1 里氏代换原则 (LSP)

Liskov Substitution Principle

1988 年, 由麻省理工的 Barbara Liskov 提出, 其思想表达为: 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 的行为没有变化, 那么类型 T2 是类型 T1 的子类型.

里氏代换原则是继承复用的基石. 只有当派生类可以替换掉基类, 软件功能不会受到影响, 基类才能真正被复用, 而派生类也才能够在基类的基础上增加新的功能.

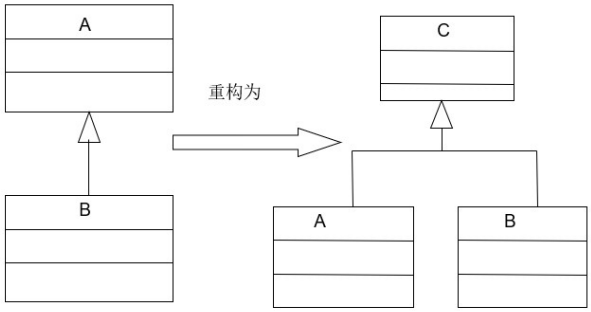
JAVA 语言对里氏代换原则的支持: 在编译时期,JAVA 语言会检查一个程序是否符合里氏代换原则. 里氏代换原则要求凡是使用基类的地方, 子类型一定适用, 因此子类必须具备基本类型的全部接口.

Java 语言对里氏代换支持的局限:Java 编译器的检查是有局限的,JAVA 编译器不能检查一个系统在实现和业务逻辑上是否满足里氏代换原则.

1.1 从代码重构的角度理解

里氏代换原则讲的是基类与子类的关系. 只有当这种关系存在时, 里氏代换关系才能存在, 反之不存在. 如果有两个具体类 A 和类 B 之间的关系违反了里氏代换原则的设计, 根据具体情况可以在下面的两种重构方案中选择一个:

创建新的抽象类 C, 作为两个类的基类:



下面介绍一个判断正方形是否是长方形的问题. 一个长方形（Rectangle）类和正方形（Square）类的代码如下:

```
public class Rectangle{
...
  void setWidth(int width){
      this.width=width;
   }
  void setHeight(int height){
      this.height=height
   }
}
```

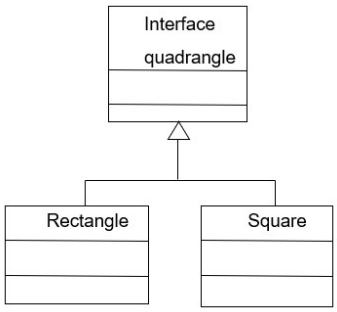
```
public class Square{
...
```

```
3   void setWidth(int width){
4       this.width=width;
5       this. height=width;
6   }
7   void setHeight(int height){
8       this.setWidth(height);
9   }
10  }
```

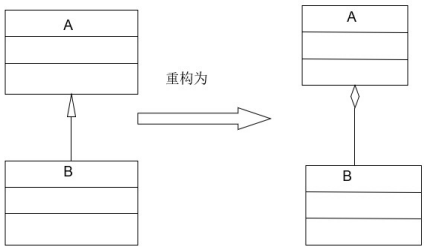
假设有如下函数, 需要对长方形对象的边长进行改动:

```
1  public void resize(Rectangle r){
2    while(r.getHeight()<=r.getWidth()){
3      r.setHeight(r.getWidth()+1);
4    }
5  }
```

如果将正方形作为长方形的子类, 这样, 只要 width 或 height 被赋值, 那么 width 和 height 会同时赋值, 从而长方形的长和宽总是一样的. 换言之, 里氏代换原则就被破坏了. Rectangle 和 Square 都应该同属于四边形（Quadrangle）类的子类, 如下图所示:



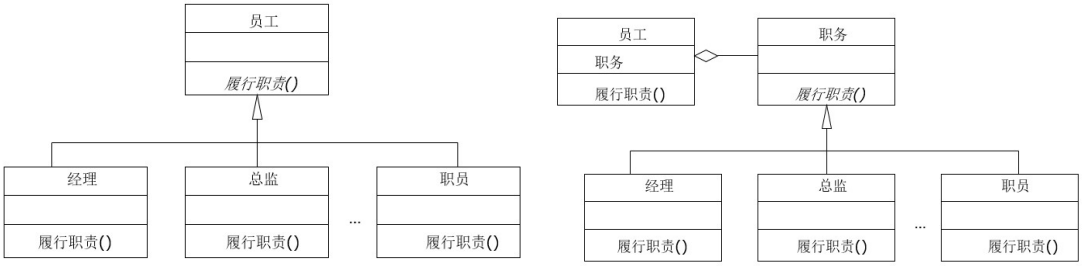
从 B 到 A 的继承关系改为委派 delegate 关系, 如下图:




```
1  class A {
2    void method1();
3    void method2() {
4      b.method2();
5    }
6    B b;
```

```
}
class B {
  void method2();
}
```

从“员工”类派生出“经理”、“总监”、“组长”、“职员”等:



将不恰当的继承关系改为组合/聚合关系。将“职务”作为“员工”的一个成员: