17 所示，供在输入对话框类 InputWidget 中的库存管理类子对象调用实现。

对于商品的删除操作，为不影响交易记录中查看已经购买的商品，并不会将其直接从 vector 中删除，而是调用商品的 setOffSale 方法将商品下架。在记录交易记录时，下架后的商品可通过 searchNoAll 方法通过商品号查找。

对于商品信息的文件存取操作，保存时基于 RTTI 机制和 dynamic_cast 运算符将基类指针转化为子类指针，并调用类中重载的输入流运算符保存在对应的文件中。读取时，将分别读取四个商品类对应的文件，并通过 new 运算符和重载的输入流运算符完成文件中商品的添加。

```cpp
std::list<Commodity*> RepManager::searchName(std::string name)
{
    std::list<Commodity*> resList;//符合条件的商品
    auto check = [&](Commodity* c)
    {

        return c->getName().find(name) != string::npos;
    };
    for (auto com : _repository)
    {
        if (check(com)&&com->onSale())
        {
            resList.push_back(com);
        }
    }
    return resList;
}
```

图 3-16 按照商品名查询的实现

```cpp
bool addFood(std::string name, std::string brand, //新增食物商品
        std::string manufacturer, double price,
        double cost, int amount, double weight,
        Food::FoodKind kind);

    bool addCosmetic(std::string name, std::string brand,  //新增化妆品
        std::string manufacturer, double price,
        double cost, int amount, double weight, Cosmetic::CosmeticKind kind);

    bool addBeverage(std::string name, std::string brand,  //新增饮料
                     std::string manufacturer, double price,
                     double cost, int amount, int capacity,
                     Beverage::BeverageKind kind);


    bool addDaily(std::string name, std::string brand,  //新增日用品
                  std::string manufacturer, double price,
                  double cost, int amount, double weight, Daily::DailyKind kind);
```

图 3-17 新增商品的接口

3.4.3 销售管理子类

该类主要对外提供购买商品和结算的接口以及对继承自基类的商品信息中的销量字段和统计信息进行维护。此外，该类内能够记录交易记录，并对外提供交易记录的查询修改以及退货的接口。

对于交易记录的保存，主要通过图 3-18 示的静态 vector 容器保存。具体形式如_tradeComs[i]记录第 i 次交易的交易号；_tradeAmount[i][j]为指向第 i 次交易的第 j 项商品的基类指针。其中交易号为自增主键，用于标识唯一的交易，可通过图 3-19 所示的方法获得上述的下标 i。其次，类内还提供了将交易记录进行文件存取的接口。

对于交易记录的修改与退货，主要由图 3-20 所示的公有成员函数实现。该函数具有默认参数，若不提供商品指针与新购买数量，则表示进行退货操作，将该交易的所有商品全部退去，若不提供购买数量，则意味着将该商品全部退去。若是增加购买数量，当增量大于库存数量时也会抛出异常。

对于商品的购买，由图 3-21 所示的 buy 方法实现，需要提供要购买的商品的基类指针以及购买数量，若购买的数量大于该商品的库存数，则会抛出异常。反之则将其信息存入本次购买的缓存容器中。结算时可调用图 3-22 所示的 endBuy 方法将本次购买的信息存入上述静态的 vector 容器中，并通过 std::tuple 返回自动生成的交易编号和总金额。

```
protected:
    static vector<Commodities>_tradeComs;  //交易商品记录
    static vector<vector<int>>_tradeAmount;  //商品数量记录
    static vector<vector<double>>_tradePrices;  //商品成交价格记录
    static vector<double> _tradeIncome; //交易额记录
    static vector<std::string> _tradeId; //交易号
```

图 3-18 保存交易记录的静态 vector 容器

```
int Saler::getPosition(std::string trid){
    if(trid.size()<9) throw runtime_error("非合法订单号");
    trid.erase(trid.begin(), trid.begin() + 3);
    long long  id = stoull(trid); //转为数字
    if (id < 0 || id - 100000 - 1 >= _tradeId.size()){
        throw runtime_error("没有该订单");
    }
    return id - 100000 - 1;  //对应数组下标
}
```

