里氏替换原则

2020年2月24日 14:36

1. 定义

1. 如果对每一个类型为 T1的对象 o1，都有类型为 T2 的对象o2，使得以 T1定义的所有程序 P 在所有的对象 o1 都代换成 o2 时，程序 P 的行为没有发生变化，那么类型 T2 是类型 T1 的子类型。
2. 所有引用基类的地方必须能透明地使用其子类的对象。

2. 问题由来

有一功能P1，由类A完成。现需要将功能P1进行扩展，扩展后的功能为P，其中P由原有功能P1与新功能P2组成。新功能P2由类A的子类B来完成，则子类B在完成新功能P2的同时，有可能会导致原有功能P1发生故障。

3. 解决方案

当使用继承时，遵循里氏替换原则。类B继承类A时，除添加新的方法完成新增功能P2外，**尽量不要重写父类A的方法，也尽量不要重载父类A的方法。**

继承包含这样一层含义：父类中凡是已经实现好的方法（相对于抽象方法而言），实际上是在设定一系列的规范和契约，虽然它不强制要求所有的子类必须遵从这些契约，但是如果**子类对这些非抽象方法任意修改，就会对整个继承体系造成破坏。**而里氏替换原则就是表达了这一层含义。

里氏替换原则通俗的来讲就是：子类可以扩展父类的功能，但不能改变父类原有的功能。它包含以下4层含义：

- 子类可以实现父类的抽象方法，但不能覆盖父类的非抽象方法。
- 子类中可以增加自己特有的方法。
- 当子类的方法重载父类的方法时，方法的前置条件（即方法的形参）要比父类方法的输入参数更宽松。比如父类的输入参数为HashMap，则子类应加更为宽松，为Map。
- 当子类的方法实现父类的抽象方法时（重写/重载/实现抽象方法），方法的后置条件（即方法的返回值）要比父类更严格。比如父类的输入参数为Map，则子类应加更为严格，为Map或HashMap。