计算机科学系实验报告

课程名称	UML 与可视化建模	班级	2 班
实验名称	淘淘商城	教导教师	曾少宁
姓名	陈保锋	学号	1414080901221
日期	2016.5.18		

一、实验目的

掌握基于 UML 2.0 的建模概念与方法，掌握各种 UML 图的概念与画法，其中包括用例图、活动图、类图、顺序图、组件图和状态图等。

二、实验设备与环境

操作系统：Windows 10；建模工具：StarUML。

四、实验要求

1. 实验及实验报告以增量方式完成，每次作业都在上一次作业的基础上完成，作业提交网站不提供报告下载，所以请同学们自行保管好自己的实验报告；

2. 请将实验报告中“占位符”信息替换为自己的实验相关信息；

3. 请认真撰写实验体会，实验课结束时立即上传实验报告：http://zeng.shaoning.net/uml/。

四、实验内容、程序清单及运行结果

淘淘商城

角色:用户

1、单点登录

2、下单

3、加入购物车

4、 实验一：需求建模 - 用例模型

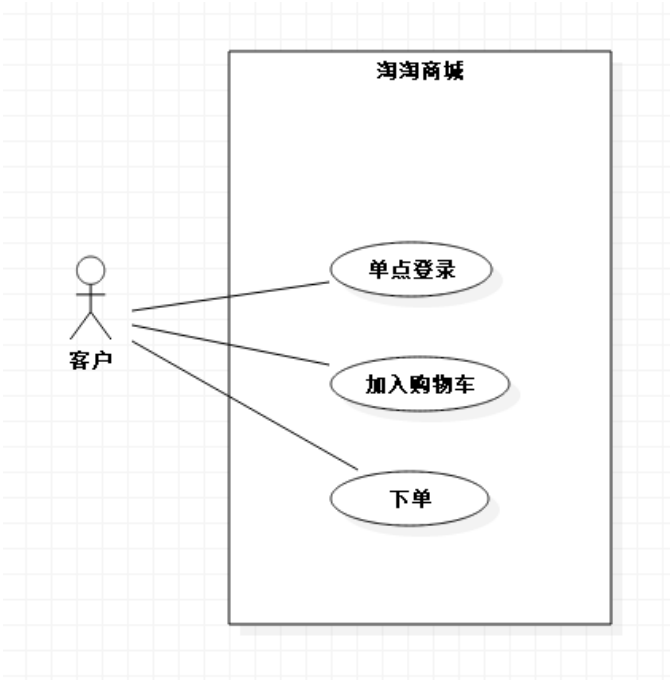


图 1：淘淘商城用例图

注：用例规约内容及项目可自行增加。

用例编号：	UC001
用例名称：	单点登录
用例描述：	分布式环境中,根据页面并发量的不同同一个系统将拆分为不同的模块部署到不同的服务器中,单点登录实现用户的 session 共享.
前置条件：	Redis 缓存
基本流程：	1.用户访问登录页面 2.后端拦截器拦截用户登录 URL，从 Cookie 中获取 TOKEN_KEY 信息 3.后端利用获取的 TOKEN_KEY 查询 Redis 服务器，获取到用户信息 4.向 RabbitMQ 的交换机发送用户登录的消息 5.登录成功，重定向到主页
扩展流程：	2.1 后端没有获取到 TOKEN_KEY 信息，转发到登录页面 2.2 用户填写账号密码，提交 POST 表单 2.3 后端查询用户表，匹配账号密码 2.4 后端生成 TOKEN_KEY,保存该用户信息到 Redis 服务器,向客户带有 TOKEN_KEY 信息的一级域名的 Cookie

	2.5 向 RabbitMQ 的交换机发送用户登录的消息 2.6 用户登录成功,页面重定向到主页 3.1 没有命中 TOKEN_KEY，用户信息可能过期，后端转发到登录页面 2.3.1 账号密码匹配失败，转发到登录页面，做出“账号或密码错误”提示
后置条件：	其他二级域名对应的系统在访问TOKEN_KEY信息的时候需要提供相应的服务程序给其他系统查询.

用例编号：	UC004
用例名称：	加入购物车
用例描述：	用户在离线或者登陆状态都能将商品加入购物车，需要维护登陆与 cookie 缓存的购物车信息。
前置条件：	购物车系统消息接收队列需要注册接收用户登录成功的 RabbitMQ 交换机
基本流程：	1、用户在商品详情页点击加入购物车,校验表单数据 2、购物车系统接收到商品 id，对所有 id 信息进行校验，重新从商品表查询所有商品信息 3、购物车系统查询单点登录系统的用户登录信息服务，返回用户登录信息 4、购物车系统根据商品信息和用户 id 将商品信息添加到购物车表 5、购物车系统添加商品成功，向客户端反馈成功添加信息
扩展流程：	1.1 用户没有选择商品信息，作出提示

	2.1 没有查询到商品信息，用户请求参数不可靠，向用户返回错误信息 3.1 购物车系统没有从单点登录系统中查询到用户登录信息，用户处于离线状态 3.2 查询 cookie 中的 cart_token 信息 3.3 根据 cart_token 信息查询 Redis 服务器，获取购物车信息 3.4 将返回的购物车信息进行反序列化，将用户添加的商品加入到其中 3.5 重新将所有购物车信息序列化并写入到 Redis 服务器 3.6 购物车系统添加商品成功，返回添加成功信息给客户端 3.2.1 没有查询到 cart_token 信息，生成一个 cart_token 唯一码 3.2.2 将所有商品信息进行序列化，将 cart_token 和商品信息写入到 Redis 服务器 3.2.3 生成 cart_token 的 cookie 信息，设置到响应头中 3.2.4 购物车系统添加商品到购物车成功，返回添加成功信息 3.3.1 用户有 cart_token 的 cookie 信息但是没有从 Redis 服务器中查询到商品信息，该 cookie 信息非法，将 cart_token 为 null 的 cookie 信息写入到响应头，即删除该 cookie 信息，重走 3.2.1 流程
后置条件：	1.购物车系统从消息队列中接收到用户登录成功消息 2.从消息中获取所有 cookie 信息，获取 cart_token 信息 3.根据 cart_token 信息查询 Redis 服务器 4.反序列化所有商品信息 5.将商品信息同步到数据库 2.1 没有获取到 cart_token 信息，直接结束，不做任何操作 3.1 没有查询到商品信息，直接结束，不做任何操作

