**武汉大学计算机学院**

**本科生实验报告**

**武汉大学活动发布中心设计模式分析**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：软件设计与体系结构

指 导 教 师 一：王健

团 队 成 员 一： 陈翔宇（2018326660093）

团 队 成 员 二： 陈郎（2018326660226）

# 设计模式选择

## 简单工厂模式

### 简单工厂模式概述

简单工厂模式是属于创建型模式，又叫做静态工厂方法（Static Factory Method）模式，但不属于23种GOF设计模式之一。简单工厂模式是由一个工厂对象决定创建出哪一种产品类的实例。简单工厂模式是工厂模式家族中最简单实用的模式，可以理解为是不同工厂模式的一个特殊实现。

### 简单工厂模式的优点

工厂类是整个模式的关键.包含了必要的逻辑判断,根据外界给定的信息,决定究竟应该创建哪个具体类的对象.通过使用工厂类,外界可以从直接创建具体产品对象的尴尬局面摆脱出来,仅仅需要负责“消费”对象就可以了。而不必管这些对象究竟如何创建及如何组织的．明确了各自的职责和权利，有利于整个[软件体系结构](https://baike.baidu.com/item/%E8%BD%AF%E4%BB%B6%E4%BD%93%E7%B3%BB%E7%BB%93%E6%9E%84" \t "_blank)的优化。

### 简单工厂模式的缺点

由于工厂类集中了所有实例的创建逻辑，违反了[高内聚](https://baike.baidu.com/item/%E9%AB%98%E5%86%85%E8%81%9A" \t "_blank)责任分配原则，将全部创建逻辑集中到了一个工厂类中；它所能创建的类只能是事先考虑到的，如果需要添加新的类，则就需要改变工厂类了。

当系统中的具体产品类不断增多时候，可能会出现要求工厂类根据不同条件创建不同实例的需求．这种对条件的判断和对具体产品类型的判断交错在一起，很难避免模块功能的蔓延，对系统的维护和扩展非常不利；

这些缺点在[工厂方法模式](https://baike.baidu.com/item/%E5%B7%A5%E5%8E%82%E6%96%B9%E6%B3%95%E6%A8%A1%E5%BC%8F)中得到了一定的克服。

## 工厂方法模式

### 工厂方法模式概述

工厂方法(Factory Method)模式的意义是定义一个创建产品对象的工厂接口，将实际创建工作推迟到子类当中。核心工厂类不再负责产品的创建，这样核心类成为一个抽象工厂角色，仅负责具体工厂子类必须实现的接口，这样进一步抽象化的好处是使得工厂方法模式可以使系统在不修改具体工厂角色的情况下引进新的产品。

### 工厂模式优点

工厂方法模式是[简单工厂模式](https://baike.baidu.com/item/%E7%AE%80%E5%8D%95%E5%B7%A5%E5%8E%82%E6%A8%A1%E5%BC%8F" \t "_blank)的衍生，解决了许多简单工厂模式的问题。首先完全实现‘开－闭 原则’，实现了可扩展。其次更复杂的层次结构，可以应用于产品结果复杂的场合。

### 工厂模式缺点

如果需要创建的子类越来越多，每个子类都要对应一个工厂，会导致代码过于繁琐，抽象工厂模式从一定程度上解决了这个问题。

## MVC模式

### MVC模式概述

[MVC](https://baike.baidu.com/item/MVC)是模型(model)－视图(view)－控制器(controller)的缩写，该模型用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC被独特的发展起来用于映射传统的输入、处理和输出功能在一个逻辑的图形化用户界面的结构中。

* Model（模型）是应用程序中用于处理应用程序数据逻辑的部分,通常模型对象负责在数据库中存取数据。
* View（视图）是应用程序中处理数据显示的部分,通常视图是依据模型数据创建的。
* Controller（控制器）是应用程序中处理用户交互的部分,通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

### MVC模式优点

在MVC模式中，三个层各施其职，所以如果一旦哪一层的需求发生了变化，就只需要更改相应的层中的代码而不会影响到其它层中的代码。而且分层后更有利于组件的重用。如控制层可独立成一个能用的组件，视图层也可做成通用的操作界面。