图 3-19　通过交易号字符串解析对应下标

```cpp
void Saler::reSetTrRecord(const string& trid, Commodity* com, int newAmount){
    int pos = getPosition(trid);
    bool noFind = true;
    for (int i = 0; i < _tradeAmount[pos].size(); ++i) { //在记录中寻找
        if (com == nullptr || _tradeComs[pos][i] == com){//找到该商品 或需要全部删除
            noFind = false;
            int delta = newAmount - _tradeAmount[pos][i];  //新旧数量差距
            if (delta > 0 && com->getAmount() < delta){
                throw("当前库存不足，无法修改");
            }
            Manager::_sumRevenue += delta * _tradePrices[pos][i]; //更新总收入
            _tradeIncome[pos] += delta * _tradePrices[pos][i];  //更新交易额
            if (newAmount != 0) { //新数量不为零时更新数量
                _tradeAmount[pos][i] = newAmount;
            }
            _tradeComs[pos][i]->setAmount(
                _tradeComs[pos][i]->getAmount() - delta);  //更新库存数量
            _tradeComs[pos][i]->setSaleCount(
                _tradeComs[pos][i]->getSaleCount() + delta);//更新商品的销量记录
            if (newAmount == 0){  //等于0时清除数据
                _tradeComs[pos].erase(_tradeComs[pos].begin() + i);
                _tradePrices[pos].erase(_tradePrices[pos].begin() + i);
                _tradeAmount[pos].erase(_tradeAmount[pos].begin() + i);
                --i;//有数据删除时指针不移动
            }
        }
    }
    if (com != nullptr && noFind)  throw runtime_error("未购买该商品");
}
```

图 3-20 交易记录的修改与退货方法的实现

```cpp
void Saler::buy(Commodity* com, int amount) {
    if (com==nullptr || amount > com->getAmount()) {
        throw runtime_error("库存不足"); //商品库存不足已买这么多  或不存在该商品
    }
    else {
        bool noSame=true;
        for(int i = 0;auto boughtCom:_coms) {  //若有重复则合并
            if(boughtCom==com) {
                _amount[i]+=amount;
                noSame=false;
            }
            i++;
        }
        if(noSame) {  //无重复则新增
            _coms.push_back(com);
            _amount.push_back(amount);
            _price.push_back(com->getPrice());  //购买时的价格
        }
        com->setAmount(com->getAmount() - amount); //更新库存中的数量
        com->setSaleCount(com->getSaleCount() + amount);
        _sumRevenue += (com->getPrice()) * amount; //更新总营业额
        _income += (com->getPrice()) * amount;
    }
}
```

图 3-21 购买商品方法的实现

```
tuple<string,double> Saler::endBuy(){
    _tradeAmount.push_back(move(_amount));
    _amount.clear();
    _tradePrices.push_back(move(_price));
    _price.clear();
    _tradeIncome.push_back(_income);
    _tradeComs.push_back(move(_coms));
    _coms.clear();
    _tradeId.push_back("Tr-" + to_string(++_maxId));
    auto res=_income;
    _income=0;
    return {*(_tradeId.end()-1),res}; //返回商品的总金额
}
```

图 3-22　结算方法的实现

4 系统测试

设计测试用例，给出程序每个功能模块的运行结果截图。

4.1 用户登录测试

在密码文件“passWord.pwd”文件内容为 123456 时，分别输入 123456 与 123556，运行结果分别如图 4-1 与 4-2 所示。在 密码设置为 123456 时，输入正确密码后，修改密码为“bigwhites123”，运行结果和文件内容如图 4-3 所示。

图 4-1 输入密码正确登入系统

图 4-2 输入密码错误提示

图 4-3 修改密码操作成功

4.2 新增商品测试

在主界面点击新增商品会弹出选择框，提示用户选择新增商品的种类，如下图 4-4 所示。

图 4-4 用户新增商品提示框

选择对应的商品类型后会弹出一个添加商品信息的窗口，如下图 4-5 所示。添加的时候会检查是否存在此商品，如果存在会提示用户已存在，如果不存在则会提示用户添加成功，添加成功如图 4-6 所示。