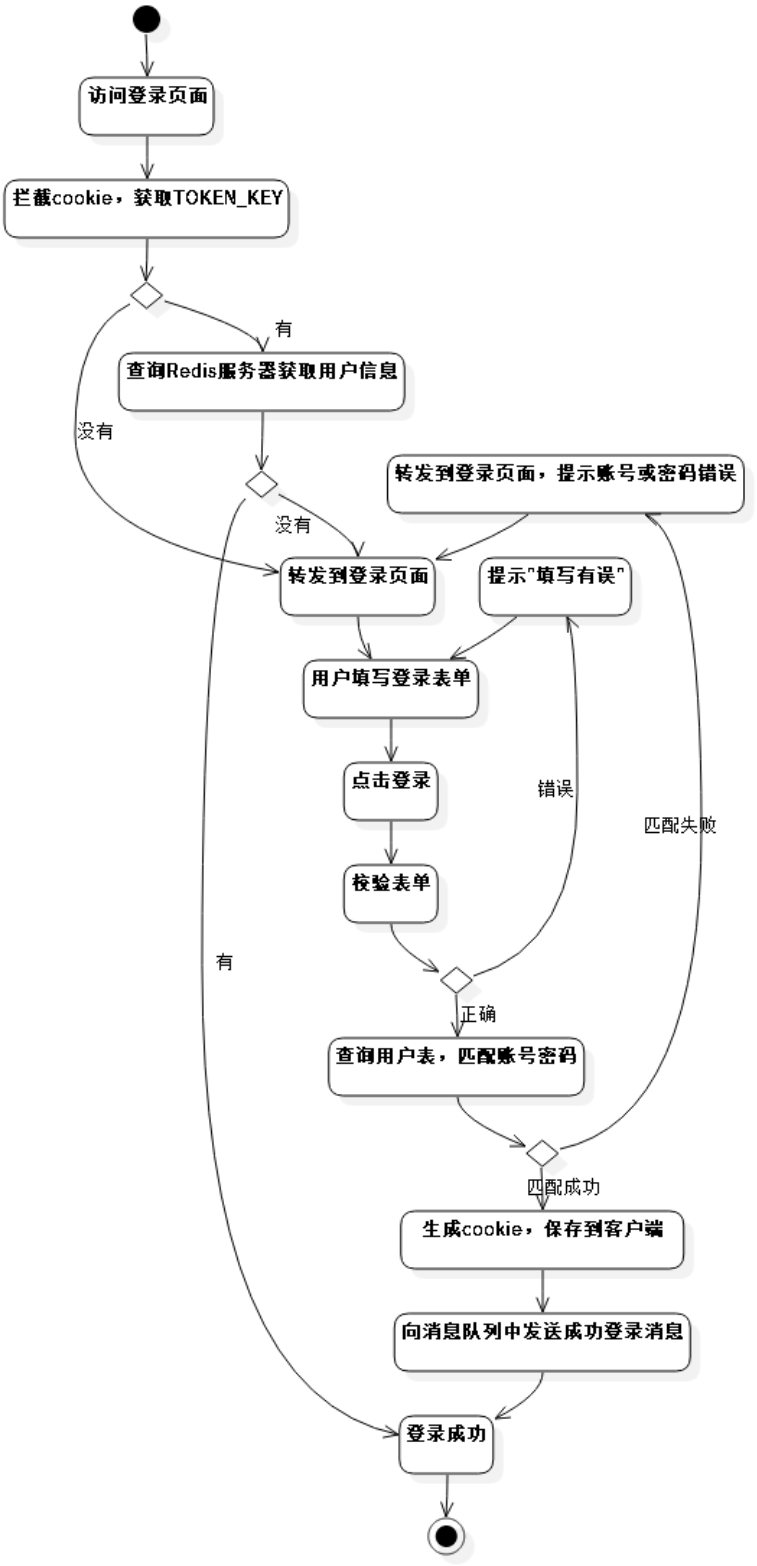
用例编号：	UC003
用例名称：	下单
用例描述：	下单

前置条件：	下单并利用第三方支付
基本流程：	1.用户勾选需要下单产品点击下单，校验表单数据 2.后端根据商品信息查询商品数据库，整合商品信息，计算商品总价，转向确认订单页面 3.用户点击提交订单，校验用户是否选择收货地址 4.后端接收订单信息，重新计算商品总价，保存订单信息到订单表，重定向到第三方支付 5.客户支付完毕，第三方支付回调支付成功服务 6.后端接收到支付成功信息，修改订单状态，页面重定向到支付完成页面
扩展流程：	1.1 用户没有选择商品，做出提示 2.1 没有查询到商品信息，向客户端反馈“数据异常” 3.1 用户没有选择收获地址，做出提示 4.1 用户 30 分钟内没有支付,订单修改为过期状态
后置条件：	

