# 1.1抽象过程

作者：someone756
原文链接：https://ld246.com/article/1440505593510
来源网站：链滴

　所有编程语言都提供抽象机制。可以认为，人们所能够解决的问题的复杂性直接取决于抽象的型和质量。所谓的“类型”是指”所抽象的是什么“。汇编语言是对底层机器的轻微抽象（这种程度抽象可以想象当时的汇编程序员编程有点难）。然后出现C、BASIC、FORTRAN等对汇编语言的抽象虽然它们对汇编语言有了比较高程度的抽象，但是这种程度的抽象依然要求程序员编程时必须要考虑器模型（”解空间“内，问题建模地，例如：计算机）跟实际问题模型（”问题空间“内，问题存在地方，例如：一项业务）之间的关联。建立这种映射是费时，难以编写并维护代价高昂。
　另一种对机器建模的方式就是对待问题建模。早期的编程语言，例如：APL和LISP，都选择考世界的某些铁定视图。（分别对应：”所有问题都是列表“和”所有问题都是算法“）。PROLOG则所有问题都转换为决策链。此外还产生了基于”约束条件编程“的语言和专门通过对”图形符号操作来实现的语言（后者被证明限制性过强）。这些方式对他们要解决的特定类型的问题还是不错的，但如果超过其特定领域就会力不从心。
　面向对象方式通过向程序员提供表示问题空间中的元素的工具而更进了一步。这种表示方式非通用，使得程序员不会受限于任何特定类型的问题。我们将”问题空间“中的元素及其在”解空间“的表示称为对象。（当然，我们还需要一些无法类比为问题空间元素的对象）。这种思想的实质是：序可以通过添加新类型的对象使自身适应与模特特定问题。因此，你在阅读代码的同时，就是在阅读题的表述。相比以前的语言，这是一种更灵活和更强有力的语言抽象。所以，OOP允许根据问题描述题（阅读代码即阅读问题的表述），而不是根据运行解决方案的计算机来描述问题（编程时同时考虑器模型跟问题模型之间的关联）。但他对计算机仍有联系：每个对象看起来都有点向微型计算机——具有状态，还具有操作，用户可以要求对象执行这些操作。如果要对现实中的对象作类比，那么说它有特性和行为似乎是不错的。
　Alan Kay曾经总结了第一个成功的面向对象的语言、同时也是Java所给予的语言之一的Smalltalk的五个基本特性，这些特性表现了一种纯粹的面向对象的程序设计模式：
　1）万物皆对象。将对象视为奇特的变量，它可以存储数据，除此之外你还可以要求它在自身上执行操作。理论上讲，你可以抽取待解决问题的任何概念化的构件（狗、建物、服务等），将其作为程序中的对象。
　2）程序是对象的集合，它们可以发送消息来告知彼此要做的。要想请一个对象，就必须给给该对象发送一条消息。更具体地说，可以把消息想象成对某个特定对象的方法调用请求。
　3）每个对象都有自己的由其它对象构成的存储。换句话说，可以通过建包含现有对象的包的方式来创建新类型的对象。因此，可以在程序中构建复杂的体系，同时将其复性隐藏在对象的简单性之后。
　4）每个对象都有其类型。按照通用的说法，”每个对象都是某个类（class）的一个实例（instance）“，这里”类“就是”类型“的同义词。每个类最重要的区别于其它类特性就是”可以发什么消息给它“。
　5）某一特定类型的所有对象都可以接收同样的消息。这是一句意味深的表述，你在稍后便会看到。因为”圆形“类型的对象同时是”几何类型“的对象，所以一个”圆形对象必定能够接收发送给”几何形“对象的消息。这意味着可以编写与”几何形“并自动处理所有与何形性质相关的事物的代码。这种可替代性（substitutability）是OOP中最强有力的概念之一。
　Booch对对象提出了更加简洁的描述：对象具有状态、行为、标识。这意味着每个对象都可以拥有内部据（它们给出了对象的状态）和方法（它们产生的行为），并且每一个对象又可以唯一地与其它对象分开来，具体地说，就是每个对象在内存中都有一个唯一的地址。