OOP大作业实验报告

马珩峻 171860627

1. 引言

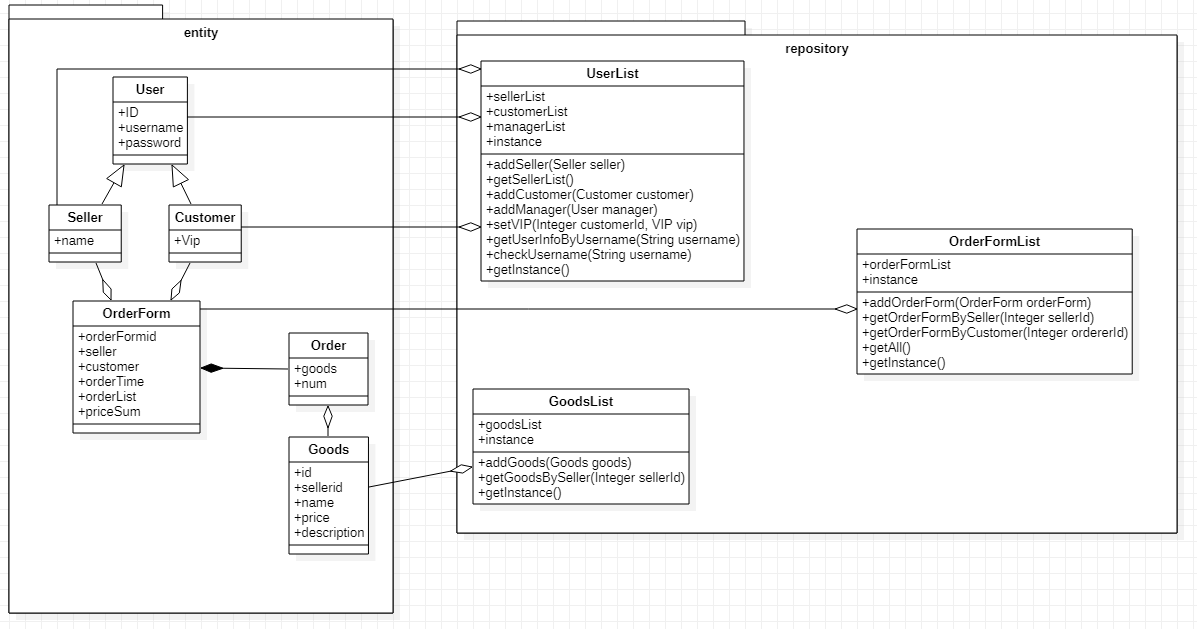
本项目使用java语言进行编写，客户端与服务端之间使用socket进行通信，客户端使用javafx实现了GUI设计。

