第四章 类与接口

2018年7月11日

17:26

**第13条： 使用类和成员的可访问性最小化**

信息隐藏或封装，尽可能地使每个类或者成员不被外界访问。

* 私有类
* 暴击私有的
* 受保护
* 公有

实例域绝不能是共有的。

类具有共有的静态final数组域，或者返回这种域的访问方法，这几乎是总是错误的。

**第14条： 在公有类中使用访问方法而非公有域**

共有类永远都不应该暴露可变的域

如果类是包级私有的，或者是私有的嵌套类，直接暴露它的数据域并没有本质的错误。

**第15条：使可变性最小化**

不可变类只是其实例不能被修改的类。

为了使类变成不可变，要遵循下面五条规则：

1. 不要提供任何修改对象状态的方法。
2. 保证类不会被扩展。
3. 使所有的域都是fianl。
4. 使所有的域都成为私有的。
5. 确保对于任何可变组件的互斥访问。

不可变对象本质上是线程安全的，它们不要求同步。

不仅可以共享不可变对象，甚至也可以共享它们的内部信息。

不可变类真正唯一的缺点是，对于每个不同的值都需要一个单独的对象。

除非有令人信服的理由要使域变成是final的，否则要使每个域都是final。

**第16条： 复合优于继承**

与方法调用不同的是，继承打破了封装性。

为了避免继承的脆弱性，可以使用**复合**和**转发**机制来代替继承，尤其是当存在适合的接口可以实现包装类时。

包装类不仅比子类更加健壮，而且功能更加强大。

**第17条： 要么为继承而设计，并提供文档说明，要么禁止继承。**

每个专门用于继承的类必须有文档说明它可覆盖的方法的自用性。

对于每个为了继承设计的类，唯一的测试方法就是编写子类。

无论是clone还是readObject,都不可以调用可覆盖的方法。

为了继承而设计类，对这个类会有一些实质性的限制。比如抽象类，包括接口的**骨架实现**。

**第18条： 接口优于抽象类**

**现有的类可以很容易被更新，以实现新的接口。**

**接口是定义mixin（混合 类型）的理想选择。**

**接口允许我们构造非层次接口的类型框架。**

通过对你导出的每个重要接口都提供一个抽象的骨架实现类，把接口和抽象类的优点结合起来。

**第19条：接口只用于定义类型**

常量接口模式是对接口的不良使用。

导出常量可以使用枚举类型或者工具类。

工具类可以使用静态导入 static import常量或者方法。

接口应该只被用来定义类型，不应该用来导出常量。

**第20条：类层次优于标签类**

子类化能够表示多种风格对象的单个数据类型。

**第21条：用函数对象表示策略**

Java通过函数对象来实现策略模式。

具体策略只使用一次时，通常使用匿名类来声明和实例化这个具体策略类。

设计重复使用的具体策略时，它的类通常就要被实现为私有的静态成员类，并通过公有的静态final域被导出，其类型为该策略接口。

**第22条：优先考虑静态成员类**

嵌套类有四种：静态成员类，非静态成员类，匿名类，局部类。

如果需要单个方法外可见或者太长不适合在方法内部，使用成员类。

如果成员类的每个实例都需要一个指向其外围实例的引用，就要把成员类做成非静态的，否则做成静态。

在方法内，如果需要创建实例，并且预置的类型说明了这个类型的特征，就把它做成匿名类，否则局部类。