03-面向对象、设计原则、设计模式、编程规范、重构，这五者有何关系？

在上一节课中，我们讲到，要具备编写高质量代码的能力，你需要学习一些编程方法论，其中就包含面向对象（我们可以把它看成一种设计思想）、设计原则、设计模式、编程规范、重构技巧等。而我们整个专栏的内容也是围绕着这几块展开讲解的。所以，今天我就先来简单介绍一下这几个概念，并且说一说它们之间的联系。

今天的内容相当于专栏的一个教学大纲，或者说学习框架。它能让你对整个专栏所涉及的知识点，有一个全局性的了解，能帮你将后面零散的知识更系统地组织在大脑里。

话不多说，我们就一块来看一下，接下来的这8个月我们到底要学习哪些内容吧！

面向对象

现在，主流的编程范式或者是编程风格有三种，它们分别是面向过程、面向对象和函数式编程。面向对象这种编程风格又是这其中最主流的。现在比较流行的编程语言大部分都是面向对象编程语言。大部分项目也都是基于面向对象编程风格开发的。面向对象编程因为其具有丰富的特性（封装、抽象、继承、多态），可以实现很多复杂的设计思路，是很多设计原则、设计模式编码实现的基础。

所以，在专栏的最开始，我们会详细地讲解面向对象编程的相关的知识，为学习后面的内容做铺垫。对于这部分内容，你需要掌握下面这7个大的知识点。

- 面向对象的四大特性：封装、抽象、继承、多态
- 面向对象编程与面向过程编程的区别和联系
- 面向对象分析、面向对象设计、面向对象编程
- 接口和抽象类的区别以及各自的应用场景
- 基于接口而非实现编程的设计思想
- 多用组合少用继承的设计思想
- 面向过程的贫血模型和面向对象的充血模型

设计原则

设计原则是指导我们代码设计的一些经验总结。设计原则这块儿的知识有一个非常大的特点，那就是这些原则听起来都比较抽象，定义描述都比较模糊，不同的人会有不同的解读。所以，如果单纯地去记忆定义，对于编程、设计能力的提高，意义并不大。对于每一种设计原则，我们需要掌握它的设计初衷，能解决哪些编程问题，有哪些应用场景。只有这样，我们才能在项目中灵活恰当地应用这些原则。

对于这一部分内容，你需要透彻理解并且掌握，如何应用下面这样几个常用的设计原则。

- SOLID原则-SRP单一职责原则
- SOLID原则-OCP开闭原则
- SOLID原则-LSP里式替换原则
- SOLID原则-ISP接口隔离原则
- SOLID原则-DIP依赖倒置原则

- DRY原则、KISS原则、YAGNI原则、LOD法则

设计模式

设计模式是针对软件开发中经常遇到的一些设计问题，总结出来的一套解决方案或者设计思路。大部分设计模式要解决的都是代码的可扩展性问题。设计模式相对于设计原则来说，没有那么抽象，而且大部分都不难理解，代码实现也并不复杂。这一块的学习难点是了解它们都能解决哪些问题，掌握典型的应用场景，并且懂得不过度应用。

经典的设计模式有23种。随着编程语言的演进，一些设计模式（比如Singleton）也随之过时，甚至成了反模式，一些则被内置在编程语言中（比如Iterator），另外还有一些新的模式诞生（比如Monostate）。

在专栏中，我们会重点讲解23种经典的设计模式。它们又可以分为三大类：创建型、结构型、行为型。对于这23种设计模式的学习，我们要有侧重点，因为有些模式是比较常用的，有些模式是很少被用到的。对于常用的设计模式，我们要花多点时间理解掌握。对于不常用的设计模式，我们只需要稍微了解即可。