图 4-5 增加商品信息输入框

图 4-6 增加商品提示成功

4.3 显示所有商品测试

在程序的主界点击显示所有商品，会进入商品库存界面，如下图 4-7 所示。

图 4-7 商品库存界面

在此界面的正下方有排序根据和排序规则两种组合，选择完成后点击显示在此界面显示所有商品根据排序依据和排序规则排序后的所有信息。如下图 4-8 是根据价格升序排序的商品信息。

图 4-8 库存排序后的显示结果

4.4 收银台测试

在主界面点击收银台后，会进入到收银台界面，收银台界面如 4-9 所示。

图 4-9 收银台界面

如图所示，需要输入商品号在库存中查找是否有此商品,若有则在商品信息栏里显示，用户可以更改数量，如果搜索的商品号不存在，则会提示用户商品不存在。下图 4-10 为添加可购商品，4-11 为添加不存在商品的提示，在此页面的右下角，添加完商品会更新当前应收的总价。

图 4-10 添加存在商品

4-11 添加不存在商品的提示

在添加完商品过后点击结算，会出现一个提示用户购买成功的窗口，这个窗口会输出此次购买的订单号以及总金额，测试如图 4-12 所示。

图 4-12 结算提示窗口

4.5 销售统计测试

在主界面点击销售统计，会进入销售统计界面，销售统计界面如图 4-13 所示。

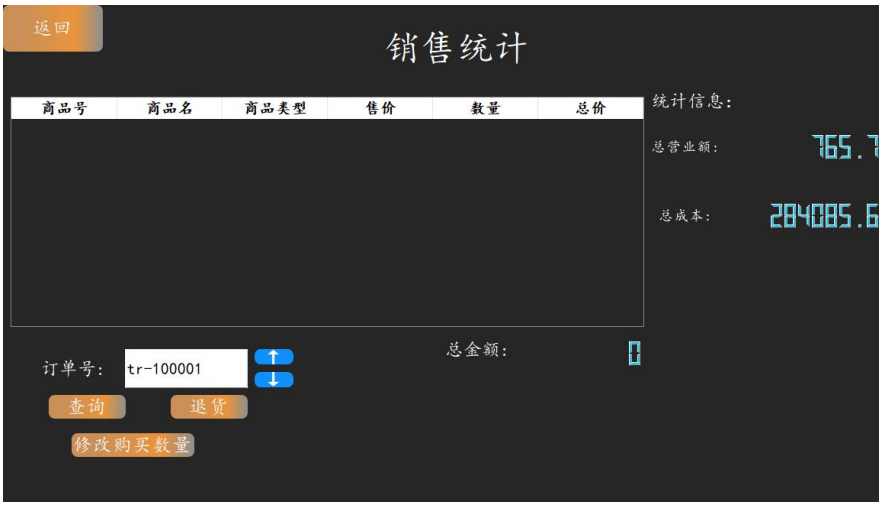


图 4-13 销售统计界面

在此界面可以查询订单号，并输出该订单的购买商品信息。在订单号右边有上下两个蓝色的按键，按下后会在信息输出框分别输出前一条、后一条销售信息。在此页面右边会展示统计信息，一个是总营业额，一个是所有的商品的总成本。

