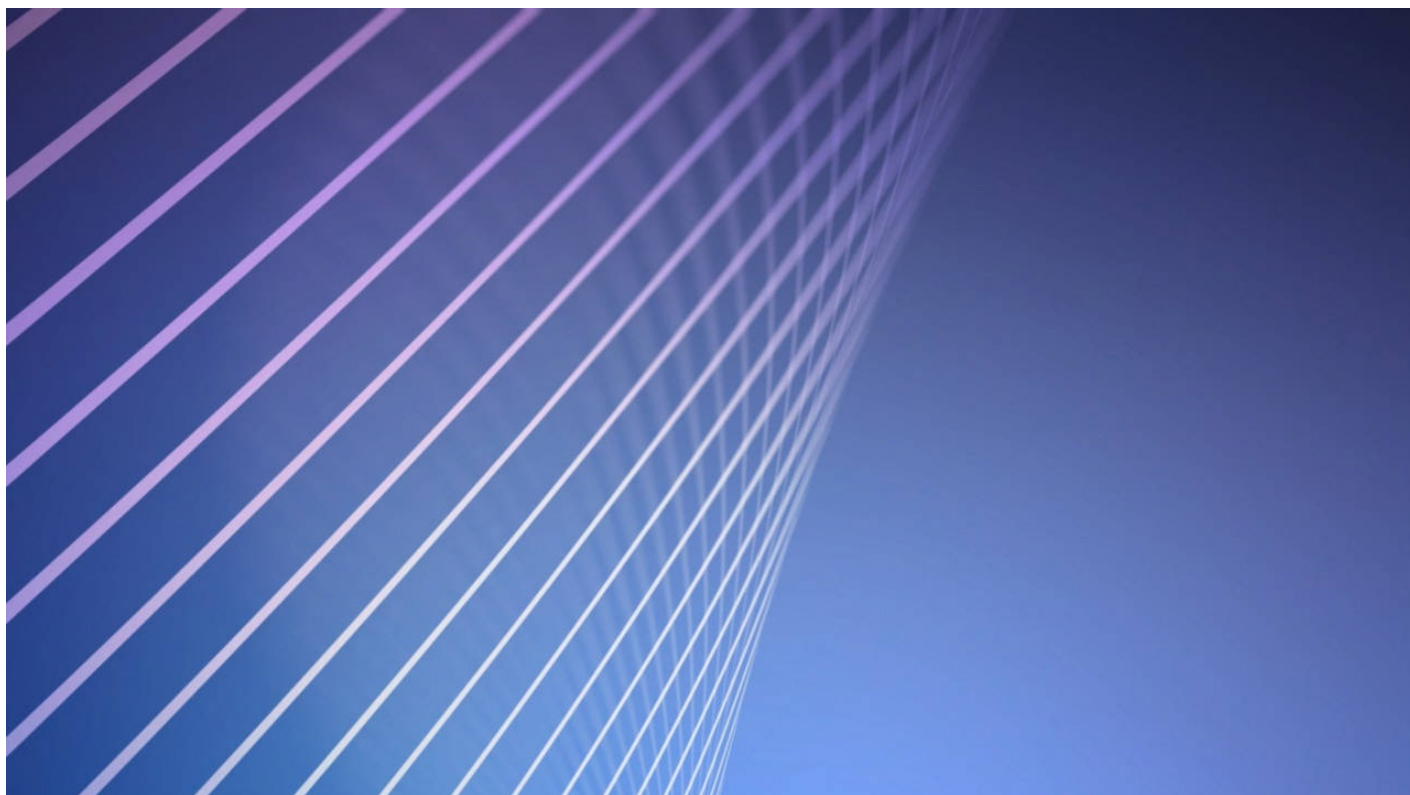


31讲理论五：让你最快速地改善代码质量的20条编程规范（上）



前面我们讲了很多设计原则，后面还会讲到很多设计模式，利用好它们可以有效地改善代码质量。但是，这些知识的合理应用非常依赖个人经验，用不好有时候会适得其反。而我们接下来要讲的编码规范正好相反。编码规范大部分都简单明了，在代码细节方面，能立竿见影地改善质量。除此之外，我们前面也讲到，持续低层次、小规模重构依赖的基本上都是编码规范，这也是改善代码可读性的有效手段。

