# 04 | 理论一：当谈论面向对象的时候，我们到底在谈论什么？

2019-11-11 王争

设计模式之美

进入课程 >

**讲述：冯永吉**

时长 14:38　大小 13.41M

考虑到各个水平层次的同学，并且保证专栏内容的系统性、全面性，我会循序渐进地讲解跟设计模式相关的所有内容。所以，专栏正文的第一个模块，我会讲一些设计原则、设计思想，比如，面向对象设计思想、经典设计原则以及重构相关的知识，为之后学习设计模式做铺垫。

在第一个模块中，我们又首先会讲到面向对象相关的理论知识。提到面向对象，我相信很多人都不陌生，随口都可以说出面向对象的四大特性：封装、抽象、继承、多态。实际上，面向对象这个概念包含的内容还不止这些。所以，今天我打算花一节课的时间，先大概跟你聊一下，当我们谈论面向对象的时候，经常会谈到的一些概念和知识点，为学习后面的几节更加细化的内容做一个铺垫。

特别说明一下，对于今天讲到的概念和知识点，大部分我都是点到为止，并没有展开详细讲解。如果你看了之后，对某个概念和知识点还不是很清楚，那也没有关系。在后面的几节课中，我会花更多的篇幅，对今天讲到的每个概念和知识点，结合具体的例子，一一做详细的讲解。

## 什么是面向对象编程和面向对象编程语言？

面向对象编程的英文缩写是 OOP，全称是 Object Oriented Programming。对应地，面向对象编程语言的英文缩写是 OOPL，全称是 Object Oriented Programming Language。

面向对象编程中有两个非常重要、非常基础的概念，那就是类（class）和对象（object）。这两个概念最早出现在 1960 年，在 Simula 这种编程语言中第一次使用。而面向对象编程这个概念第一次被使用是在 Smalltalk 这种编程语言中。Smalltalk 被认为是第一个真正意义上的面向对象编程语言。

1980 年左右，C++ 的出现，带动了面向对象编程的流行，也使得面向对象编程被越来越多的人认可。直到今天，如果不按照严格的定义来说，大部分编程语言都是面向对象编程语言，比如 Java、C++、Go、Python、C#、Ruby、JavaScript、Objective-C、Scala、PHP、Perl 等等。除此之外，大部分程序员在开发项目的时候，都是基于面向对象编程语言进行的面向对象编程。

以上是面向对象编程的大概发展历史。在刚刚的描述中，我着重提到了两个概念，面向对象编程和面向对象编程语言。那究竟什么是面向对象编程？什么语言才算是面向对象编程语言呢？如果非得给出一个定义的话，我觉得可以用下面两句话来概括。

> 面向对象编程是一种编程范式或编程风格。它以类或对象作为组织代码的基本单元，并将封装、抽象、继承、多态四个特性，作为代码设计和实现的基石 。
>
> 面向对象编程语言是支持类或对象的语法机制，并有现成的语法机制，能方便地实现面向对象编程四大特性（封装、抽象、继承、多态）的编程语言。

一般来讲， 面向对象编程都是通过使用面向对象编程语言来进行的，但是，不用面向对象编程语言，我们照样可以进行面向对象编程。反过来讲，即便我们使用面向对象编程语言，

写出来的代码也不一定是面向对象编程风格的，也有可能是面向过程编程风格的。这里听起来是不是有点绕？不过没关系，我们在后面的第 7 节课中，会详细讲解这个问题。

除此之外，从定义中，我们还可以发现，理解面向对象编程及面向对象编程语言两个概念，其中最关键的一点就是理解面向对象编程的四大特性。这四大特性分别是：封装、抽象、继承、多态。不过，关于面向对象编程的特性，也有另外一种说法，那就是只包含三大特性：封装、继承、多态，不包含抽象。为什么会有这种分歧呢？抽象为什么可以排除在面向对象编程特性之外呢？关于这个问题，在下一节课详细讲解这四大特性的时候，我还会再拿出来说一下。不过，话说回来，实际上，我们没必要纠结到底是四大特性还是三大特性，关键还是理解每种特性讲的是什么内容、存在的意义以及能解决什么问题。

