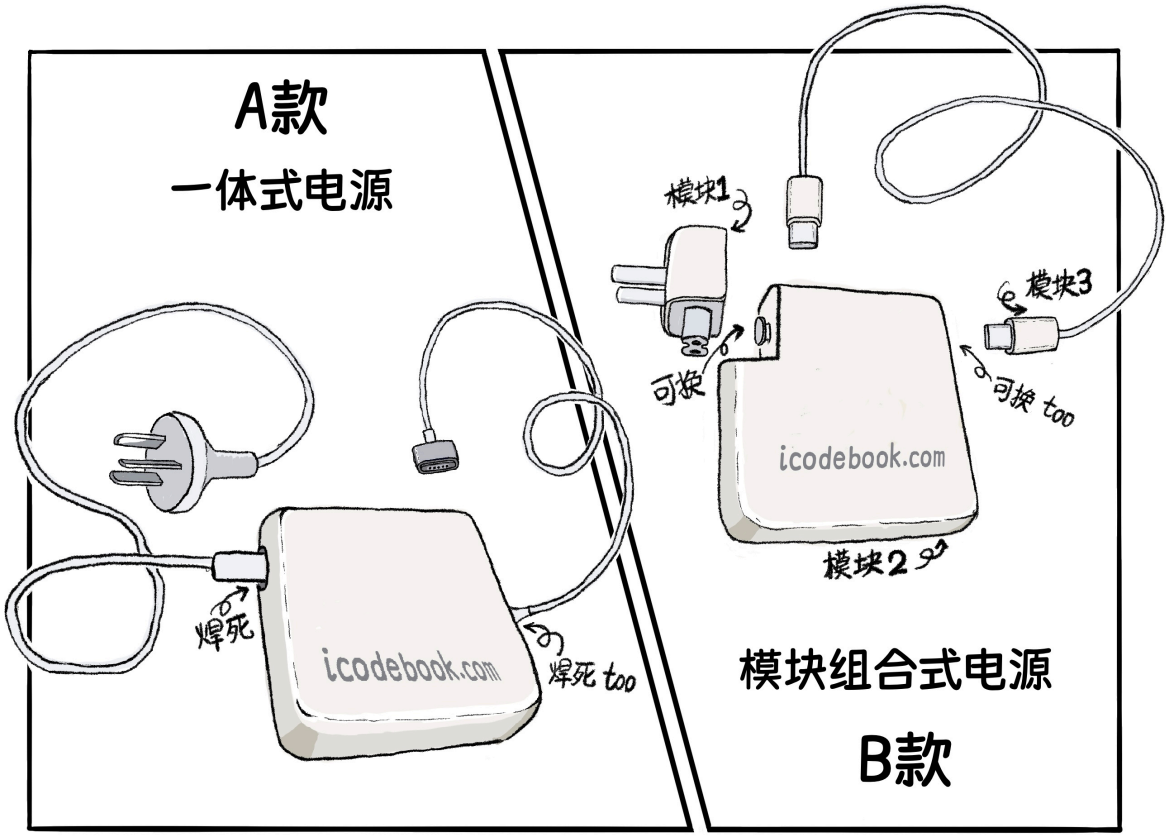
设计模式给我们带来了什么

在我不知道设计模式的时候，也写了一年 Java 代码。代码质量很好，没什么bug，运行稳定。那么设计模式能够带来什么好处呢?

前文说软件设计来自于现实世界，那么我们就看一个现实世界的例子----电脑电源。



A款电源的设计，会导致如果接口发生任何变化，都需要对整个电源进行调整。这就像我们在同一个类的方法中不断地加入代码。我想有一个类似的功能时，我只能copy整段代码再改一份出来。

B款电源的主体是可以复用的，两头的接口可以灵活搭配。这是低耦合、可复用的产品设计。

作为电脑厂商，采用B款电源设计可以令生产更为灵活。三个组件可以分开生产，互不影响。

在经营上，一旦某个地区电脑滞销，要转移到另外的区域销售，厂商只需要重新生产符合当地标准的电源插头即可。如果采用了A款设计，则需要重新生产整个电源。

作为电脑使用者，如果有出国使用的需要，可以购买对应的电源头。而不需要重新购买整个电源（当然也可以购买转接头，这是适配器模式，还是逃不掉设计模式）。如果想把该电源用于其他接口的电脑，也只需要购买对应接口的线材即可。另外线材是易损件，如果坏了，很容易更换。

