这个原则有两个特征：对于扩展是开放的（open for extension）；对于更改是封闭的（closed for modification）。其目的是：

在面对需求的改变时，软件设计应保持相对稳定，从而使系统可以在第一个版本以后不断推出新的版本。即面对需求，对程序的改动是通过增加新代码进行的，而不是更改现有的代码，这就是“开放-封闭原则”的精神所在。

无论模块是多么的“封闭”，都会存在一些无法对之封闭的变化。既然不可能完全封闭，设计人员必须对于他设计的模块应该对哪种变化封闭做出选择。他必须先猜测出最有可能发生的变化种类，然后构造抽象来隔离那些变化。

在变化发生时立即采取行动：在最初编写代码时，假设变化不会发生；当变化发生时，创建抽象来隔离以后发生的同类变化。

我们希望的是在开发工作展开不久就知道可能发生的变化。查明可能发生的变化所等待的时间越长，要创建正确的抽象就越困难。

开放一封闭原则是面向对象设计的核心所在，遵循这个原则可以带来面向对象技术所声称的巨大好处，即可维护、可扩展、可复用、灵活性好。开发人员应该仅对程序中呈现出频繁变化的那些部分做出抽象，然而，对于应用程序中的每个部分都刻意地进行抽象同样不是一个好主意。拒绝不成熟的抽象和抽象本身一样重要。