### MVC模式缺点

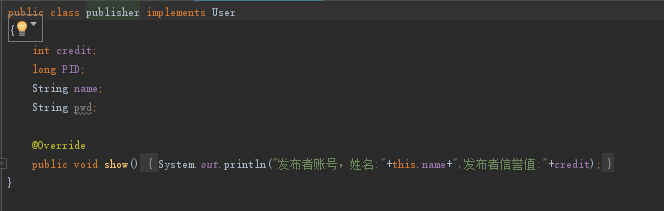
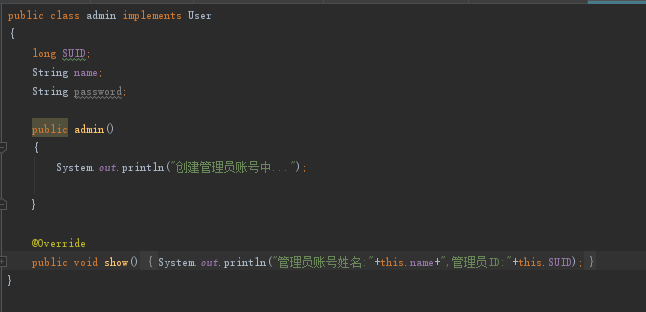
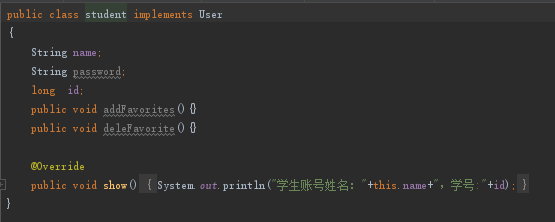
增加了系统结构和实现的复杂性。对于简单的界面，严格遵循MVC，使模型、视图与控制器分离，会增加结构的复杂性，并可能产生过多的更新操作，降低运行效率。还有视图与控制器间的过于紧密的连接，视图没有控制器的存在，其应用是很有限的，反之亦然，这样就妨碍了他们的独立重用。

1. **设计思路**

## 简单工厂设计思路

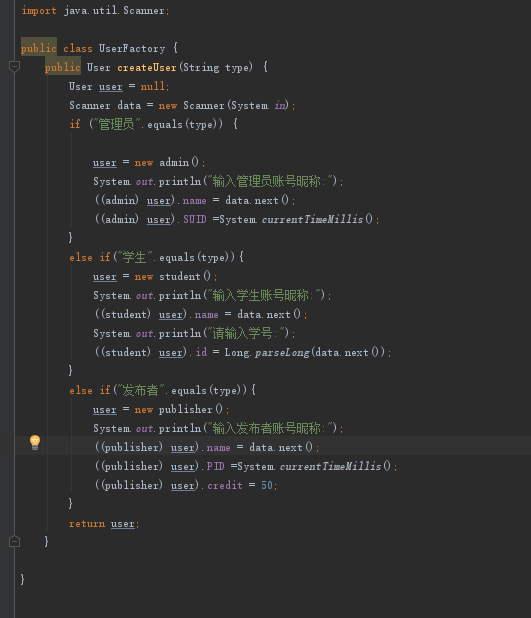
在我们的活动中心平台中，我们用简单工厂模式实现用户的创建，因为用户创建需要一系列阀组的初始化操作，如后期需要关联其他对象成员、查配置文件、查数据库表，如果把这些初始化逻辑都卸载构造函数中构造函数的代码会变得很长，大大降低程序的可读性，考虑到用户的种类比较少，且一般不会有新增的种类，所以选用简单工厂模式对其进行生成。

具体思路是将用户类User作为一个抽象接口，用三个子类：学生(Student),管理员(Admin),发布者(Publisher)分别实现这个接口



三个子类有不同的字段和方法，都重写了接口中的show()方法用来展示账号信息。

定义一个工厂类UserFactory。在创建方法时传入参数type，根据参数来做条件判断，决定创建什么样的用户。程序其他部分无需了解工厂内部的实现过程。



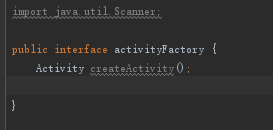
在工厂创建对象时对对象进行构造。这里的PID和SID都用了时间戳来进行生成。

## 工厂方法模式

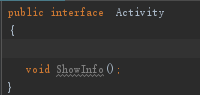
我们的项目用到工厂方法模式来发布信息，不选用简单工厂模式的原因是，考虑到信息的发布有很多类别，无法在第一次编程时全部考虑，如果用简单工厂模式要增加新的信息子类，信息工厂的创建方法中就要不断增加新的if-else语句，这不符合面对对象的开放-封闭原则，不但容易出错，可读性也不佳。所以我们用每一个信息子类创建对应的工厂，这些工厂子类分别实现抽象的工厂接口，这样一来，只要我们实例化不同的工厂子类，得到的就是对应的信息对象。