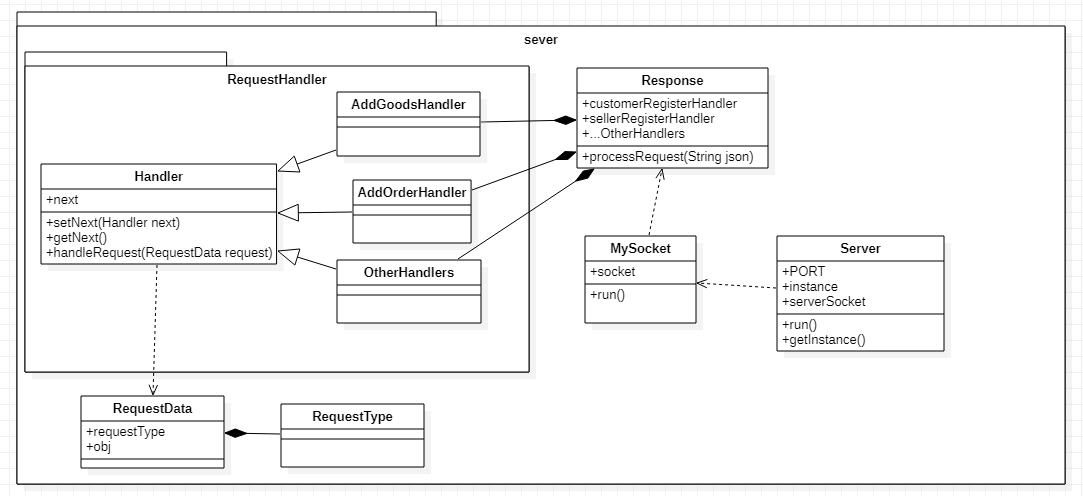
1. 分析设计

简易外卖软件首先需要分别设计客户端和服务端，然后总共有3种类型的客户：商家，客户，管理员，然后涉及到要操作的对象有商品，订单，购物车等，另外客户端和服务器通信时需要一个Socket对象进行通信，需要对请求数据包进行处理，需要将用户发送的数据存入服务器中。

对于服务端，服务器Server只会有一个实例，所以采用单例模式，同样的，各种与服务器数据交互的类也只会有一个实例，也采用单例模式，另外java原生的socket是没有多线程的，需要转换为能够多线程运行的MySocket类，这里使用了适配器模式，处理客户端请求时，创建不同请求的处理类，使用责任链模式对客户端的发来的请求进行处理。