而且，在技术圈里，封装、抽象、继承、多态也并不是固定地被叫作“四大特性”（features），也有人称它们为面向对象编程的四大概念（concepts）、四大基石（cornerstones）、四大基础（fundamentals）、四大支柱（pillars）等等。你也发现了吧，叫法挺混乱的。不过，叫什么并不重要。我们只需要知道，这是前人进行面向对象编程过程中总结出来的、能让我们更容易地实现各种设计思路的几个编程套路，这就够了。在之后的课程讲解中，我统一把它们叫作“四大特性”。

## 如何判定某编程语言是否是面向对象编程语言？

如果你足够细心，你可能已经留意到，在我刚刚的讲解中，我提到，“如果不按照严格的定义来说，大部分编程语言都是面向对象编程语言”。为什么要加上“如果不按照严格的定义”这个前提呢？那是因为，如果按照刚刚我们给出的严格的面向对象编程语言的定义，前面提到的有些编程语言，并不是严格意义上的面向对象编程语言，比如 JavaScript，它不支持封装和继承特性，按照严格的定义，它不算是面向对象编程语言，但在某种意义上，它又可以算得上是一种面向对象编程语言。我为什么这么说呢？到底该如何判断一个编程语言是否是面向对象编程语言呢？

还记得我们前面给出的面向对象编程及面向对象编程语言的定义吗？如果忘记了，你可以先翻到上面回顾一下。不过，我必须坦诚告诉你，那个定义是我自己给出的。实际上，对于什么是面向对象编程、什么是面向对象编程语言，并没有一个官方的、统一的定义。而且，从 1960 年，也就是 60 年前面向对象编程诞生开始，这两个概念就在不停地演化，所以，也无法给出一个明确的定义，也没有必要给出一个明确定义。

实际上，面向对象编程从字面上，按照最简单、最原始的方式来理解，就是将对象或类作为代码组织的基本单元，来进行编程的一种编程范式或者编程风格，并不一定需要封装、抽象、继承、多态这四大特性的支持。但是，在进行面向对象编程的过程中，人们不停地总结发现，有了这四大特性，我们就能更容易地实现各种面向对象的代码设计思路。

比如，我们在面向对象编程的过程中，经常会遇到 is-a 这种类关系（比如狗是一种动物），而继承这个特性就能很好地支持这种 is-a 的代码设计思路，并且解决代码复用的问题，所以，继承就成了面向对象编程的四大特性之一。但是随着编程语言的不断迭代、演化，人们发现继承这种特性容易造成层次不清、代码混乱，所以，很多编程语言在设计的时候就开始摒弃继承特性，比如 Go 语言。但是，我们并不能因为它摒弃了继承特性，就一刀切地认为它不是面向对象编程语言了。

实际上，我个人觉得，只要某种编程语言支持类或对象的语法概念，并且以此作为组织代码的基本单元，那就可以被粗略地认为它就是面向对象编程语言了。至于是否有现成的语法机制，完全地支持了面向对象编程的四大特性、是否对四大特性有所取舍和优化，可以不作为判定的标准。基于此，我们才有了前面的说法，**按照严格的定义，很多语言都不能算得上面向对象编程语言，但按照不严格的定义来讲，现在流行的大部分编程语言都是面向对象编程语言。**

所以，多说一句，关于这个问题，我们一定不要过于学院派，非要给面向对象编程、面向对象编程语言下个死定义，非得对某种语言是否是面向对象编程语言争个一清二白，这样做意义不大。

## 什么是面向对象分析和面向对象设计？

前面我们讲了面向对象编程（OOP），实际上，跟面向对象编程经常放到一块儿来讲的还有另外两个概念，那就是面向对象分析（OOA）和面向对象设计（OOD）。面向对象分析英文缩写是 OOA，全称是 Object Oriented Analysis；面向对象设计的英文缩写是 OOD，全称是 Object Oriented Design。OOA、OOD、OOP 三个连在一起就是面向对象分析、设计、编程（实现），正好是面向对象软件开发要经历的三个阶段。