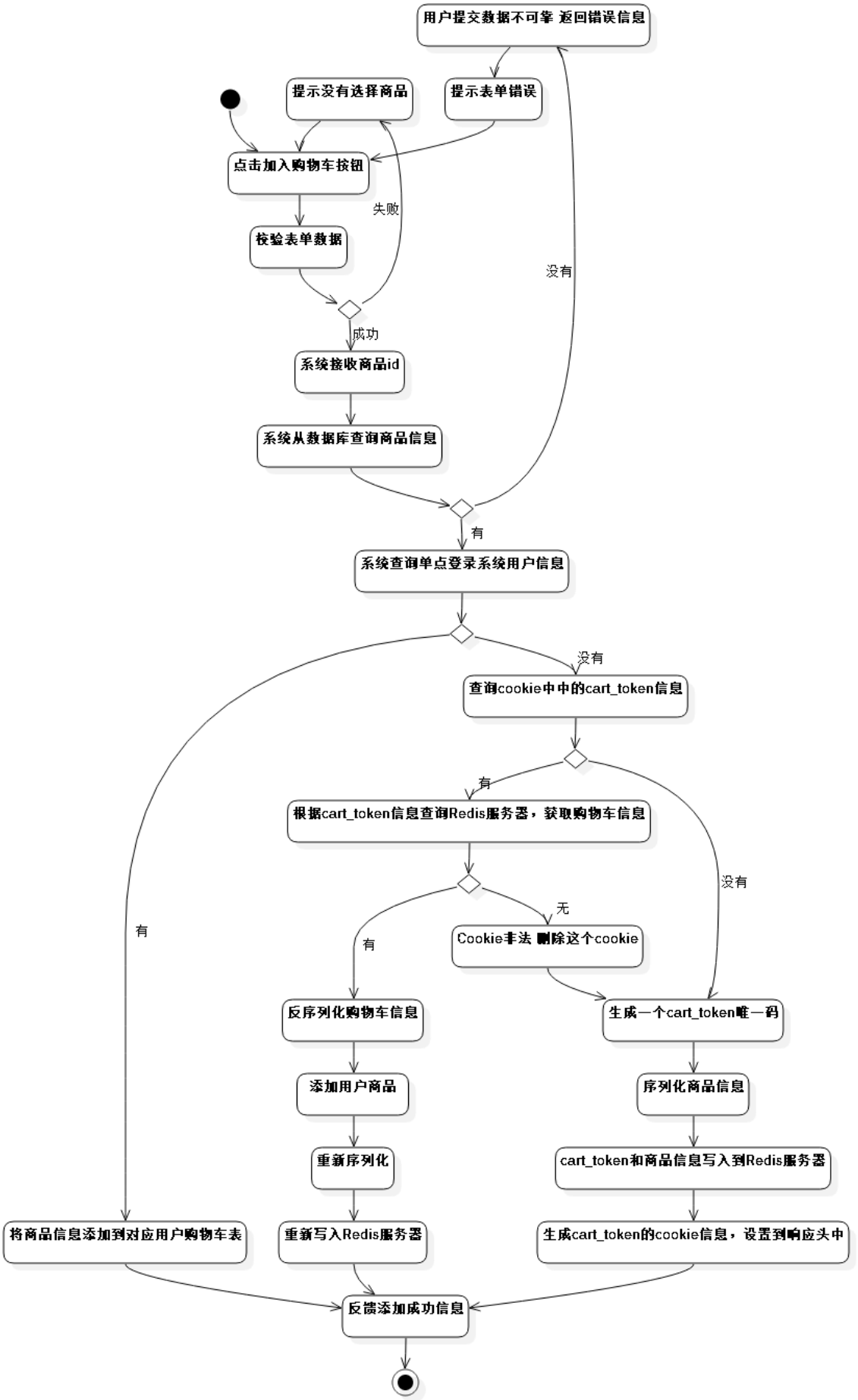
5、 实验二：过程建模 – 活动模型

使用活动图描述系统的业务过程。

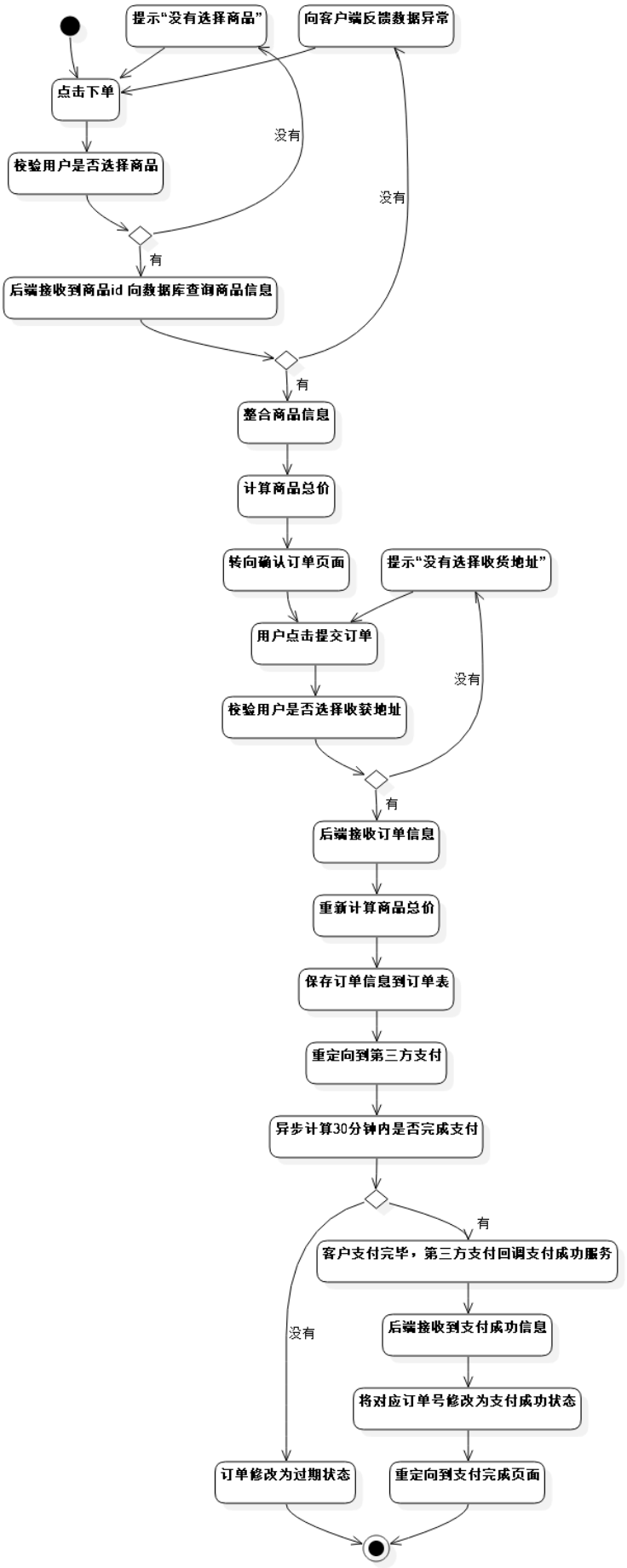
方法：将用例规约中的基本流程与扩展流程抽象为过程步骤（Action），画出对应的活动图。