具体代码如下：

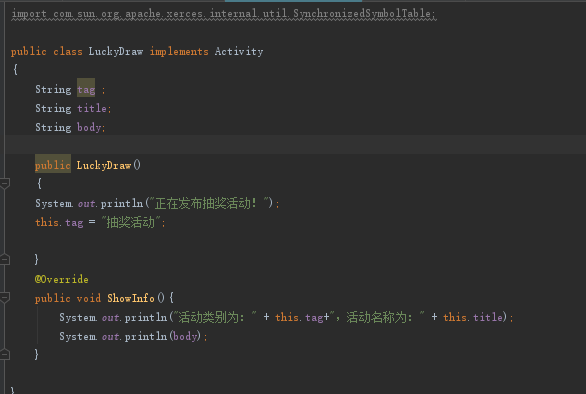
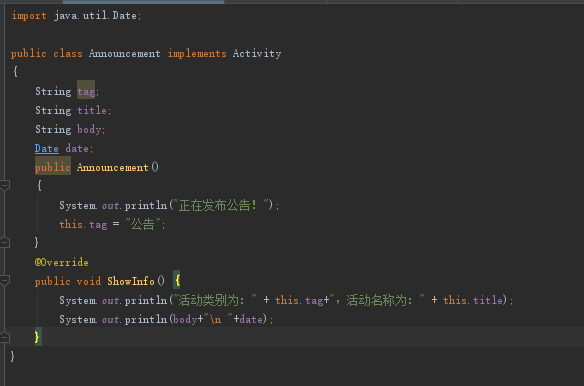
抽象工厂接口activityFactory



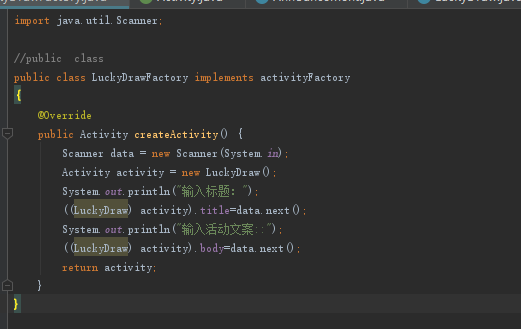
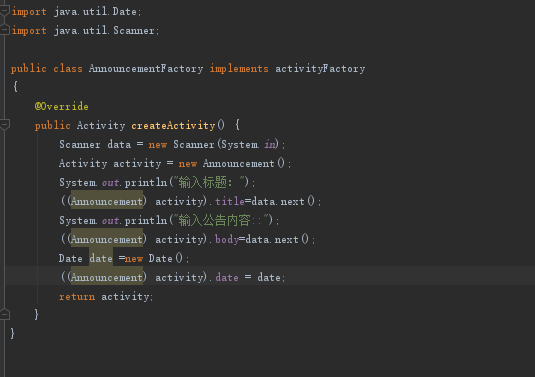
信息对象Activity



信息对象有两个子类分别实现了它:公告类(Announcement)、抽奖类(LuckDraw)

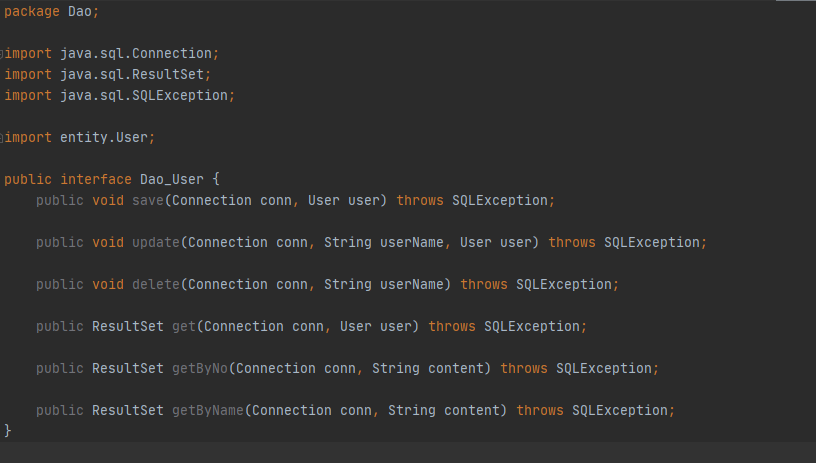


同样为他们创建两个工厂 实现于抽象工厂接口,分别是LuckDrawFactory、AnnouncementFactory.