关于编码规范、如何编写可读代码，很多书籍已经讲得很好了，我在前面的加餐中也推荐过几本经典书籍。不过，这里我根据我自己的开发经验，总结罗列了20条我个人觉得最好用的编码规范。掌握这20条编码规范，能让你最快速地改善代码质量。因为内容比较多，所以，我分为三节课来讲解，分别介绍编码规范的三个部分：命名与注释（Naming and Comments）、代码风格（Code Style）和编程技巧（Coding Tips）。

命名

大到项目名、模块名、包名、对外暴露的接口，小到类名、函数名、变量名、参数名，只要是做开发，我们就逃不过“起名字”这一关。命名的好坏，对于代码的可读性来说非常重要，甚至可以说是起决定性作用的。除此之外，命名能力也体现了一个程序员的基本编程素养。这也是我把“命名”放到第一个来讲解的原因。

取一个特别合适的名字是一件非常有挑战的事情，即便是对母语是英语的程序员来说，也是如此。而对于我们这些英语非母语的程序员来说，想要起一个能准确达意的名字，更是难上加难了。

实际上，命名这件事说难也不难，关键还是看你重不重视，愿不愿意花时间。对于影响范围比较大的命名，比如包名、接口、类名，我们一定要反复斟酌、推敲。实在想不到好名字的时候，可以去GitHub上用相关的关键词联想搜索一下，看看类似的代码是怎么命名的。

那具体应该怎么命名呢？好的命名有啥标准吗？接下来，我就从4点来讲解我的经验。

1.命名多长最合适？

在过往的团队和项目中，我遇到过两种截然不同的同事。有一种同事特别喜欢用很长的命名方式，觉得命名一定要准确达意，

哪怕长一点也没关系，所以，这类同事的项目里，类名、函数名都很长。另外一种同事喜欢用短的命名方式，能用缩写就尽量用缩写，所以，项目里到处都是包含各种缩写的命名。你觉得这两种命名方式，哪种更值得推荐呢？

在我看来，尽管长的命名可以包含更多的信息，更能准确直观地表达意图，但是，如果函数、变量的命名很长，那由它们组成的语句就会很长。在代码行长度有限的情况下，就会经常出现一条语句被分割成两行的情况，这其实会影响代码可读性。

实际上，在足够表达其含义的情况下，命名当然是越短越好。但是，大部分情况下，短的命名都没有长的命名更能达意。所以，很多书籍或者文章都不推荐在命名时使用缩写。对于一些默认的、大家都比较熟知的词，我比较推荐用缩写。这样一方面能让命名短一些，另一方面又不影响阅读理解，比如，sec表示second、str表示string、num表示number、doc表示document。除此之外，对于作用域比较小的变量，我们可以使用相对短的命名，比如一些函数内的临时变量。相反，对于类名这种作用域比较大的，我更推荐用长的命名方式。

总之，命名的一个原则就是以能准确达意为目标。不过，对于代码的编写者来说，自己对代码的逻辑很清楚，总感觉用什么样的命名都可以达意，实际上，对于不熟悉你代码的同事来讲，可能就不这么认为了。所以，命名的时候，我们一定要学会换位思考，假设自己不熟悉这块代码，从代码阅读者的角度去考量命名是否足够直观。

2.利用上下文简化命名

我们先来看一个简单的例子。

```
public class User {  
    private String userName;  
    private String userPassword;  
    private String userAvatarUrl;  
    //...  
}
```

在User类这样一个上下文中，我们没有在成员变量的命名中重复添加“user”这样一个前缀单词，而是直接命名为name、password、avatarUrl。在使用这些属性时候，我们能借助对象这样一个上下文，表意也足够明确。具体代码如下所示：

```
User user = new User();  
user.getName(); // 借助user对象这个上下文
```