访问登录页面
拦截cookie，获取TOKEN_KEY
有
查询Redis服务器获取用户信息
没有
转发到登录页面，提示账号或密码错误
没有
转发到登录页面
提示"填写有误"
用户填写登录表单
点击登录
错误
匹配失败
校验表单
有
正确
查询用户表，匹配账号密码
匹配成功
生成cookie，保存到客户端
向消息队列中发送成功登录消息
登录成功

单点登录活动图


用户提交数据不可靠 返回错误信息
提示没有选择商品
提示表单错误
点击加入购物车按钮
失败
校验表单数据
没有
成功
系统接收商品id
系统从数据库查询商品信息
有
系统查询单点登录系统用户信息
没有
查询cookie中中的cart_token信息
有
根据cart_token信息查询Redis服务器，获取购物车信息
无
Cookie非法 删除这个cookie
没有
有
有
反序列化购物车信息
添加用户商品
重新序列化
重新写入Redis服务器
生成一个cart_token唯一码
序列化商品信息
cart_token和商品信息写入到Redis服务器
生成cart_token的cookie信息，设置到响应头中
将商品信息添加到对应用户购物车表
反馈添加成功信息

加入购物车活动图


提示“没有选择商品”
向客户端反馈数据异常
点击下单
没有
校验用户是否选择商品
没有
有
后端接收到商品id 向数据库查询商品信息
有
整合商品信息
计算商品总价
转向确认订单页面
提示“没有选择收货地址”
用户点击提交订单
没有
校验用户是否选择收获地址
有
后端接收订单信息
重新计算商品总价
保存订单信息到订单表
重定向到第三方支付
异步计算30分钟内是否完成支付
有
客户支付完毕，第三方支付回调支付成功服务
没有
后端接收到支付成功信息
将对应订单号修改为支付成功状态
订单修改为过期状态
重定向到支付完成页面

