第二周总结

首先是课程学习过程中的笔记：

2.1 从编程历史看面向对象的本质与未来

1. 简述了计算机行业的由来以及面向对象编程产生的原因
2. 面向对象的编程需要对现实中的实体进行抽象，弄清楚内涵和外延
3. 掌握好多态才能掌握好面向对象
4. 由框架来控制整体的运行结构，而我们的代码则依附于这些框架
5. 框架Vs工具：框架调用程序，程序调用工具

2.2 设计臭味：糟糕的代码有哪些特点

1. 一个不能交互，没有行为的东西是不能成为对象的
2. 软件设计的最终目的，是使软件“强内聚，松耦合”

易扩展：易于添加新的功能

更强壮：不容易被粗心的程序员破坏

可移植：能够在多样的环境下运行

更简单：容易理解，容易维护

2.3 开闭原则介绍以及代码分析

1. 开闭原则含义：对扩展开放，对更改封闭。更进一步说就是扩展功能时不需要修改代码。
2. 对照课程内容和自己的日常开发，目前已经无意识用到的有：

1）对于实体本身的抽象和扩展，比如button及其继承类

2）当需要产生依赖时，依赖一个接口，而不是具体的某个对象，耦合非常的松。比如课程中的Dailer，可以轻松替换成密码箱，电脑之类的东西

3. 对照课程内容，自己工作中尚未使用的可借鉴的方法：

1）adaptor模式，把对一个方法的调用转换成另一个

2）观察者模式：这个比较妙，调用自己的每个listener，新增功能时只要往里面注册新的listener即可。

2.4 依赖倒置原则介绍及代码案例分析

1. 用自己的话描述依赖倒置原则：高层代码不依赖低层代码，只依赖一个抽象，并且这个抽象属于高层。高层实现时可以没有低层的存在，而低层的接入则只需要实现抽象。最后达到的结果是高层调用低层，但是高层不依赖低层。此处高层是框架，比如说tomcat，tomcat调用业务的代码，但是tomcat不依赖业务本身，业务接入时需要实现tomcat的接口。tomcat可以被任何人复用。其中一些著名高层所依赖的抽象，便是行业的规范，比如tomcat依赖j2ee规范
2. 依赖倒置是框架代码所遵循的一个基本原则。
3. 好莱坞（框架）：don’t call me. I’ll call you

2.5 里氏替换原则

1. 不同于传统的“is a”的理解方式。里氏替换原则主要看程序里用子类替换父类程序能否正常运行，能正常运行才是一个合理的继承
2. 不符合“is a”的理解方式的一定不符合里氏替换原则
3. 看待一个对象核心是要看待它的行为

4. 一些违反里氏替换原则的征兆：

1）派生类中对某些方法产生了退化（最近工作中某个类的设计确实有这个问题）

2）派生类中抛出了基类不会产生的异常

2.6 单一职责接口隔离：

1. 单一职责是指一个类只有一个引起它变化的原因

2. 违反单一职责原则会导致程序脆弱，并且不可移植

3. 一个类不应该超过一屏

4. 接口隔离原则：通过接口隔离不同的实现类的方法。使得调用方不被强迫依赖不需要的方法。

5. 一些很难做到单一职责的类可以通过接口隔离原则进行拆解

2.7 案例：反应式编程框架FLower的设计

1. flower解决的问题：高并发环境下，线程的阻塞导致资源的占用，最终导致系统的崩溃

感想：

这一周的课程可以说是干货满满，以前我做过2件事，一件是尝试阅读Spring代码，没读懂，另一件是看设计模式相关的书，知道了有23种设计模式，然而并没有什么用处。在这周课程的帮助下，这两个点开始连成线了，读框架的代码，写框架的代码都需要带着框架设计的知识去看。课程中李老师说的依赖倒置原则算是道出了框架的本质了吧，以后可以带着这个思路，再去试着研究一些开源的代码，并促进自己的思考和分析。