如图 4-14 所示为在此页面显示某一条订单的信息。

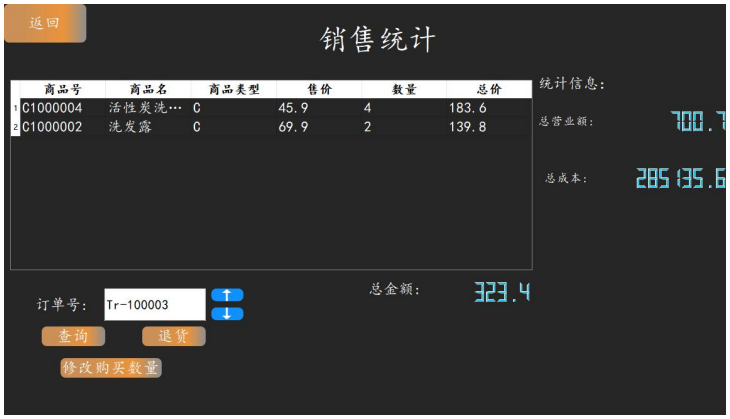


图 4-14 销售统计界面显示某条订单信息

此界面还可以删除和修改订单购买的数量，如图 4-15 为点击修改订单数量后弹出的提示窗口。图 4-16 为选择需要修改购买商品数量后的提示窗口，需要用户修改购买的数量，如果需要个别商品退货，则可以通过这将购买数量修改为 0。如果需要全部退货，则搜索对应的订单后点击退货即可。

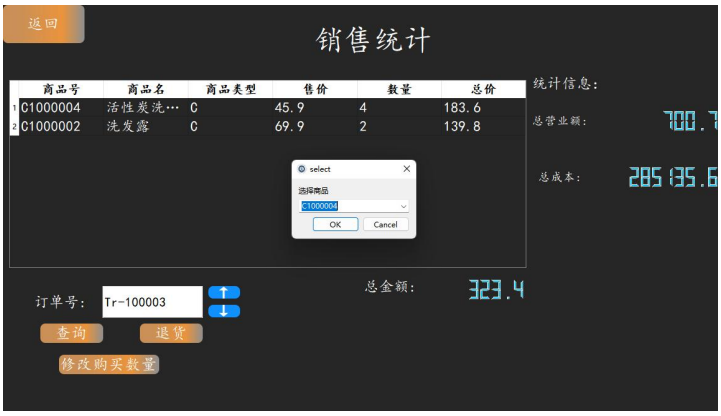


图 4-15 修改订单数量提示窗口

图 4-16 修改数量提示窗口

4.6 商品管理界面测试

在主界面点击商品管理，则会进入商品管理界面，商品管理界面如图 4-17 所示。

图 4-17 商品管理界面

如图所示，商品管理界面初次打开，不会显示商品的信息，必须点击右下角查询的确定按键才会显示所有商品信息，显示所有商品信息如图 4-18 所示。

4-18 商品管理界面显示所有商品信息

在此界面，可以通过选择右下角的查询方式来查找商品，所有的查询方式如下图 4-19 所示。

图 4-19 所有的查询方式

以下是演示使用商品名查询商品，如图 4-20 和 4-21 所示。

图 4-20 提示用户输入商品名

4-21 搜索结果

在此界面不仅可以查询商品的信息，还可以修改或者删除商品的信息。我们可以双击某个商品号，此时会弹出修改信息的窗口，图 4-22 为双击商品号后弹出的修改商品信息的窗口。我们也可以选择某一商品，然后单击商品管理界面的删除，删除商品信息。