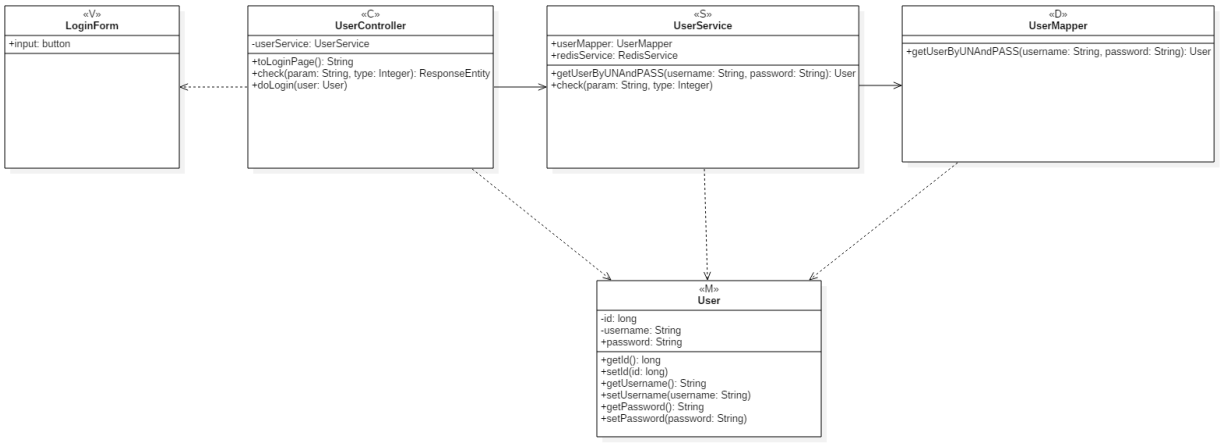
下订单活动图

6、 实验三：逻辑建模 – 类模型

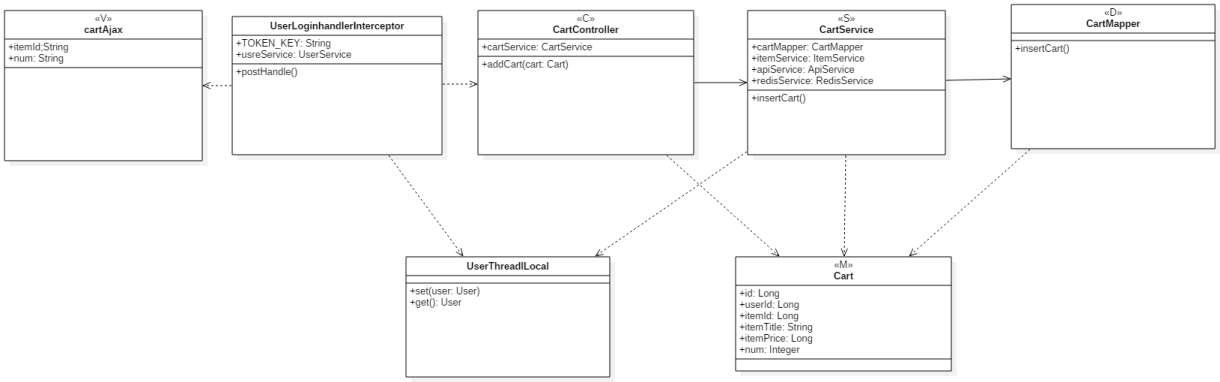
基于 MVC 设计模式找出实现用例的类。

方法：分别找出实现用例的模型（Model）、视图（View）和控制器（Controller）类，确定类之间的关系及其关键属性，画出类图。

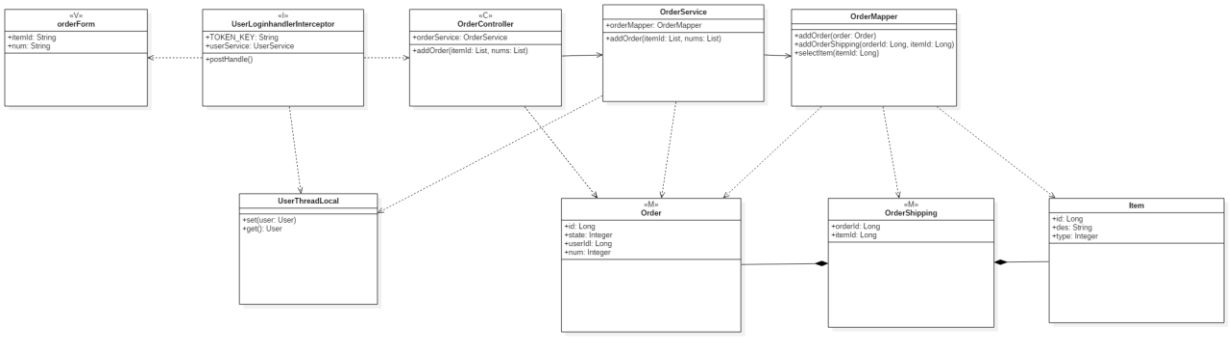
参考：讲义 P26 页。