除了类之外，函数参数也可以借助函数这个上下文来简化命名。关于这一点，我举了下面这个例子，你一看就能明白，我就不多啰嗦了。

```
public void uploadUserAvatarImageToAliyun(String userAvatarImageUri);  
//利用上下文简化为：  
public void uploadUserAvatarImageToAliyun(String imageUri);
```

3.命名要可读、可搜索

首先，我们来看，什么是命名可读。先解释一下，我这里所说的“可读”，指的是不要用一些特别生僻、难发音的英文单词来命名。

过去我曾参加过两个项目，一个叫plateaux，另一个叫eyrie，从项目立项到结束，自始至终都没有几个人能叫对这两个项目的

名字。在沟通的时候，每当有人提到这两个项目的名字的时候，都会尴尬地卡顿一下。虽然我们并不排斥一些独特的命名方式，但起码得让大部分人看一眼就能知道怎么读。比如，我在Google参与过的一个项目，名叫inkstone，虽然你不一定知道它表示什么意思，但基本上都能读得上来，不影响沟通交流，这就算是一个比较好的项目命名。

我们再来讲一下命名可搜索。我们在IDE中编写代码的时候，经常会用“关键词联想”的方法来自动补全和搜索。比如，键入某个对象“.get”，希望IDE返回这个对象的所有get开头的方法。再比如，通过在IDE搜索框中输入“Array”，搜索JDK中数组相关的类。所以，我们在命名的时候，最好能符合整个项目的命名习惯。大家都用“selectXXX”表示查询，你就不要用“queryXXX”；大家都用“insertXXX”表示插入一条数据，你就要不用“addXXX”，统一规约是很重要的，能减少很多不必要的麻烦。

4.如何命名接口和抽象类？

对于接口的命名，一般有两种比较常见的方式。一种是加前缀“I”，表示一个Interface。比如IUserService，对应的实现类命名为UserService。另一种是不加前缀，比如UserService，对应的实现类加后缀“Impl”，比如UserServiceImpl。

对于抽象类的命名，也有两种方式，一种是带上前缀“Abstract”，比如AbstractConfiguration；另一种是不带前缀“Abstract”。实际上，对于接口和抽象类，选择哪种命名方式都是可以的，只要项目里能够统一就行。

注释

命名很重要，注释跟命名同等重要。很多书籍认为，好的命名完全可以替代注释。如果需要注释，那说明命名不够好，需要在命名上下功夫，而不是添加注释。实际上，我个人觉得，这样的观点有点太过极端。命名再好，毕竟有长度限制，不可能足够详尽，而这个时候，注释就是一个很好的补充。

1.注释到底该写什么？

注释的目的就是让代码更容易看懂。只要符合这个要求的内容，你就可以将它写到注释里。总结一下，注释的内容主要包含这样三个方面：做什么、为什么、怎么做。我来举一个例子给你具体解释一下。

```
/**
 * (what) Bean factory to create beans.
 *
 * (why) The class likes Spring IOC framework, but is more lightweight.
 *
 * (how) Create objects from different sources sequentially:
 * user specified object > SPI > configuration > default object.
 */
public class BeansFactory {
    // ...
}
```

有些人认为，注释是要提供一些代码没有的额外信息，所以不要写“做什么、怎么做”，这两方面在代码中都可以体现出来，只需要写清楚“为什么”，表明代码的设计意图即可。我个人不是特别认可这样的观点，理由主要有下面3点。

- 注释比代码承载的信息更多

命名的主要目的是解释“做什么”。比如，void increaseWalletAvailableBalance(BigDecimal amount)表明这个函数用来增加钱包的可用余额，boolean isValidatedPassword表明这个变量用来标识是否是合法密码。函数和变量如果命名得好，确实可以不用再在注释中解释它是做什么的。但是，对于类来说，包含的信息比较多，一个简单的命名就不够全面详尽了。这个时候，在注释中写明“做什么”就合情合理了。