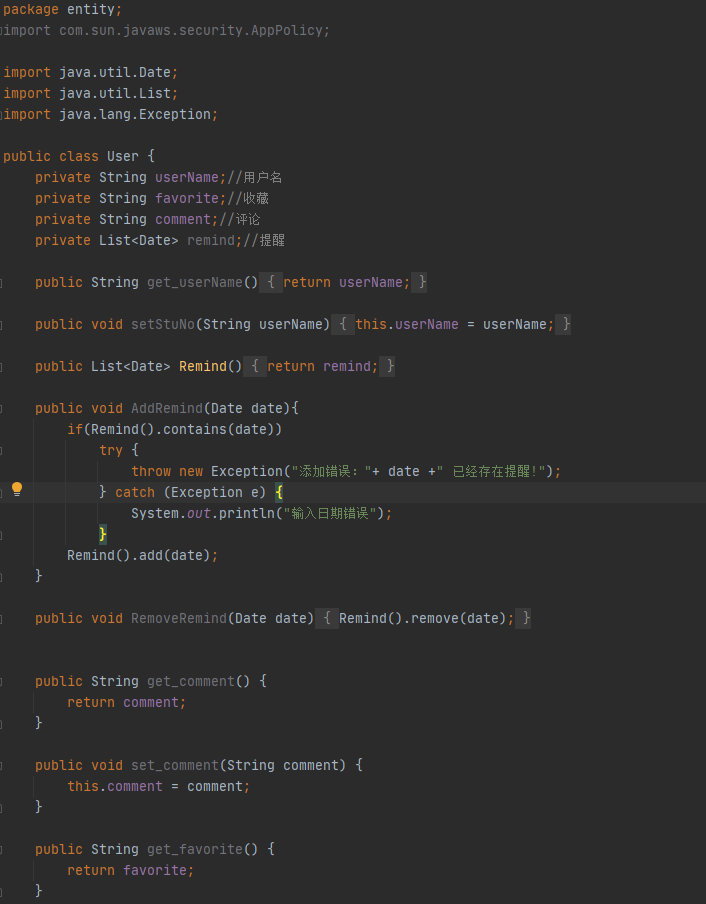


这样在客户端，想创建什么样的信息对象只需要实例化不同的工厂子类，调用相同的方法createActivity()，无需再传入参数。程序的耦合度大大降低。

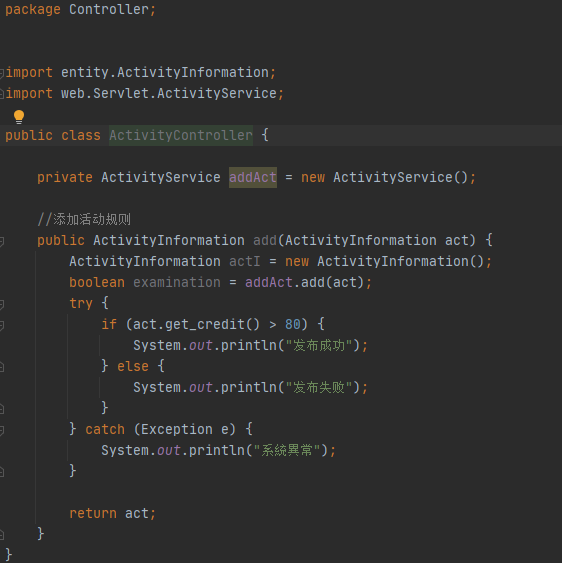
## MVC设计思路

Servlet用于过程控制，implement用于数据库储存，Dao用于数据库接口，entity用于封装业务逻辑，Web页面显示未完成。首先Controller接受用户的请求，并决定用哪个Model来进行处理，然后Model用业务逻辑来处理用户的请求并返回数据，最后Contriller用响应的View格式化Model返回的数据，并通过表示层呈现给用户。

Dao







controller