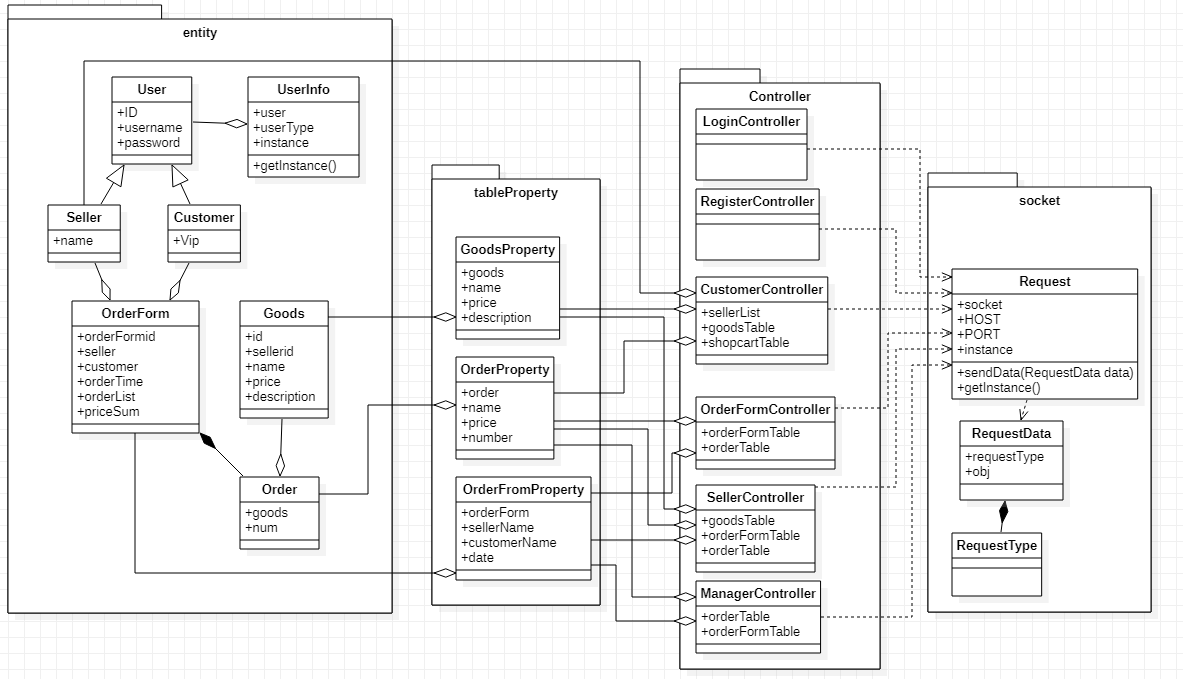
分析后可以得到如下UML建模（所有类方法省略了对属性的set和get方法）：





对于客户端，首先一个客户端只会有一个用户使用，所以使用UserInfo类管理单前用户，并且使用单例模式。对于javaFx 主要使用Controller来对GUI上的元素进行操作。对于javaFx的表格，所需要的数据需要特别定义的属性才能显示，将原本entity中的实体对象显示到table中时需要使用适配器模式将类转换成能在table中显示的对象。然后对于javafx的表格和列表的点击响应事件，需要添加监听器对象来对事件进行监听，这里使用了观察者模式。另外负责发送请求的Request类也只应该有一个实例，所以使用单例模式。

分析后可以得到如下UML建模：（Controller中的GUI控件属性太多且不影响不具体写明，只说明有聚合关系的，控件方法同理）



1. 实现方案

如分析中所说，按上一步的分析给出了具体的实现，对于服务器上的数据存储使用了HashMap的数据结构存储相关数据，数据库与服务端通信时主要将数据先转为Json字符串通过socket字符流传输，收到后再将json字符串反序列化获得相应的对象。

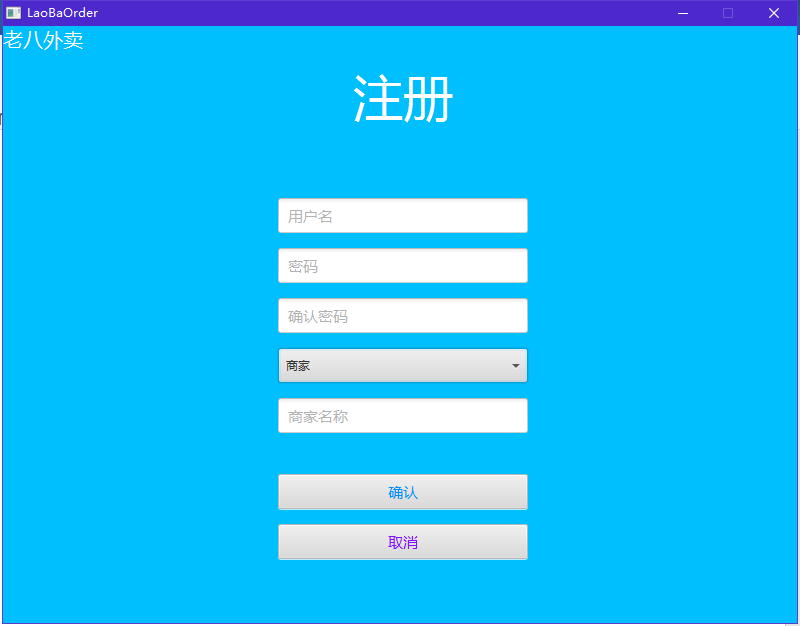
设计模式又上一步分析中有用到：单例模式（并且大量使用），责任链模式，适配器模式，观察者模式。