- 注释起到总结性作用、文档的作用

代码之下无秘密。阅读代码可以明确地知道代码是“怎么做”的，也就是知道代码是如何实现的，那注释中是不是就不用写“怎么做”了？实际上也可以写。在注释中，关于具体的代码实现思路，我们可以写一些总结性的说明、特殊情况的说明。这样能够让阅读代码的人通过注释就能大概了解代码的实现思路，阅读起来就会更加容易。

实际上，对于有些比较复杂的类或者接口，我们可能还需要在注释中写清楚“如何用”，举一些简单的quick start的例子，让使用者在不阅读代码的情况下，快速地知道该如何使用。

- 一些总结性注释能让代码结构更清晰

对于逻辑比较复杂的代码或者比较长的函数，如果不好提炼、不好拆分成小的函数调用，那我们可以借助总结性的注释来让代码结构更清晰、更有条理。

```
public boolean isValidPasword(String password) {  
    // check if password is null or empty  
    if (StringUtils.isBlank(password)) {  
        return false;  
    }  
  
    // check if the length of password is between 4 and 64  
    int length = password.length();  
    if (length < 4 || length > 64) {  
        return false;  
    }  
  
    // check if password contains only a~z,0~9,dot  
    for (int i = 0; i < length; ++i) {  
        char c = password.charAt(i);  
        if (!((c >= 'a' && c <= 'z') || (c >= '0' && c <= '9') || c == '.')) {  
            return false;  
        }  
    }  
    return true;  
}
```

2.注释是不是越多越好？

注释太多和太少都有问题。太多，有可能意味着代码写得不够可读，需要写很多注释来补充。除此之外，注释太多也会对代码本身的阅读起到干扰。而且，后期的维护成本也比较高，有时候代码改了，注释忘了同步修改，就会让代码阅读者更加迷惑。当然，如果代码中一行注释都没有，那只能说明这个程序员很懒，我们要适当督促一下，让他注意添加一些必要的注释。

按照我的经验来说，类和函数一定要写注释，而且要写得尽可能全面、详细，而函数内部的注释要相对少一些，一般都是靠好的命名、提炼函数、解释性变量、总结性注释来提高代码的可读性。

重点总结

好了，今天的内容到此就讲完了。我们来一块总结回顾一下，你需要掌握的重点内容。

1.关于命名

- 命名的关键是能准确达意。对于不同作用域的命名，我们可以适当地选择不同的长度。作用域小的变量（比如临时变量），可以适当地选择短一些的命名方式。除此之外，命名中也可以使用一些耳熟能详的缩写。
- 我们可以借助类的信息来简化属性、函数的命名，利用函数的信息来简化函数参数的命名。
- 命名要可读、可搜索。不要使用生僻的、不好读的英文单词来命名。除此之外，命名要符合项目的统一规范，不要用些反直觉的命名。
- 接口有两种命名方式：一种是在接口中带前缀“I”；另一种是在接口的实现类中带后缀“Impl”。对于抽象类的命名，也有两种方式，一种是带上前缀“Abstract”，一种是不带前缀。这两种命名方式都可以，关键是要在项目中统一。

2.关于注释

- 注释的目的就是让代码更容易看懂。只要符合这个要求的内容，你就可以将它写到注释里。总结一下，注释的内容主要包含这样三个方面：做什么、为什么、怎么做。对于一些复杂的类和接口，我们可能还需要写明“如何用”。
- 注释本身有一定的维护成本，所以并非越多越好。类和函数一定要写注释，而且要写得尽可能全面、详细，而函数内部的注释要相对少一些，一般都是靠好的命名、提炼函数、解释性变量、总结性注释来提高代码可读性。

课堂讨论

- 在讲到“用总结性注释让代码结构更清晰”的时候，我们举了一个isValidPassword()函数的例子，在代码可读性方面，这个函数还有哪些可以继续优化的地方呢？
- 关于注释，你推荐使用英文还是中文来书写呢？理由是什么呢？