可复用是这个例子展示的核心思想。可复用带来了灵活性。**所谓的灵活性，本质上就是修改成本更低。**

一切都是为了应对变化

一体电源显然更容易生产，因为不用设计和制作复杂的接口。由于没有接口，使用的材料更少，成本也更低。

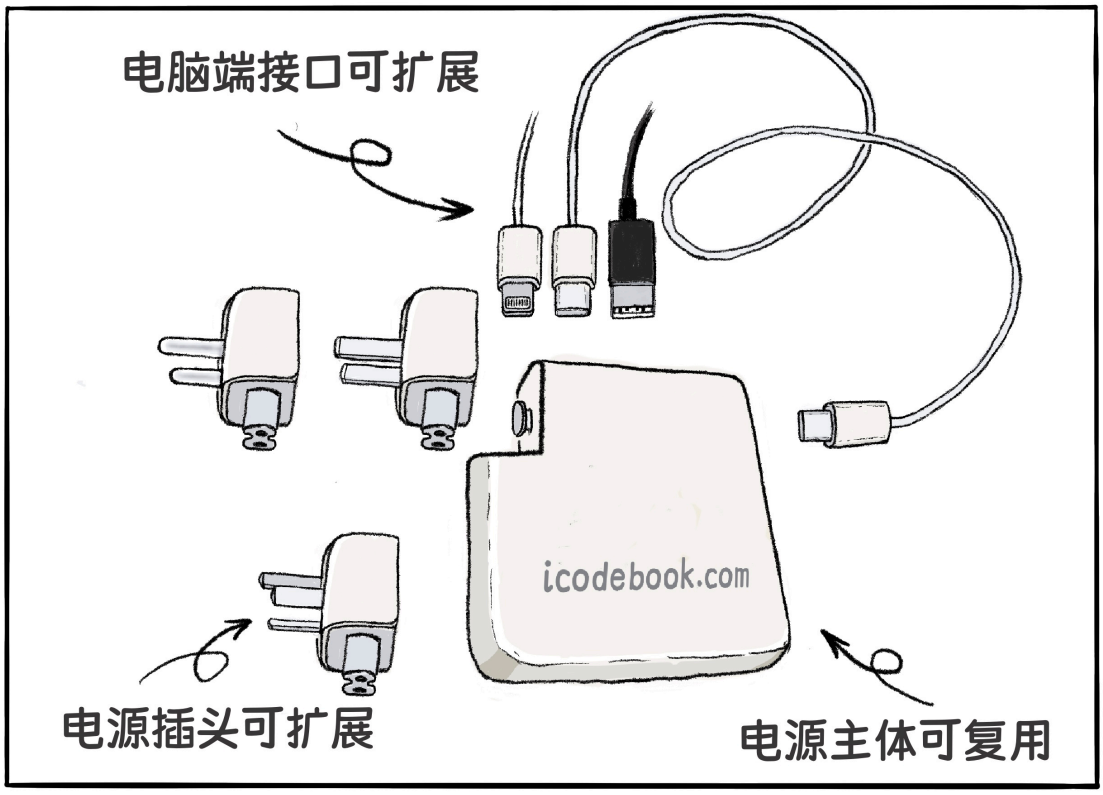
没错，如果电源仅用于此台电脑，并且仅在国内使用，一体电源是最佳的设计。但是一旦变化产生，一体电源的劣势便尽显无遗。

组合式的电源虽然设计和生产都变得麻烦一些，但是却能够灵活应对各种变化。在科技飞速发展的今天，新的接口标准迭代的更快。另外全球一体化也在加速，出国旅游、工作的机会越来越多。组合式电源也就更加符合现状。

说到变化，程序员最怕的就是需求又变了！虽然软件项目管理在尽量控制需求变更，但不可否认需求变化依然频繁发生。为了应对频繁的变化，我们需要可复用的软件设计。

设计模式带来的好处

设计模式带来的好处其实就是优秀程序设计的特征。



复用

组合式电源的主体是可以复用的。当有了新的接口标准，厂商不需要重新设计和生产电源主体，只需要设计电脑侧的线材接口即可。软件也是一样的，很多需求都是在原有基础上增强或者修改。主体功能并没有改变。那么我们做扩展的时候，就可以复用已有组件。

可扩展

我们来看一体电源，**一体**二字恰恰体现了它的紧耦合。两端插头、电源主体，三个主要部分耦合在一起。导致无法进行扩展，电源接口变了就无法使用了。而组合式电源可以找到变化点，只进行相关功能的扩展。换一个组件继续使用。

易于维护

组合式电源我们能一眼看出设计的层次。三种不同组件已经明示了设计层次、模块构成以及模块间的连接方式。而一体式电源没有层次划分，我们只有拆开才能看到内部构造，导致所有的组件看起来都在同一层次。