关于什么是面向对象编程，我们前面已经讲过了。我们现在再来讲一下，什么是面向对象分析和设计。这两个概念相对来说要简单一些。面向对象分析与设计中的“分析”和“设计”这两个词，我们完全可以从字面上去理解，不需要过度解读，简单类比软件开发中的需

求分析、系统设计即可。不过，你可能会说，那为啥前面还加了个修饰词“面向对象”呢？有什么特殊的意义吗？

之所以在前面加“面向对象”这几个字，是因为我们是围绕着对象或类来做需求分析和设计的。分析和设计两个阶段最终的产出是类的设计，包括程序被拆解为哪些类，每个类有哪些属性方法，类与类之间如何交互等等。它们比其他的分析和设计更加具体、更加落地、更加贴近编码，更能够顺利地过渡到面向对象编程环节。这也是面向对象分析和设计，与其他分析和设计最大的不同点。

看到这里，你可能会问，那面向对象分析、设计、编程到底都负责做哪些工作呢？简单点讲，面向对象分析就是要搞清楚做什么，面向对象设计就是要搞清楚怎么做，面向对象编程就是将分析和设计的的结果翻译成代码的过程。今天，我们只是简单介绍一下概念，不展开详细讲解。在后面的面向对象实战环节中，我会用两节课的时间，通过一个实际例子，详细讲解如何进行面向对象分析、设计和编程。

## 什么是 UML？我们是否需要 UML？

讲到面向对象分析、设计、编程，我们就不得不提到另外一个概念，那就是 UML（Unified Model Language），统一建模语言。很多讲解面向对象或设计模式的书籍，常用它来画图表达面向对象或设计模式的设计思路。

实际上，UML 是一种非常复杂的东西。它不仅仅包含我们常提到类图，还有用例图、顺序图、活动图、状态图、组件图等。在我看来，即便仅仅使用类图，学习成本也是很高的。就单说类之间的关系，UML 就定义了很多种，比如泛化、实现、关联、聚合、组合、依赖等。

要想完全掌握，并且熟练运用这些类之间的关系，来画 UML 类图，肯定要花很多的学习精力。而且，UML 作为一种沟通工具，即便你能完全按照 UML 规范来画类图，可对于不熟悉的人来说，看懂的成本也还是很高的。

所以，从我的开发经验来说，UML 在互联网公司的项目开发中，用处可能并不大。为了文档化软件设计或者方便讨论软件设计，大部分情况下，我们随手画个不那么规范的草图，能够达意，方便沟通就够了，而完全按照 UML 规范来将草图标准化，所付出的代价是不值得的。

所以，我这里特别说明一下，专栏中的很多类图我并没有完全遵守 UML 的规范标准。为了兼顾图的表达能力和你的学习成本，我对 UML 类图规范做了简化，并配上了详细的文字解释，力图让你一眼就能看懂，而非适得其反，让图加重你的学习成本。毕竟，我们的专栏并不是一个讲方法论的教程，专栏中的所有类图，本质是让你更清晰地理解设计。

## 重点回顾

今天的内容讲完了，我们来一起总结回顾一下，你需要重点掌握的几个概念和知识点。

**1. 什么是面向对象编程?**

面向对象编程是一种编程范式或编程风格。它以类或对象作为组织代码的基本单元，并将封装、抽象、继承、多态四个特性，作为代码设计和实现的基石 。

**2. 什么是面向对象编程语言?**

面向对象编程语言是支持类或对象的语法机制，并有现成的语法机制，能方便地实现面向对象编程四大特性（封装、抽象、继承、多态）的编程语言。

**3. 如何判定一个编程语言是否是面向对象编程语言?**

如果按照严格的的定义，需要有现成的语法支持类、对象、四大特性才能叫作面向对象编程语言。如果放宽要求的话，只要某种编程语言支持类、对象语法机制，那基本上就可以说这种编程语言是面向对象编程语言了，不一定非得要求具有所有的四大特性。