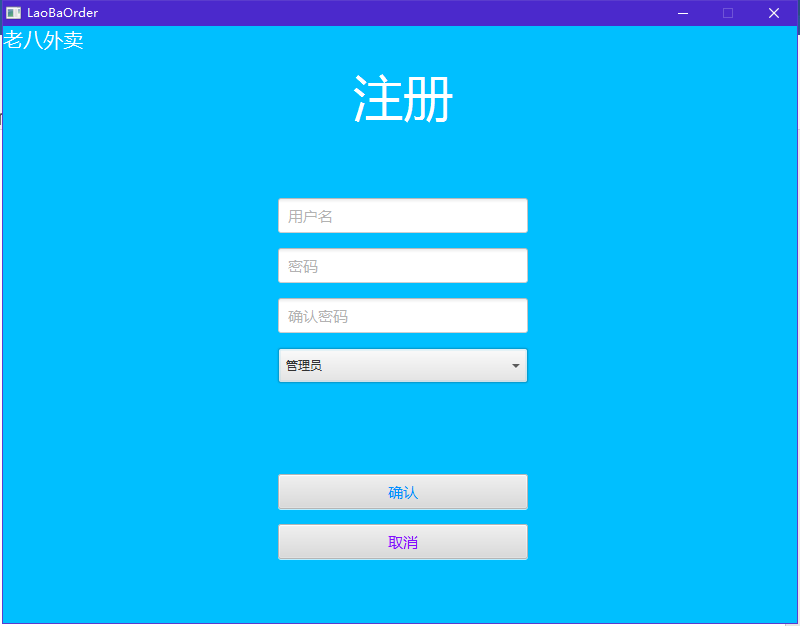
1. 功能实现

实现了所有基础功能和扩展功能

展示：用户注册登录

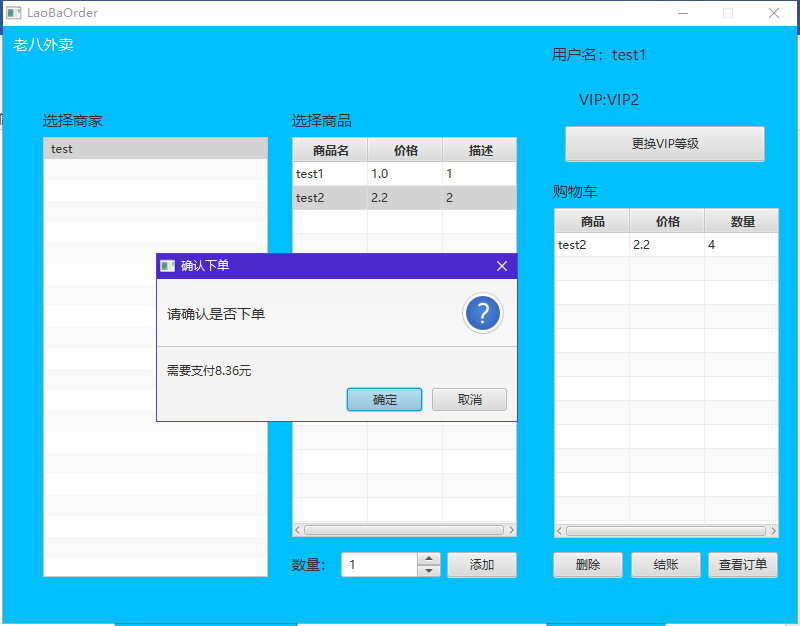




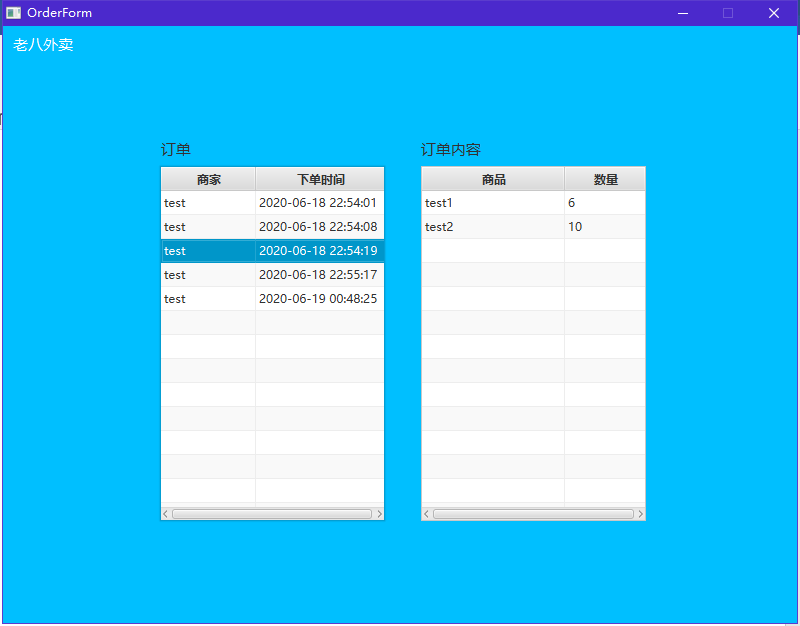




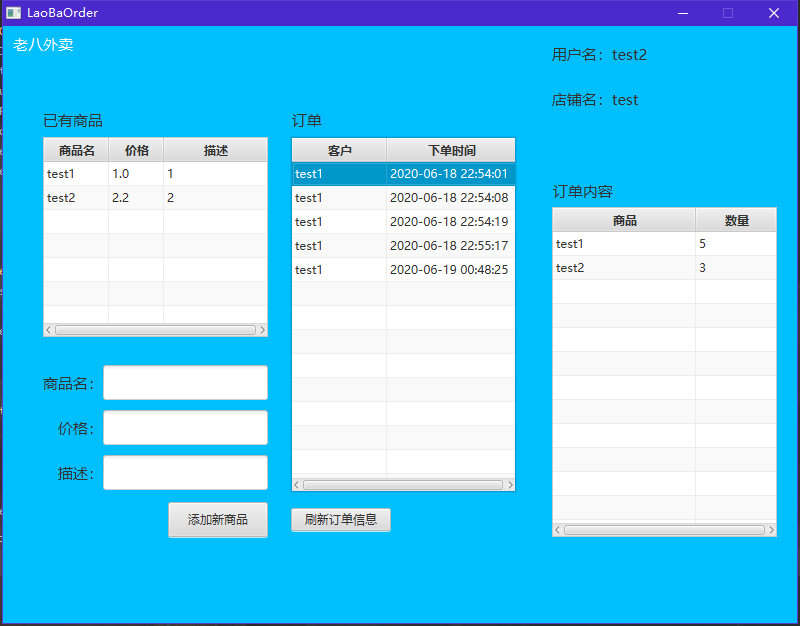