欢迎在留言区写下你的答案，和同学一起交流和分享。如果有收获，也欢迎你把这篇文章分享给你的朋友。

精选留言



牛顿的烈焰激光剑
课堂讨论：

1. 关于 isValidPassword() 可读性优化：示例代码中的单行注释已经把验证规则清楚列明，但是必须打开源代码才能看见。我认为可以在函数声明处使用文档注释（即多行注释）对规则进行描述，这样函数的使用者借助 IDE 的代码提示功能就能看到具体的规则，同时也为用工具生成的项目文档提供注释。

2. 关于注释用中文还是英文的问题：对于团队开发，如果有外国人当然要用英文；但是如果只有中国人，我认为最好用中文。首先是每个人的外语水平不一，外语水平好的看到别人的语法错误甚至连单词都拼错，真的很影响心情。对于外语水平不太好的，使用外语写注释不友好且心理压力大，甚至回过头再看都不知道自己当初想表达什么。团队中最重要的是相互合作和最后上线的产品，而不是相互折磨，如果要求使用英文注释会抬高沟通成本，那就得不偿失。对于个人项目，选择中英注释均可，但应统一风格。我认为注释只是一个工具，用于降低沟通成本和提醒自己写代码时的思维逻辑和一些关键步骤，但是切不要对这个工具有过高的期望，譬如提高个人甚至团队的外语水平。

2020-01-13 10:39



zyl
什么时候开始 进入正题呀，前奏太长了

2020-01-13 16:36

作者回复

先从第一篇开始就是正题啊 前面的比后面的更重要呢 建议你回过头去再看下前几篇文章

2020-01-15 10:11



辣么大

There are only two hard things in Computer Science: cache invalidation and naming things.-- Phil Karlton

命名达意、准确：

不知道如何命名，推荐：Codelf(变量命名神器) <https://unbug.github.io/codelf/>

Search over projects from Github, Bitbucket, Google Code, Codeplex, Sourceforge, Fedora Project, GitLab to find real-world usage variable names.

关于注释语言：

公司的项目看项目要求（中英文都可以）

自己的个人项目一定要用英文，因为一开始我就考虑到要做国际化的项目（目标是全球用户）。

如何写注释可以多看看JDK源码中的注释，能够学到很多东西。

2020-01-13 11:53



阿卡牛

编程的两大难题：并发和起名

2020-01-13 11:03



李小四

设计模式_31

作业

1. 可能是为了举例吧，我认为这里的4个注释都是多余的。这些注释并没有比代码更简洁，阅读注释与阅读代码的理解成本是一样的。

2. 注释用中文还是英文，工作中一定要看团队成员的情况。个人推荐全部使用英文，从成长的角度来看，这有助于加强程序的英文读写能力，而这个能力是非常有价值的。

感想

写代码的时候，我常常问自己一个问题。如果后面有一个人要接收我的项目，他会不会骂我。。。我就暗自骂过很多人。。。所以在某些与常规逻辑明显不同的地方，我的注释量甚至多于代码量。当然注释也不是越多越好。

2020-01-13 09:07



Frank

所谓无规矩不成方圆。同样我们在做一些事情的时候是有套路可寻的。在写代码的时候我们也需要遵循一些前人总结好的规范，使得自己写代码更具有可读性，可维护性。

今天从这篇文章中学到了关于命名和注释的一些见解。关于命名，以前自己会犯的一个错是将拼音和英文会混在一起用，现在想想估计太年轻，命名搞得不伦不类。现在已经改了，一直督促自己用英文。命名的原则是准确达意，在平时的开发中都我都是尽量使用完整的单词，即使这样命名会很长，但是能够清楚的表达。文章中说的借助上下文来简化属性、函数的命名自己没有实践过。