图 4-22 商品修改信息窗口

4.7 退出系统测试

在主界面选择退出系统后，会先在对应的存储文件保存各商品及订单记录。图 4-23 和 4-24 分别为退出系统后，食品类库存和订单记录的存储文件内容。

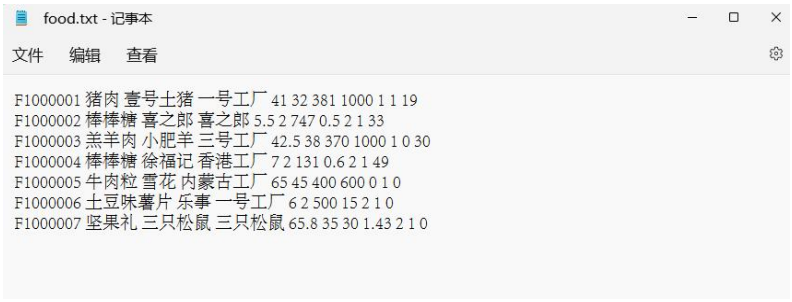
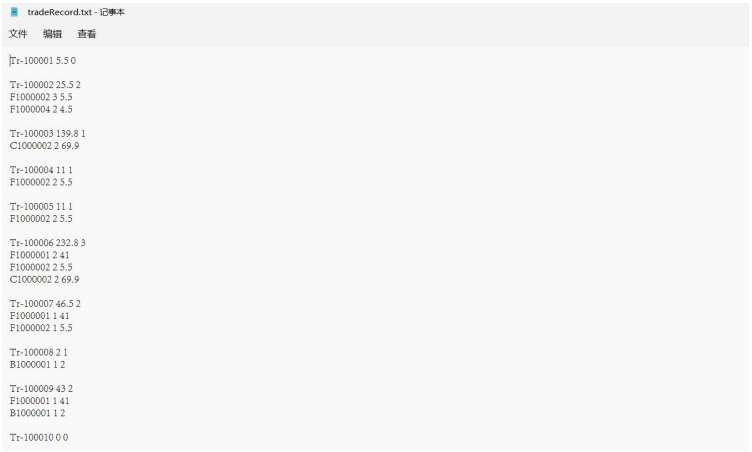


图 4-23 退出系统后食品库存文件内容



4-24 退出系统后订单记录文件内容

5 系统总结

5.1 具体分工

本次面向对象课程设计由本人与赵鹿均同完成。本人在本次课设中的具体任务如下：

5.1.1 类的层次结构设计

完成了对系统种各种类层次结构设计，确定了系统所需要的类以及其组合关系，明确了各类需要的接口。具体为：设计商品基类存放商品的共有属性，同时设计不同类型商品的四个子类，每个子类中再存放其特性的信息；设计管理基类保存所有的商品对象，再设计两个子类分别负责库存管理以及商品销售。为保证商品数据的全程序共享与实现运行时多态，将商品的采用基类指针的方式进行维护，并采用引用计数的方式进行资源的释放操作。通过交流与讨论的方式将设计与组员沟通。组员基于此项设计完成各类的代码实现工作。

5.1.2 图形界面的设计与美化

将小组成员设计的商品、管理库移植到 Qt 框架中，基于 Qt 框架以及商品、管理类暴露的接口与 Qt 窗口类完成数据交互与输入输出。同时完成主窗口类、登录对话框类以及商品信息输入窗口类的设计与实现。同时使用 Qt creator 的窗体设计器完成对以上三种的界面的骨架设计。窗口骨架如图 5-1 所示。

在完成系统的界面设计并与组员编写的类库整合完毕后，使用 QSS 对界面进行美化与调整，调整后的界面如图 5-2 所示。具体工作为，设置背景、调整按钮颜色、主界面加入时间显示控件以及设置字体大小及字体等。

图 5-1 初始时的界面

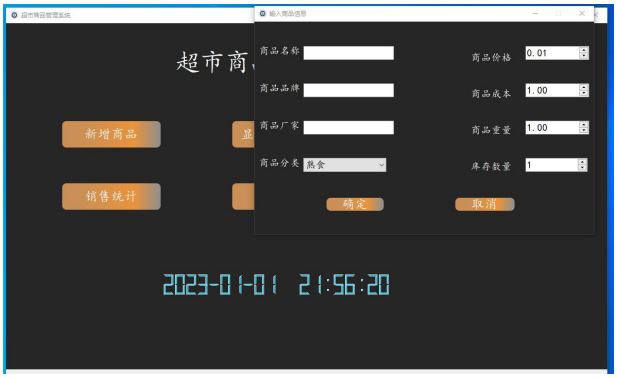


图 5-2 QSS 美化后的界面

5.2 项目综述

本项目实现了对超市管理系统的设计，超市可管理的商品共有四类食品、化妆品、日用品和饮料，且每种商品都包含商品编号、商品名称、价格、库存量、生产厂家和品牌等信息。具体源代码可在该仓库中查看：https://github.com/bigwhites/cpp-qt-manageSystem。

本项目通过图形界面实现了以下功能：

1) 销售功能。在购买商品时，先输入商品号，然后根据商品号查询商品相关信息，如果有库存量输入购买的数量，进行相应的计算。如果库存量不足，则弹出提示窗口提示用户库存不足。