我按照类型和是否常用，对专栏中讲到的这些设计模式，进行了简单的分类，具体如下所示。

1.创建型

常用的有：单例模式、工厂模式（工厂方法和抽象工厂）、建造者模式。

不常用的有：原型模式。

2.结构型

常用的有：代理模式、桥接模式、装饰者模式、适配器模式。

不常用的有：门面模式、组合模式、享元模式。

3.行为型

常用的有：观察者模式、模板模式、策略模式、职责链模式、迭代器模式、状态模式。

不常用的有：访问者模式、备忘录模式、命令模式、解释器模式、中介模式。

编程规范

编程规范主要解决的是代码的可读性问题。编码规范相对于设计原则、设计模式，更加具体、更加偏重代码细节。即便你可能对设计原则不熟悉、对设计模式不了解，但你最起码要掌握基本的编码规范，比如，如何给变量、类、函数命名，如何写代码注释，函数不宜过长、参数不能过多等等。

对于编码规范，考虑到很多书籍已经讲得很好了（比如《重构》《代码大全》《代码整洁之道》等）。而且，每条编码规范都非常简单、非常明确，比较偏向于记忆，你只要照着来做可以。它不像设计原则，需要融入很多个人的理解和思考。所以，在这个专栏中，我并没有花太多的篇幅来讲解所有的编码规范，而是总结了我认为的最能改善代码质量的20条规范。如果你暂时没有时间去看那些经典的书籍，看我这些就够了。

除此之外，专栏并没有将编码规范单独作为一个模块来讲解，而是跟重构放到了一起。之所以这样做，那是

因为我把重构分为大重构和小重构两种类型，而小重构利用的知识基本上就是编码规范。

除了编码规范，我们还会介绍一些代码的坏味道，让你知道什么样的代码是不符合规范的，应该如何优化。参照编码规范，你可以写出可读性好的代码；参照代码的坏味道，你可以找出代码存在的可读性问题。

代码重构

在软件开发中，只要软件在不停地迭代，就没有一劳永逸的设计。随着需求的变化，代码的不停堆砌，原有的设计必定会存在这样那样的问题。针对这些问题，我们就需要进行代码重构。重构是软件开发中非常重要的一个环节。持续重构是保持代码质量不下降的有效手段，能有效避免代码腐化到无可救药的地步。

而重构的工具就是我们前面罗列的那些面向对象设计思想、设计原则、设计模式、编码规范。实际上，设计思想、设计原则、设计模式一个最重要的应用场景就是在重构的时候。我们前面讲过，虽然使用设计模式可以提高代码的可扩展性，但过度不恰当地使用，也会增加代码的复杂度，影响代码的可读性。在开发初期，除非特别必须，我们一定不要过度设计，应用复杂的设计模式。而是当代码出现问题的时候，我们再针对问题，应用原则和模式进行重构。这样就能有效避免前期的过度设计。

对于重构这部分内容，你需要掌握以下几个知识点：

- 重构的目的（why）、对象（what）、时机（when）、方法（how）；
- 保证重构不出错的技术手段：单元测试和代码的可测试性；
- 两种不同规模的重构：大重构（大规模高层次）和小重构（小规模低层次）。

希望你学完这部分内容之后，不仅仅是掌握一些重构技巧、套路，更重要的是建立持续重构意识，把重构当作开发的一部分，融入到日常的开发中。

五者之间的联系

关于面向对象、设计原则、设计模式、编程规范和代码重构，这五者的关系我们前面稍微提到了一些，我这里再总结梳理一下。

- 面向对象编程因为其具有丰富的特性（封装、抽象、继承、多态），可以实现很多复杂的设计思路，是很多设计原则、设计模式等编码实现的基础。
- 设计原则是指导我们代码设计的一些经验总结，对于某些场景下，是否应该应用某种设计模式，具有指导意义。比如，“开闭原则”是很多设计模式（策略、模板等）的指导原则。
- 设计模式是针对软件开发中经常遇到的一些设计问题，总结出来的一套解决方案或者设计思路。应用设计模式的主要目的是提高代码的可扩展性。从抽象程度上来讲，设计原则比设计模式更抽象。设计模式更加具体、更加可执行。
- 编程规范主要解决的是代码的可读性问题。编码规范相对于设计原则、设计模式，更加具体、更加偏重代码细节、更加能落地。持续的小重构依赖的理论基础主要就是编程规范。
- 重构作为保持代码质量不下降的有效手段，利用的就是面向对象、设计原则、设计模式、编码规范这些理论。