关于注释，在自己的开发习惯中对于类和方法都是要写注释的，但是自己对注释的理解比较浅，只是写了做什么。并没有从文中提到的“做什么、为什么、怎么做”这三个维度去考虑。对于一些非常简单的方法，如只有几行，一眼就能读懂其要表达的逻辑的，我一般不写注释。对于注释的维护的确是有一定的成本。某些时刻一个逻辑复杂的方法有变动，可能会忘记同步修改其对应的注释，就会导致之后的阅读理解有误导倾向。个人觉得类、方法的注释很重要，以前遇到自己不懂的，我都会到某度上去查，发现往往很多搜索结果解释的不够准确，常常会误导我。后面我几乎都是通过阅读官方文档来学习，如通过阅读JDK中Javadoc的文档就能准确的理解某个函数能做什么，该怎么用。

关于注释该用英文还是中文这个问题，个人觉得用中文可能比较好，因为大家母语都是中文，你不能按照自己的标准来想别人也能读懂你写的英文。除非是做国际化项目以及所在团队能接受注释全用英文那自然更好。

2020-01-13 23:10



逍遥思

在 User 类这样一个上下文中，我们没有在成员变量的命名中重复添加“user”这样一个前缀单词，而是直接命名为 name、password、avatarUrl。

但示例代码好像都带了 user 前缀？

2020-01-13 10:21

作者回复

这节课里的代码都没带吧

2020-01-27 18:47



helloworld

关于中英文注释的问题，强烈建议用英文注释！因为每个人的机器上用的编码方式不一样很容易造成乱码的问题。如果英文不好的话，这也正好是锻炼自己的一次机会

2020-02-15 23:48



javaadu

最近在看《编写可读代码的艺术》，里面有两章就是在讲本文的主题，也推荐给大家

2020-01-22 20:43



tang

全是细节，感觉要培养自己的代码意识

2020-01-17 12:40



JM

1. isValidPasword函数里面代码有重复逻辑check if password contains only lowercase characters可以被 check if password contains only a~z,0~9,dot包含，所以可以去掉check lowercase char这个判断

2. 关于英文和中文注释哪个好，取决于你的代码将来要给谁看。国内的开源framework要走向世界，就需要程序员添加英文注释。

2020-01-15 07:27



平风造雨

注释推荐使用英文，只有一个要求，英文水平要过硬。如果英文水平不够，写英文注释没什么收益。

2020-01-14 18:47



龙疯疯

不够看啊，每天很快就看完了。

2020-01-13 10:39



失火的夏天

我司接口类前面有个I，实现类后面也有impl。。。

思考题1 比如判空那个注释就没必要了。我感觉是个人就应该知道是啥意思吧。

思考题2 个人认为，如果英语水平过关的，可以写英文注释，但是英语三脚猫那种，你还是老老实实写中文吧。真人真事，我刚工作那会，做个手机维修得项目，有个后台服务叫做上门服务。然后我那睿智同事就写了个ondoorService，我看到了之后。。。Are you kidding me?

2020-01-13 08:03



密码123456

注释肯定中文啊。毕竟母语。

2020-01-13 08:02



乘坐Tornado的线程魔法师

1. 可以考虑合并一部分简单的条件表达式，并且分拆过于复杂的条件表达式。

2. 注释的语言可以考虑这段代码的维护者的情况，如果中国人外国人都有那么可以考虑使用双语：)

2020-01-13 00:18



aoe

用中文写注释，因为英文读写有障碍

2020-02-16 18:25



好饿早知道送外卖了

我个人习惯在业务相关的判断旁边写上个注释

比如：（审核开关开启 || 课程开关开启）&& 本地记录次数未达上限
在代码review和问题排查的时候的时候 可以提升不少效率

2020-02-13 17:23



好饿早知道送外卖了

关于起名，有一个网站推荐，忘记在哪看到的了

<https://i18ns.com/zh/index.html>

这里收录了很多App的国际化翻译文本

如果是特别业务抽象的名字，我会去github

2020-02-13 16:36



拉欧



命名和注释体现了一个程序员的品味，说几种不好的命名吧：

- 1 英文单词拼写错误，这个特别影响可读性
- 2 函数名不明确，比如get方法返回空值，实际上是在函数内部update了变量
- 3 非驼峰
- 4 拼音式命名

如果原作者在这方面不动脑子，读代码的人就特别费脑子

2020-02-09 21:55