2) 商品管理功能。可以通过图形界面提示用户输入相关的商品信息；可以按商品编号、商品名称、商品品牌、商品类型进行查询，并在输出窗口输出对应的商品信息。

3) 商品售出功能。记录每一次商品售出记录，并保存售出商品信息。并且允许用户退货。

4) 删除功能。主要完成商品信息的删除。在商品管理界面，查询到商品信息后可点击商品退货进行退货处理。

5) 修改功能。可以对查询信息进行相应的修改。

6) 统计功能。可按商品的价格、库存量、生产厂家进行统计，可按从小到大和从大到小两种方式进行排序。

7) 商品信息及订单信息存盘。可将程序中的商品信息和订单信息存入到文件中。

8) 读出信息。每次启动程序时自动读入存储信息的文件，将磁盘的内容读取到内存中。

5.2.1 设计亮点与优势

① 类的设计较为合理

使用了类的继承特性，通过抽象基类和虚函数等设计，提高代码的复用性和可扩展性。

通过使用静态成员变量实现数据所在对象间的共享。

类的封装性比较好，只对外提供必要的接口，数据成员设为保护权限。

② 基于 Qt 框架开发

基于 Qt 框架开发，有图形界面，交互边界直观，使用 QSS 美化，界面简洁美观。

使用了 QString、QTime 等 qt 框架中的类，基于 Qt 的对象树机制进行内存管理。

③ 使用了现代 C++特性

运用了结构化绑定、RTTI 机制、lambda 表达式、多返回值等现代 C++特性。

④ 鲁棒性较强

对不合理的输入及操作具有检测和过滤功能。如添加商品时会对商品名、品牌名进行验证，若库存当中有相同商品则会提醒用户；添加商品的时候，如果商品名、品牌名、生产厂家为空会提醒用户输入信息不全。

⑤ 部分代码具有较高的可复用性

在本项目当中，部分代码具有较高的可复用性。商品信息的录入和商品信息的修改是用的同一个窗口，只改变了判别条件，使得代码复用。

5.2.2 设计难点与不足

① 部分功能代码较为冗长

在本项目中虽然有些代码具有较高的复用性，但是也有部分代码比较冗长，功能相似的

函数较多，比如 RepManager 类中的添加商品的接口，用了四个接口来实现对四个商品类的添加；四个商品子类中的左、右移运算符分别重载了一次也使得代码显得冗长。

② 没有将交易记录单独作为一个类

导致销售类的代码逻辑较为复杂，可读性和拓展性比较差，退货的复杂度比较高。

5.3 个人收获与展望

本次面向对象课程设计圆满完成，大体上实现了任务的要求，系统能够正常运行且在简单的测试中并无发现逻辑错误和崩溃现象。

本次的课程设计是我第一次设计有一定复杂程度的应用程序，同时初步掌握了 Qt 框架的使用，借助 Qt 框架中的类库以及 Qt creator 的设计工具，够编写出比以往的控制台程序更加易用、美观的程序。经过本次课程设计，我对桌面客户端的开发流程有了一定成都上的了解。

同时，在设计阶段，由于需要与组员协同开发，为保证代码的同步和版本控制，使用了 git 作为版本控制系统，并将代码同步到 github 上述的代码仓库中。这种方式大大提高了协同开发的速度以及效率，在编写代码出现难以修复的问题时也能便捷的回溯到以前的版本。

另外，通过本次课设，我更加理解了面向对象的编程思想以及类的使用及设计。通过学习 Qt 类库的设计，例如窗口类库中的基类 QWidget 以及各种子类 QMainWindows、QDialog 等的组合关系，以及所有 Qt 类中的总基类 QObject 的设计与作用，让我对面向对象这一变成范式以及意义有了更加深刻的认知。

最后，本次的课设仍有可以改进之处，若在今后的学习中掌握了更多的知识，就可以将该系统修改为基于数据库保存商品和用户信息，这样可以保证在数据量较大时系统仍能以较快的速度运行。这样的模式也更加接近实际使用中的超市商品管理系统。