单点登录类图



加入购物车类图



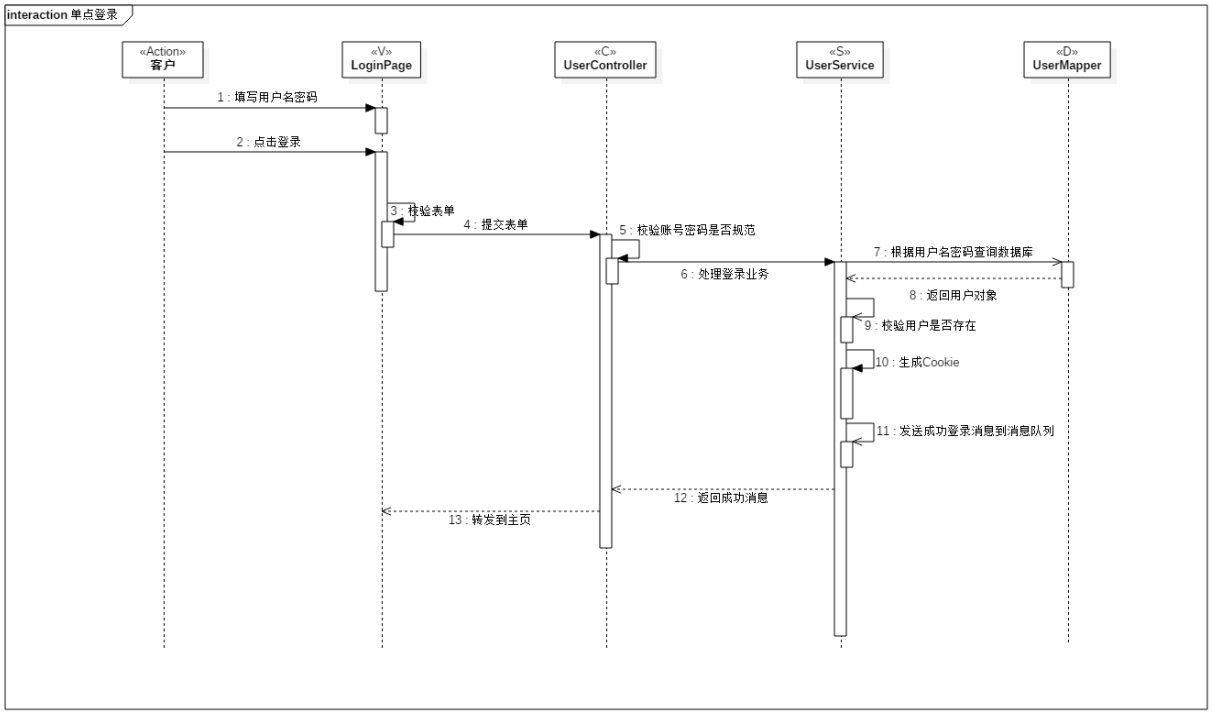
下单

7、 实验四：交互建模 – 顺序模型

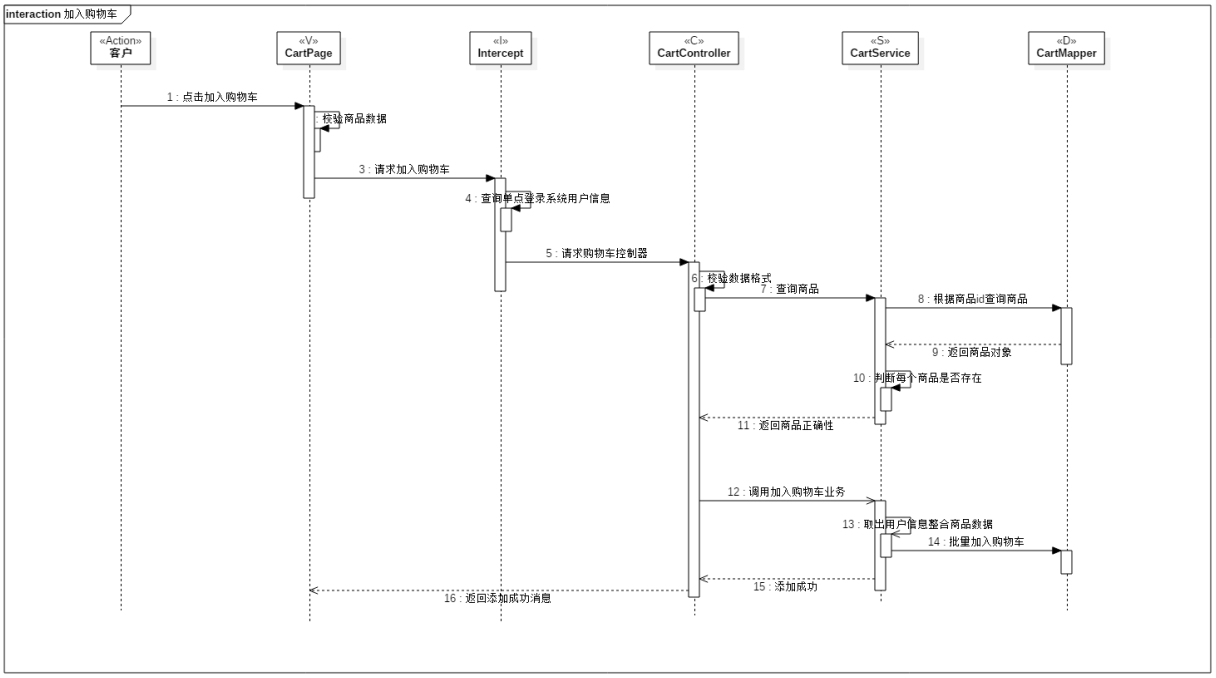
创建各个类（MVC 及 Actor）的对象，并描述对象之间的交互。

方法：分别创建参与者（Actor）、界面类（View）、控制器类（Controller）和模型类（Model）的对象，描述各个对象之间的消息及其顺序，画出顺序图。