**4. 面向对象编程和面向对象编程语言之间有何关系?**

面向对象编程一般使用面向对象编程语言来进行，但是，不用面向对象编程语言，我们照样可以进行面向对象编程。反过来讲，即便我们使用面向对象编程语言，写出来的代码也不一定是面向对象编程风格的，也有可能是面向过程编程风格的。

**5. 什么是面向对象分析和面向对象设计?**

简单点讲，面向对象分析就是要搞清楚做什么，面向对象设计就是要搞清楚怎么做。两个阶段最终的产出是类的设计，包括程序被拆解为哪些类，每个类有哪些属性方法、类与类之间

如何交互等等。

## 课堂讨论

今天我们要讨论的话题有两个：

1. 在文章中，我讲到 UML 的学习成本很高，沟通成本也不低，不推荐在面向对象分析、设计的过程中使用，对此你有何看法？
2. 有关面向对象的概念和知识点，除了我们今天讲到的，你还能想到其他哪些吗？

欢迎在留言区发表你的观点，积极参与讨论。你也可以把这篇文章分享给你的朋友，邀请他一起学习。



上一篇　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？

## 精选留言 (95)

写留言

**王争** 置顶

2019-11-11

在这篇文章中，“面向对象编程”一词多义，不同的场景、语境下，解释不同。文章中没有点到这一点，我这里稍微补充说明一下：
1. 文章前半部分，面向对象编程指的是一种编程风格或者范式。
2. 文章后半部分，在讲到面向对象分析、设计、编程的时候，面向对象编程是一种行为。

展开

1 31

**王争** 置顶
2019-11-11

UML中定义了类之间的关系：泛化、实现、关联、聚合、组合、依赖，试问下小伙伴们，你们都能搞清楚这几个的区别吗？能否准确的用不同的箭头、图线来画出来吗？即便你能画出来，团队里的小伙伴都能看懂吗？ 不过，关于类之间的关系，我后面会在实战篇中讲到的，但是，我会简化成四种关系，更好理解。

展开

4 9

**观弈道人**
2019-11-11

Uml图可以简化，但不要随意画，会导致专栏有失严谨性。

1 10

**确认过眼神**
2019-11-11

对于uml来说，简单点是可以的，但是对于规范还是要有的。如果不规范，会的人看不习惯，不会的人容易被带入误区。想学的人，画得再难也会去看，不想学的人，画得再简单易懂，也不会去学。

1 4

**唐龙**
2019-11-11

即便我们使用面向对象编程语言，写出来的代码也不一定是面向对象编程风格的，也有可能是面向过程编程风格的。嗯~刚学C++的时候干过这事。

展开

1 4

**辣么大**
2019-11-11

关于uml类图引起了大家的广泛讨论。我同意老师的观点，uml类图还是太复杂了。我给大家一个链接。Uml类图是不用记的。用的时候看一下cheat sheet就行。https://github.co

m/gdhucoder/Algorithms4/blob/master/designpattern/pic/umlcheatsheet.jpg

展开

3

**daniel李**
2019-11-11

当看到老师说uml意义不大的时候我就懵了，还好原来是指不需要按严格标准死磕uml。

我平时在功能开发初期和后期都是用uml把我的想法可视化然后让师兄审核，减少pr被reject机率。而且也容易让别的工程师接手做功能拓展。

...

展开

> 作者回复: 实际上，大厂也未必都在用。比如类图中几种类关系，同学们有几个能准确的用不同的图线画出来呢?

3

**忆水寒**
2019-11-11

UML设计的合理，新开发者也能去快速开发。关键还是看自己开发还是别人开发。作为学习来说，简单的UML能表达意思即可。

展开

3

**香蕉派2号**
2019-11-11

1.同意老师观点，UML是提供一种规范和准则，如果严格的按照规范来做可能过犹不及，在时间成本和规范之间必尽量要做到平衡。
2.除了以上的概念，还想到了低耦合高内聚，模块化，可维护性，可扩展性，可复用性，对象的唯一性，对象的分类（是is-a还是has的关系）等。

