LSP（圆&椭圆）

2020年3月4日 1:16

我们通过之前的学习可以了解到LSP是所有引用基类（父类）的地方必须能透明地使用其子类的对象。

LSP的特征为：

- 所以使用基类代码的地方，用派生类代码替换后，能够正确的执行动作处理。
- 换句话说，如果派生类替换了基类后，不能够正确执行动作，那么他们的继承关系就应该废除。

我们通过以下来说明圆如果是椭圆的子类，会违反里氏代换原则

```
//椭圆
public class Oval{
  //半轴a b
  private double a;
  private double b;

  public double get_a(){
    return a;
  }
  public double get_b(){
    return b;
  }
  public void set_a(double a){
    this.a = a;
  }
  public void set_b(double b){
    this.b = b;
  }
}

//圆
class Circle extends Oval{
  public void set_a(double a){
    super.set_a(a);
    super.set_b(b);
  }
  public void set_b(double b){
    super.set_a(b);
    super.set_b(b);
```

```
    }
}
```

这里Ovel是基类，Circle从Ovel继承。

假如已有的系统中存在以下既有的求面积的业务逻辑代码：

```
double Area(Oval r){

  double area =  3.14 * r.get_a() * r.get_b();
  if( 37.68 ! = area ){
       throw new RuntimeException;
   }
  return area;
};
```

则对应于扩展类Circle，在调用既有业务逻辑时：

```
Oval circle = newCircle();
area(circle);
```

会抛出一个RuntimeException异常。这显然违反了LSP原则

LSP体现了：

类的继承原则：如果一个继承类的对象可能会在基类出现的地方出现运行错误，则该子类不应该从该基类继承，或者说，应该重新设计它们之间的关系。

动作正确性保证：从另一个侧面上保证了符合LSP设计原则的类的扩展不会给已有的系统引入新的错误。

综上分析可说明：圆如果是椭圆的子类，会违反里氏代换原则