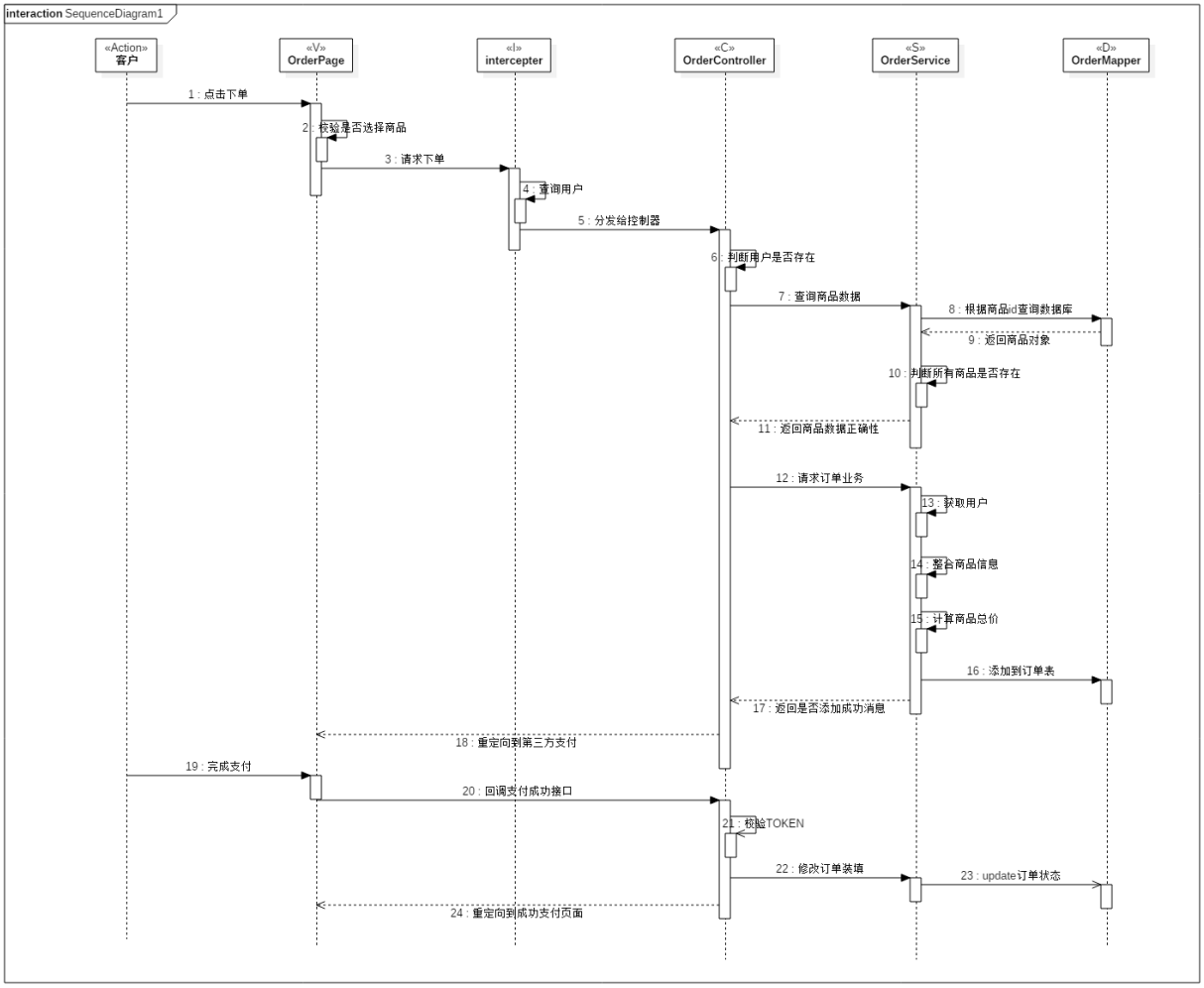
参考：讲义 P33 页 8.7.2。



单点登录顺序图



加入购物车时序图



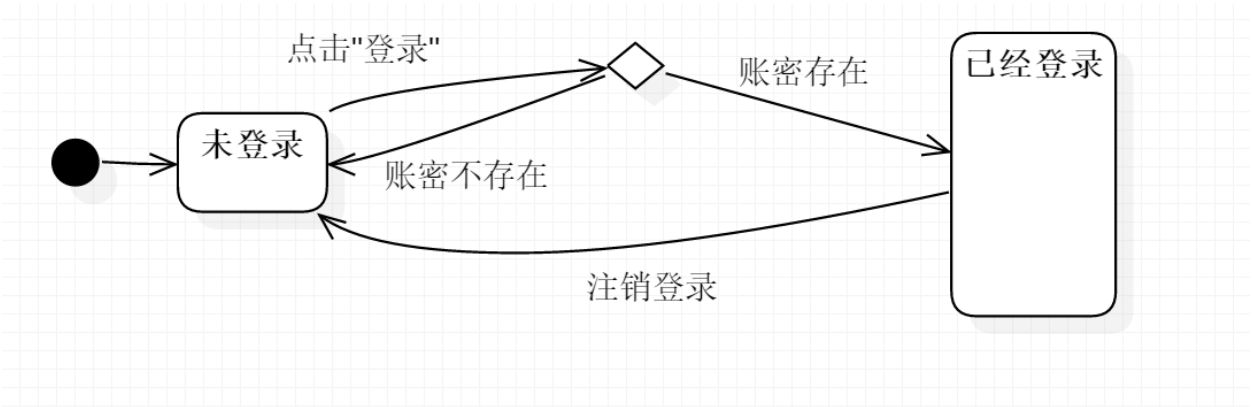
订单时序图

8、 实验五：状态建模 – 状态模型

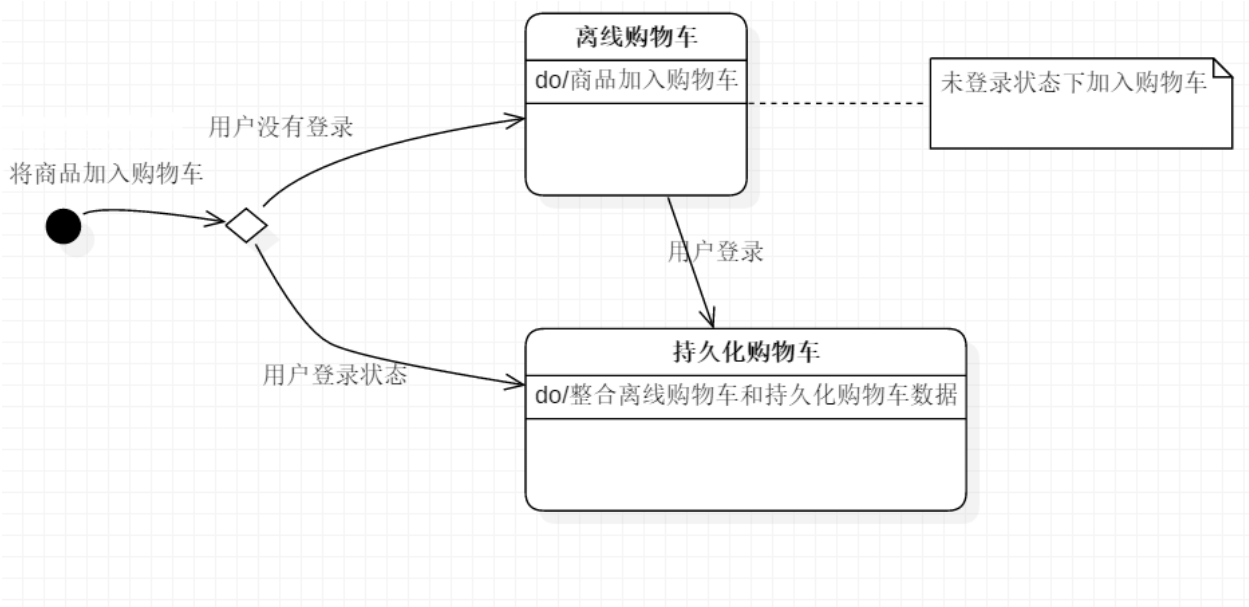
对系统中最重要的对象进行状态建模。

方法：选择一种对象，定义该对象的状态，描述状态之间的切换及条件，画出状态图。

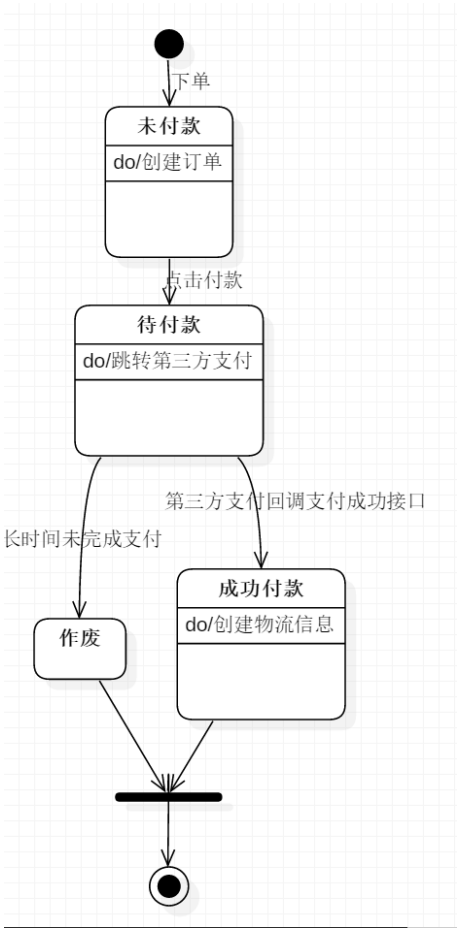
参考：讲义 P9 和 P10 页。



登录状态转换图



购物车状态转化图



订单状态转换图

五、实验体会

实验一：

不是第一次做 UML 图，原来的设想是分析之前做的淘淘商城，但是发现业务太多了， 就选了三个比较有代表性的业务来做，用例图，实际上我是将原来的项目代码做了一些优化设计之后翻译成白话文写的，之前也写过用例图，但是使用 Visio 来做的，原来作图的时候考虑得比较简单，没有关心语义上的错误，做实验得过程中在老师的指导下纠正过来了。

实验二：

因为自己对模块的业务逻辑比较了解，基本上一口气就可以写完，因为有前面用例图，按照实验一的流程用流程图的方式描述出来就可以了，因为业务逻辑比较复杂，图画得比较大，这里的程序流程实际上和流程控制引擎 Activiti 的绘制过程相当相似，但是显然软件工程在前，Activiti 在后，保证了 Activiti 和软件工程绘制的一致性，降低了自身的学习难度

实验三：

这次实验交的很慢，因为最近忙几个项目，没顾忌作业，真是抱歉，这次实验利用了 myclipse 的类图反向工程，将原来的代码导出类图，并参照其中的类图做出来的，这次作业是我最困惑的，业务逻辑都在程序代码里面，画出类图，要舍去一部分代码的考虑，画出来的效果不尽人意，但是实验五应该会弥补这个缺陷，这个图应该是我认为最没必要的软件流程了，画出来了但是对开发者没有实际的作用，和实际项目类关系差别也很大，很多表