实际上，面向对象、设计原则、设计模式、编程规范、代码重构，这五者都是保持或者提高代码质量的方法论，本质上都是服务于编写高质量代码这一件事的。当我们追本逐源，看清这个本质之后，很多事情怎么做

就清楚了，很多选择怎么选也清楚了。比如，在某个场景下，该不该用这个设计模式，那就看能不能提高代码的可扩展性；要不要重构，那就看重代码是否存在可读、可维护问题等。

重点回顾

今天的内容到此就讲完了。我画了一张图，总结了专栏中所涉及的知识点。在学习后面的课程的时候，你可以经常翻出来看一下，建立全局意识，不至于迷失在零碎的知识点中。

课堂讨论

今天课堂讨论的话题有两个。

1. 在今天讲到的内容中，你觉得哪一部分内容对提高代码质量最有效？为什么？除了我罗列的这些内容之外，你还知道哪些可以提高代码质量的方法？
2. 我们知道，最经典的设计模式书籍是GoF的《设计模式》，它的中文全称是《设计模式：可复用面向对象软件的基础》，英文全称是“Design Patterns: Elements of Reusable Object-Oriented Software”。为什么它在标题中会特意提到“面向对象”呢？

欢迎在留言区写下你的想法，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：

- 阿西吧 2019-11-04 18:33:27
  面向对象是武器，设计模式是招式，设计原则是心法
  以心法为基础，以武器运用招式应对复杂的编程问题 [35赞]

- AF 2019-11-04 19:47:26
  第二题，文中已经给出答案了？
  主要是因为面向对象编程因为具有丰富的特性(封装抽象继承多态)，可以实现很多复杂的设计思路，是很多设计原则、设计模式等编码实现的基础。 [9赞]

- 帆大肚子 2019-11-04 19:05:31
  函数是相对比较小的可复用单位
  面向对象把可复用单位提升到类层次
  设计模式把可复用单位提升到框架层次 [8赞]

- lijun 2019-11-04 21:52:31
  有的时候业务复杂，代码就被业务牵着鼻子走，尤其是老项目，既存代码写的可读性差，但是时间紧任务重，能跑就行……哎 [4赞]

- Yayu 2019-11-05 00:39:36
  面向对象的特征也未必包含“继承”这一点吧，比如 Go 语言就没有提供“继承”这个特性，取而代之的是，推荐使用“组合”。但不能说它不支持面向对象编程。那么我们在探讨“面向对象”这个范式时，需要更深刻的去思考“面向对象”的本质是什么。而不是用Java 中的概念来一以概之。希望王争老师参考。 [3赞]

- 丁丁历险记 2019-11-04 17:29:26
  建议和左耳的程序员练级攻略结合起来一起读。
  上述道理，知易行难，一起加油。 [3赞]

- 陈华应 2019-11-05 09:29:53
  1.面向对象，设计原则，内功心法，功力越深厚，设计模式，编程范式应用就越得心应手和自然而然。生搬硬套甚至会适得其反，不但自己觉得别扭，别人也看不懂 [1赞]

- 辣么大 2019-11-05 06:17:32
  GoF的提出是针对解决面向对象编程中遇到的问题，所以书名中强调“面向对象”。It’s a book of design patterns that describes simple and elegant solutions to specific problems in object-oriented software design. [1赞]

- 繁星mind 2019-11-04 23:15:14
  1.面向对象特性是设计原则的基础，设计原则又为面向对象特性使用提供指导。
  2.设计模式是为了解决某些特定问题而产生的方案，我们要了解每种设计模式解决的问题，以及优缺点才能更好的解决问题。
  3.编码规范则是一种约定，让大家以相同的语言来沟通。
  4.重构是对发展过程中，出现的一些问题，进行针对性的改造，换句话说，当前框架和设计不能满足业务发展，就要对设计框架进行改造，解决现在和未来遇到的问题。 [1赞]