在排查问题的时候，组合式电源可以很快定位到是哪个组件出了问题。而对于一体式电源，定位是哪头接口有问题都很困难。

设计模式可能带来的问题

复杂度高

设计模式引入更多的类和接口，必然带来更高的复杂度。这是为了获得复用、扩展必须要付出的代价。因此并不是任何问题都需要引入设计模式来解决。你需要考虑程序的复杂度和未来变化的可能性。如果都很低，那么完全没有必要使用设计模式。

我们很少看台灯使用组合式电源设计，因为很少有带台灯出国使用的场景。

代码可读性下降

使用了设计模式的代码，也需要阅读者有一定的设计模式功底。一镜到底的代码当然读起来更加顺畅。就像小说，平铺直叙最好理解，不过加上倒叙和插叙会更有意思。但如果过多使用，则会让读者读不懂。

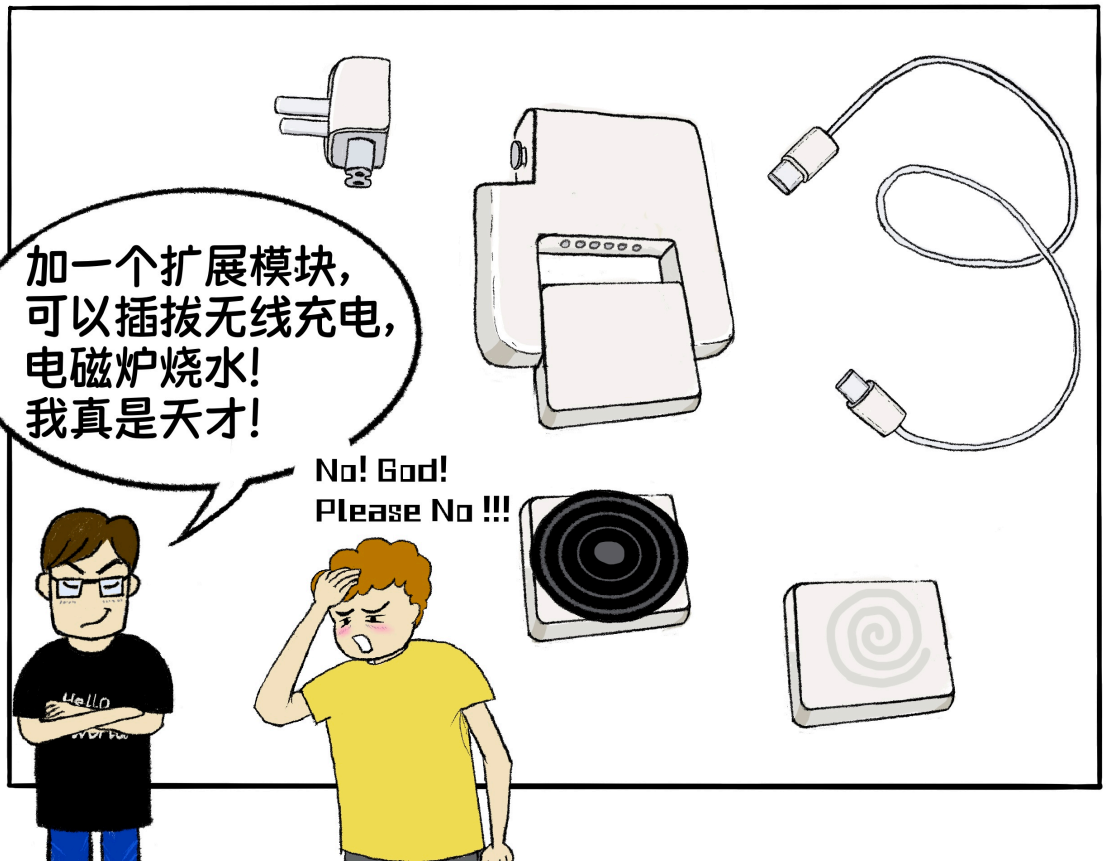
软件开发也是一样的，我们要平衡代码可读性和其他要素的权重。好在23种设计模式已经成为行业的默认标准，我们不用过分担心这个问题。而且我们可以通过合理的命名，让阅读者很容易知道所使用的设计模式。

好的设计实践

软件中遍布设计模式一定不是优秀的设计。这是对设计模式的生搬硬套，甚至有炫技的嫌疑。考虑当前需要，和未来扩展需要，再做出决定。而不是仅仅符合某个设计模式适用场景就拿来使用。

我们思考一下，一体式电源是一个不好的设计吗？一体式电源是符合它所处时代的需求，设计者平衡了未来扩展的可能性和开发的成本，放在当时来看并不能说是不好的设计。更多的设计意味着更多的成本。

开发人员要时刻警醒自己，不要过度设计！


加一个扩展模块，
可以插拔无线充电，
电磁炉烧水!
我真是天才!
No! God!
Please No !!!