示方法繁多但是又很难表示出其中的关系，比如拦截器与控制器的关系，注册服务与拉取服务的关系，感觉这个图就是多此一举，降低开发效率，软件工程很好，规范做得很到位，但是对一个企业来讲要的是敏捷、效率，就像数据库各种范式不被遵守一个道理，书上写得很好，很对，但是冗余设计和中断设计更符合实际需求。

实验四：

这次实验对比实验三的体验就好多了，由于我有些业务写了拦截器相关的操作，时序图又是只画成功状态的，所以我有些图跳过了正确的状态，时序图在企业中还是比较常见的，它并不像类图一样只是说明类之间的关系，留给程序员的发挥空间就很少，还有可能导致误解，但是时序图可以看出每个业务层次都完成了什么操作，清晰得看出了每个业务的流程，又留给了程序员书写的空间，类图在具备 J2E 的情况下可以说没有任何意义的，J2E 已经规范了所有的类结构要怎么放，甚至连接口的规范都定义得很好，例如 RESTFull，已经帮助解决了类关系，而时序图讲的是业务流程，应用场景更加接近企业应用。

实验五：

那个<<M>>没有使用，因为我理解的 M 是一个 Javabean，Javabean 不作为任何一个应用层次存在，“淘淘商城”后端架构分为三层：控制层、服务层、持久化层，每个层次为了实现解耦，消息传递的方式只通过传递基本数据类型和 Javabean，如果一定要将控制层的某些对象数据耦合到服务层使用，也不同通过消息传递，二是通过 ThreadLocal 本地线程传递，这个解耦的方案在 Spring MVC 和 Struts2 中都有体现，所以<<M>>是在每一层都被使用的关系，我再做图的过程中发现了<<M>>这个没写，如果写了每一层都会与<<M>>产生连接，时序图就会看起来很乱，所以没有写，而是作为消息传递了，如果时序图中一定要有一个<<M>>那么那个<<D>>就是<<M>>，因为<<D>>是持久化层，实现了从类到数据库表的映射关系，可以看做一个模型。

最后一个实验室状态转换图，这个图算是比较简单的吧，因为登录、下单、购物车的状态并不多，比时序图写起来要容易，写的过程中也会发现，就一个系统而言，比如订单的状态应该不止那么多，可有没有考虑全的地方。