- 小伟 2019-11-05 12:43:16
  话题2：
  其实是个怎么理解“面向对象”这四个字的问题。个人理解，“面向”是构建的意思，“面向对象”就是构建对象，而使用什么东西来构建并不是关注的重点。相对的，还有“基于对象”的概念，指的是使用的就对象，关注的也是使用的这些对象，而用这些对象构建出来的东西则不是关注重点。
  回到话题中的问题，《设计模式》关注的是对象的构建过程，所以强调的是“面向对象”，如果关注的是被使用的那些对象，则名字应该改成《设计模式：可复用基于对象软件的基础》，当然，基于对象的模式和面向对象的模式会完全不同。

- 小伟 2019-11-05 11:39:15
  话题1：在说如果提高代码质量之前，要定义清楚什么事代码质量。个人理解，高质量的代码是容易读懂的、代码结构清晰的、逻辑思路缜密的、适度抽象的、书写规范的及测试友好的。那么回头再看如何提高代码质量，就可以从改善这几点入手。除王老师文中讲到的，我觉得写代码前做到逻辑思路缜密也很重要。细细说来，如果业务思路清楚了，那么现有几个业务分支就清楚了，以后扩展时可能的业务分支也清楚了，那么定义几个类和方法、他们之间的关系、用什么模式等就清楚了，说白了就是代码框架就有了，其实也就涵盖了可读性、可扩展性、可复用性和简洁性。之后，就是在每个方法里做CRUD了。所以说，就像算法一样，思路是核心，有了明确的思路，其他的经过练手，自然就水到渠成了。借用国庆举个栗子，思路是DF-41的控制芯片，火箭发动机等助推器是设计模式，分导弹头是业务逻辑，控制芯片决定了助推器的点火和分离，助推器决定了分导弹头能否发挥出最大作用。

- EndayN 2019-11-05 11:34:38
  不过组合模式还挺有用的啊。在那种有递归的场景里啊。

- 胖大海 2019-11-05 11:32:29
  对于重构，我有话（槽）要说（吐）：我是做企业项目外包的，我们给客户交付的项目大多在客户方面庞杂缺少弹性的IT合规性要求和业务快速变动的现实的双重夹击下痛苦地演进着。重构意味着不改变系统已有功能的情况下优化项目代码，但面临着合规性和流程的限制导致这类无业务功能更新的发布不会被甲方IT放行；另一方面不断的新需求的变更又不断地劣化着已有代码的系统架构以及业务代码的设计。不知道有没有面临同样问题的同学，在无法改变现有大环境的前提下进行有限的优化？我们现在能做的也就是在每一次追加的新需求变更时进行小幅度的改善，但这无法受到任何工作流程的保护和管理，仅能靠程序员个人的良心来做。另外，如何保证重构不会因为代码结构的变化引入新的bug呢？单元测试以及覆盖率较高的自动化测试吗？

- 蝴蝶 2019-11-05 10:58:40
  打卡!

- 划时代 2019-11-05 10:03:21
  老师我有疑问，在没有充分必要的情况下是不是不采用面向对象编程（继承+多态），大多数情况下使用基于对象的方式进行编程（封装+组合）？

- 阳光很轻 2019-11-05 09:22:43
  面向对象，设计原则，设计模式，编程规范，代码重构

- WL 2019-11-05 08:28:59
  请问一下老师面向对象编程，面向过程编程，面向对象设计这些词中的“面向”是啥意思？怎样做就算“面向”了呢？

- Yuuuuu 2019-11-05 00:32:00
  在初次编写代码时可读性是最重要的，在维护过程中扩展性是最要的。因为初次编写代码时，我们可能不清楚未来程序的扩展点，而当新需求来时，就给了我们一些提示，我们可以更好的判断是否需要重构，是否需要提供扩展点

- William 2019-11-05 00:09:38
  个人认为编码规范最重要，
  代码是读的时候多，所以可读性必须好.

- 刘冬 2019-11-04 23:59:52
  初级：代码规范 面向对象
  中级：设计原则 设计模式
  高级：设计模式 重构

  oop的基本特性（封装 抽象 继承 多态）和design pattern的需求高度匹配