第 15 章 先写注释

（把注释变成设计过程的一部分）

许多开发人员会将编写文档推迟到开发过程的最后阶段，即在编码和单元测试完成之后。这是产生低质量文档的最可靠方法之一。编写注释的最佳时间是在过程开始时。首先编写注释使文档成为设计过程的一部分。这不仅可以产生更好的文档，还可以产生更好的设计，并使编写文档的过程更加愉快。

15.1 迟到的注释不是好注释

我见过的几乎每个开发人员都会推迟编写注释。当被问及为什么不更早编写文档时，他们说代码仍在更改。他们说，如果他们尽早编写文档，则必须在代码更改时重新编写文档，所以最好等到代码稳定下来。但是，我怀疑还有另一个原因，那就是他们将文档视为苦差事。因此，他们尽可能地推迟了。

不幸的是，这种方法有几个负面影响。首先，延迟文档通常意味着根本无法完成文档的编写。一旦开始延迟，就很容易再延迟一些，毕竟代码将在几周后变得更加稳定。到了代码毫无疑问地稳定下来的时候，代码已经很多了，这意味着编写文档的任务变得越来越庞大，甚至没有了吸引力。从来没有一个合适的时间可以停下来几天并填写所有缺失的注释，并且很容易以对项目最好的事情是继续迭代、修复缺陷或编写下一个新功能为理由而对其进行合理化。这将导致更多没有注释的代码。

即使您有足够的自制力回去写注释（但说实话：您可能并没有），注释也不会很好。此时，你已经心不在焉了。在你的脑海中，这段代码已经完成了，你急于开始下一个项目。你知道写注释是正确的事情，但它没有乐趣。你只想尽快地处理掉它。因此，您快速地浏览代码，添加足够的注释以使其看起来差不多就行。到目前为止，距离您设计代码已经有一段时间了，所以您对设计过程的记忆变得模糊了。您一边查看代码一边完成注释，因此注释很容易就重复了代码。即使您试图重新构建代码中不明显的设计思想，也会有您不记得的事情。因此，这些注释遗漏了它们应该描述的一些最重要的事情。

15.2 先写注释

我使用一种不同的方法来编写注释，在最开始时就写：

- 对于新类，我首先编写类接口注释。
- 接下来，我为最重要的公有方法编写接口注释和签名，但将方法主体保留为空。
- 我对这些注释进行了迭代，直到基本结构感觉正确为止。
- 此时我为类中最重要的类实例变量编写了声明和注释。
- 最后，我填写方法的主体，并根据需要添加实现注释。
- 在编写方法主体时，我通常会发现需要更多的方法和实例变量。对于每个新方法，我在方法主体之前编写接口注释。对于每个变量，我在编写其声明的同时填写了注释。

当代码完成时，注释也已经完成。从来没有积压的待编写注释。

注释优先的方法具有三个好处。首先，它会产生更好的注释。如果您在设计类时写注释，那么关键的设计问题将在您的脑海中浮现，因此很容易记录下来。最好在编写每个方法的主体之前编写接口注释，这样您就可以专注于方法的抽象和接口，而不会因其实现而分心。在编码和测试过程中，您会注意到并修复注释中的问题。结果，注释在开发过程中也得到了持续的改善。

15.3 注释是一种设计工具

在开始时编写注释的第二个也是最重要的好处是可以改善系统设计。注释提供了完全捕获抽象的唯一方法，好的抽象是好的系统设计的基础。如果您在一开始就缩写了描述抽象的注释，就可以在编写实现代码之前对其进行检查和调整。要写一个好的注释，您必须确定一个变量或一段代码的本质：这件事最重要的方面是什么？在设计过程的早期进行此操作很重要，否则，您只就是个编代码的。

注释就像复杂性煤矿中的金丝雀。如果方法或变量需要较长的注释，那它是一个危险信号，表明您没有很好的抽象。请记住，在第 4 章中提到类应该是深的：最好的类具有非常简单的接口，但可以实现强大的功能。判断接口复杂性的最佳方法是查看描述接口的注释。如果某个方法的接口注释提供了使用该方法所需的所有信息，并且既简短又简单，则表明该方法具有简单的接口。相反，如果没有冗长而复杂的注释就无法完全描述一个方法，则该方法具有复杂的接口。您可以将方法的接口注释与实现进行比较，以了解该方法有多深：如果接口注释必须描述实现的所有主要特性，则该方法是浅的。同样的想法也适用于变量：如果要花很长的注释来完整描述一个变量，那是一个危险信号，表明您可能没有进行正确的变量分解。总体而言，编写注释的行为使您可以及早评估设计决策，以便发现并解决问题。

 危险信号：难以描述 

> 描述方法或变量的注释应该简单而完整。如果您发现很难写这样的注释，则表明您对所描述事物的设计可能存在问题。

当然，仅在注释是完整而清晰的情况下，它们才是复杂性良好的一个指标。如果编写的方法接口注释未提供调用该方法所需的全部信息，或者编写的注释太过晦涩难懂，那么该注释也不能很好地衡量该方法的深度。

15.4 先写注释很有趣

尽早编写注释的第三个也是最后一个好处是，它使编写注释更加有趣。对我来说，编程中最有趣的部分之一是新类的早期设计阶段，我在这个阶段充实类的抽象和结构。我的大部分注释都是在此阶段编写的，这些注释是我记录和测试设计决策质量的方式。我将寻找可以用最少的词来完整而清晰地表达的设计。注释越简单，我对设计的感觉就越好，因此找到简单的注释也会让人有自豪感。如果您采取的是战略式的编程方式，您的主要目标是一个出色的设计，而不仅仅是编写能工作的代码，那么编写注释应该很有趣，因为这是您确定最佳设计的方法。

15.5 先写注释是否很昂贵?

现在，让我们重新审视推迟注释的论点，它避免了在代码演变时重新处理注释的成本。一个简单的粗略计算会表明这并没有节省多少。首先，估算您编写代码和注释所花费的开发时间的总和，包括修改代码和注释的时间；这不太可能超过所有开发时间的 10%。即使您的全部代码行中有一半是注释，编写注释也可能不会占开发总时间的 5% 以上。将注释推迟到最后只会节省其中的一小部分，这个时间并不多。

而如果先写注释将意味着在开始编写代码之前，抽象将更加稳定，这很可能会节省编码时间。相反，如果您首先编写代码，则抽象可能会随代码的发展而变化，与先写注释的方法相比，将需要更多的代码修改。当您考虑所有这些因素时，先写注释可能总体上还更快。

15.6 结论

如果您从未试过先写注释，请尝试一下。坚持足够长的时间来习惯它。然后思考它如何影响了您的注释质量、设计质量以及软件开发的整体乐趣。在尝试了一段时间之后，让我知道您的经历是否与我的相符，以及为什么或者为什么不是这样。