客户界面：



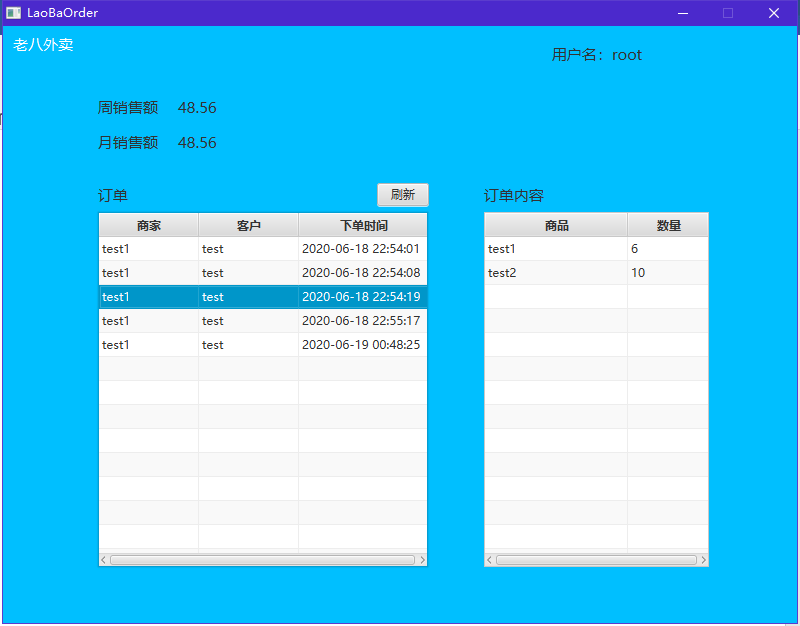
客户查看订单：



商家页面：



管理员界面：



设计原则：遵守了里氏替换原则，单一职责原则，合成复用原则

1. 小结

经过本次大作业，对设计模式的使用有了一定的理解，能够将一些设计模式用于项目实现中，另外对java的socket通信和javafx的gui编写也学到了不少。