展开

3

**黄林晴**
2019-11-11

打卡~
我觉得在工作中，如果完成一个功能需要30分钟，其实25分钟都在思考，25分钟在设计，实际编码时间只需要5分钟，而前面25分钟就是编码设计

**Geek_f9070f**
2019-11-11

对于UML，我觉得同样不要过于“学院派”，过度求其严谨，而忘记使用它的目的是什么，此谓舍本逐末。毕竟它终究只是一个工具，最终能够服务于我们的表达，方便我们的交流即可。是否要简化，当然也要看场景，至少对于学习这门课程而言，并不需要让其过于复杂而提高我们的学习成本。
另外，我特别欣赏老师这种删繁就简、力求简约和高效的风格，或许这也是一种极客精...
展开

2

**编程界的小学生**
2019-11-11

1.我觉得首先uml这东西很牛逼，很有必要去画，但是也需要分场景，比如crud还强行画一个出来那就是浪费时间，比如超级复杂的东西要画，那我觉得就可以简化，多配上文字注释。比如需求一般，不是很复杂也不是很简单的那种也可以好好画一下，必要的地方配上文案描述。uml能帮助我们瞬间理解这个东西到底要做什么，流程是怎样的，画出来不光是现在看还是以后复习看，他都很香！...
展开

2

**帆大肚子**
2019-11-11

我第一次看《设计模式》的时候，里面的uml很是让人头疼，确实学习成本很大，而且我的学习重心不在uml，被迫学习uml真的很痛苦。

2

**方向**
2019-11-11

UML在毕设时候是必须的，什么用例图，时序图，活动图，非得写上去才显得高大上，但一直不得要领，当时也是网上搜相关的模仿着填充进去。始终认为这种图的目的也是为了传达明确的设计意图，遵循最基本的规范能够达到看懂、意图明确的效果就行了。
展开

2

**逍遥思**
2019-11-11

1. 基本认同，除非某个公司内部统一要求如此，才能真正降低沟通成本并摊低学习成本。没有必须使用的场景，就不值得花时间深入

2. 除了老师今天讲的知识点之外，还有消息传递。对象之间需要相互沟通，沟通的途径就是对象之间收发消息。消息内容包括接收消息的对象标识、需要调用的函数的标识以及必要的信息

展开

2

**君君**
2019-11-11

UML 不懂的看不懂 懂得懒得写。。。

展开

2

**卢爱飞**
2019-11-11

我理解的是要因场景而异，但是最终的目的都是降低沟通的成本。
场景1：在大多数人对UML不是很熟练的情况下，如果采用UML来进行沟通，大家在理解上一定会存在Gap，无形之中会提高学习和沟通的成本，在这种情况下，建议不使用UML。举个例子，《实现领域驱动》的作者一开始是使用UML和领域专家沟通，作者认为UML很简单，但是许多领域专家或开发人员并不能很好地理解，最后又出现了ES（事件风暴...

展开

1

**aoe**
2019-11-11

uml确实很复杂，之前看过2本书学习，还是大部分没记住！支持老师简化！看到老师说UML难学，我就放心了。

展开

1

**青青子衿**
2019-11-11

我们是围绕着对象或类来做需求分析和设计的。分析和设计两个阶段最终的产出是类的设计，包括程序被拆解为哪些类，每个类有哪些属性方法，类与类之间如何交互等等。它们比其他的分析和设计更加具体、更加落地、更加贴近编码，更能够顺利地过渡到面向对象编程环节。这也是面向对象分析和设计，与其他分析和设计最大的不同点
曾经面试的时候被问到，领域驱动设计和数据表驱动设计有什么区别，我觉得王老师的...

展开

1

**shniu**

2019-11-11

1.对uml，真正掌握确实有难度，很容易忘记，原因可能是自己并没有真正的理解设计这件事，比如说uml中是用泛化还是实现不是问题的本质，本质是对于一个特定问题，要如何设计才是尽可能最好的；这中间有两层：将自己的想法转成可表达的设计和将可表达的设计让别人也能理解你的意图，而uml就是一种可表达的方式，至于是不是uml并不重要；但是uml有一套规范，而且知名度高，大家多少都有一些了解，所以就成了一种大家相互沟...

展开

1