第四章 开闭原则

在软件开发中，为了提高软件系统的可维护性和可复用性，增加软件的可扩展性和灵活性，程序员要尽量根据 7 条原则来开发程序，从而提高软件开发效率、节约软件开发成本和维护成本，它们分别是**开闭原则、里氏替换原则、依赖倒置原则、单一职责原则、接口隔离原则、迪米特法则、合成复用原则。**我们将在下面的几节中依次来介绍这 7 条原则，本节首先介绍开闭原则。

1 开闭原则的定义

开闭原则（Open Closed Principle，OCP）由勃兰特·梅耶（Bertrand Meyer）提出，他在 1988 年的著作《面向对象软件构造》（Object Oriented Software Construction）中提出：软件实体应当对扩展开放，对修改关闭（Software entities should be open for extension，but closed for modification），这就是开闭原则的经典定义。

这里的软件实体包括以下几个部分：

1. 项目中划分出的模块
2. 类与接口
3. 方法

开闭原则的含义是：当应用的需求改变时，在不修改软件实体的源代码或者二进制代码的前提下，可以扩展模块的功能，使其满足新的需求。

2 开闭原则的作用

开闭原则是面向对象程序设计的终极目标，它使软件实体拥有一定的适应性和灵活性的同时具备稳定性和延续性。具体来说，其作用如下。

2 .1 对软件测试的影响

软件遵守开闭原则的话，软件测试时只需要对扩展的代码进行测试就可以了，因为原有的测试代码仍然能够正常运行。

2 .2 可以提高代码的可复用性

粒度越小，被复用的可能性就越大；在面向对象的程序设计中，根据原子和抽象编程可以提高代码的可复用性。

2 .3 可以提高软件的可维护性

遵守开闭原则的软件，其稳定性高和延续性强，从而易于扩展和维护。

3 开闭原则的实现方法

可以通过“抽象约束、封装变化”来实现开闭原则，即通过接口或者抽象类为软件实体定义一个相对稳定的抽象层，而将相同的可变因素封装在相同的具体实现类中。

因为抽象灵活性好，适应性广，只要抽象的合理，可以基本保持软件架构的稳定。而软件中易变的细节可以从抽象派生来的实现类来进行扩展，当软件需要发生变化时，只需要根据需求重新派生一个实现类来扩展就可以了。

下面以 Windows 的桌面主题为例介绍开闭原则的应用。

【例】Windows 的桌面主题设计。

分析：Windows 的主题是桌面背景图片、窗口颜色和声音等元素的组合。用户可以根据自己的喜爱更换自己的桌面主题，也可以从网上下载新的主题。这些主题有共同的特点，可以为其定义一个接口（AbstractSubject），而每个具体的主题（SpecificSubject）是其实现类。用户

窗体可以根据需要选择或者增加新的主题，而不需要修改原来的代码，所以它是满足开闭原则的，其类图如图 1 所示。

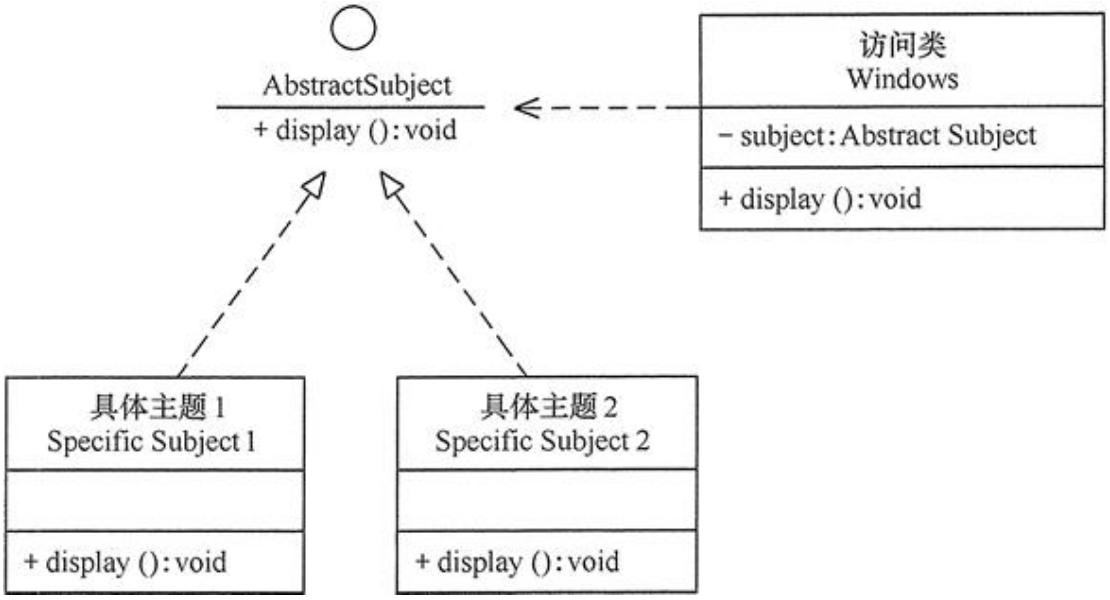

图 1 Windows 的桌面主题类图