链滴

Java 程序设计的一点总结

作者：EricTao

原文链接：https://ld246.com/article/1531817235877

来源网站：链滴

前言

《阿里巴巴JAVA开发手册》的推出，迅速成为中国JAVA程序员编码既定规范。但是在程序的设计上则需要依靠个人的自由发挥。

作为一名菜鸟程序员，在系统层次的设计上没有太多的经验，但是在细节处，我们能设计得巧妙而精。

函数

所有的编程都是从HellWorld这个小函数开始的，学会设计函数非常重要

1. 函数要 **短。**短才方便阅读、维护和设计。(每个人都经历过读不懂自己代码的尴尬)

2. 函数只做 **一件事。**依照单一职责原则设计函数。一个函数可以:流程控制，获取数据，逻辑判断数据处理，或者调用多个下一抽象级的函数。

只要逻辑上能拆分，并存在复用的可能，就拆分开。业务复杂度越高，这些函数复用的概率也就越高。

3. 如果整个函数非常复杂，应该分解成 **多个抽象层级**设计，高层函数调用下次层函数，呈树状图，层封装，最后由一个函数统筹调用各个高层函数。

4. 函数 **不应该有标识参数**(除了作为API的函数)，这意味着函数有至少两种执行方式，违反了第2条则。而且明显能拆成多个小函数。

5. 函数参数越少越好有， **多个参数应该封装**成一个整体传入的。如果逻辑上不是一个整体，则函数定能被拆成多个小函数然后被分别调用。第4条**标识参数可以封装**进整体传入函数，而不是直接作为数的参数

6. 函数 **不应该改变参数状态**，如果想改变某类的状态，就把该函数加入该类，让它自己调用函数。：把改变类x的状态的函数调用addFooter(x)，改为x.addFooter()。

如果改变参数状态的函数是工具类方法或多个类通用，不适合把该函数加入各个类（因为会有大量重代码）或继承体系。可以选择返回改变后的类，如：addFooter(x)会返回改变后的x。

7. 函数　**不要返回错误码**，这需要你有错误码的枚举类，并且违反了开放封闭原则（你需要加入新错码来扩展新错误），直接抛出异常就好了。（可以通过继承父异常来扩展）。

8. 函数　**名称应该描述清楚函数作用**，避免频繁去看文档，这对于短小的函数来说不难办到，如果很命名可能需要思考函数是否有依照以上原则设计(你一个函数可能做了很多事情)。

并且名称的命名应该不容易与其他函数名称形成混淆。如：add()在calculator中意思是加，而在List就不应该用add表示插入集合了，应该用insert或append。简单来说就是一个概念对应一个词，并且终如一。

格式

行业格式规范是强迫症的福音

请参考《阿里巴巴JAVA开发手册》编写。

注释

其实注释写了后，看的最多的还是自己

1. 好代码只需要少量注释，代码就能表达意图——回到之前的内容，这要求我们写小且精的函数。（这不是你不写注释的理由）

2. 好的注释应该是这样的。如：对抽象意图或者深远意义的解释（如我这个函数为啥用这个方法实现；阐述长且难读的函数（这种难读不是因为代码写得烂，而是业务逻辑复杂）；

警示一些关键重要的部分（这些部分一般是函数会改变某些数据，或哪些非参数的数据会影响到函数；TODO注释提醒并告知未来要做的事；学着公共API的JAVADOC写就是好注释（虽然也有少数烂注）；

3. 烂的注释往往是这样的。如：多余的注释（简单函数强行加上注释，读源码会比注释更快）；误导注释（注释本来就是错的，可能源自你更新了代码没更新注释——小函数特别可能出现这个问题）；释掉的代码（你为何不删了代码？）；废话太多的注释（语文是体育老师教的）。

类

类的设计思想其实和函数的设计是类似的，所以类应该功能单一且小巧，越小耦合性越低，最后用门模式组合起来向外提供API就好了

1. 类的设计和函数设计不同在于，类存在继承体系。可以利用继承来达到复用，同时也要避免继承带的冗余代码（逻辑上不是继承体系中通用的就不应该加入继承体系）。

2. 类存在继承体系和接口，所以在设计时要多使用继承和接口，将多个特性逐层细化。如：父类是车实现了驾驶的接口或方法；子类是坦克车，实现了开炮的接口或方法。

不同层次的类应该有不同的抽象特性。

3. 类与类之间应该依赖于接口（依赖倒置原则）。如：坦克车类，持有某类炮弹。不应该直接持有具的炮弹类（如：持有破甲弹），应该持有的是坦克炮弹这个接口。

4. 看看面向对象六大原则吧。

系统

系统整体结构的一点想法，这里不谈架构只谈代码上的设计

1. 把系统的整体构造和业务使用分开。不要让构造影响使用，也不要让程序的运行反过来影响构造。也是广泛使用的Spring就是典型例子，Spring Core就是个类容器并帮助构造系统。

2. 把业务逻辑和检查或日志方案分离，不然纠缠在一起的代码会很难看懂和修改。Spring AOP也解了这个问题。

3. 系统的各个模块应该像类和函数一样设计，内部封装好各个功能且不对外开发，最后由某个类统筹划，提供一个门面供其他模块使用。（代码的封装与复用贯穿在编程的各个层次）

测试

测试不仅仅是测试小姐姐的事情

1. 测试函数（方法）也应该短小（如果函数原本够小的话测试函数自然会小）

2. 每个类最好都测试下，测试时间会比以后debug时间少。从底层函数一点点测上去，保证底层函数正确性后，debug时定位问题会非常简单且迅速。

3. 测试类应该保存下来，方便每次修改后进行测试

以上算个人读书笔记，于2017-09-27 23:43:54

期待于您共鸣