Liskov Substitution Principle

Daum Corp.
백명석

Liskov Substitution Principle

```
public class LSP {
    static P P = new P();

    static class T {
        public void doSometing() {
            System.out.println("T#doSometing called");
        }
    }

    static class S extends T {
        public void doSometing() {
            System.out.println("S#doSometing called");
        }
    }

    static class P {
        public void doSomething(T p) {
            p.doSometing();
        }
    }

    public static void main(String[] args) {
        T p = new T();
        S c = new S();

        P.doSomething(p);
        P.doSomething(c);
    }
}
```

5. S는 T의 서브타입이다.

3. T 타입을 사용하는 모든 프로그램 P에서

1. T 타입의 객체 p

2. S 타입의 객체 c

4. c가 p를 대체해도 P에는 어떤 변경도 없다면

OCP vs LSP

- OCP
  - abstraction, polymorphism(inheritance)를 이용해서 구현
- LSP
  - OCP를 받쳐주는 polymorphism에 관한 원칙을 제공
  - LSP가 위반되면 OCP도 위반됨
  - LSP를 위반하면 subtype이 추가될때마다 클라이언트들이 수정되어야 함
  - instanceof/downcasting을 사용하는 것은 전형적인 LSP 위반의 징조

Rectangle 예제

```java
public class Rectangle {
    private int width;
    private int height;

    public int area() {
        return width * height;
    }

    public void setWidth(int width) {
        this.width = width;
    }

    public void setHeight(int height) {
        this.height = height;
    }
}
```

```java
public class RectangleTest {

    private final int width = 5;
    private final int height = 3;

    @Test
    public void
    should_return_area_by_multiplying_width_and_height() {
        // Given
        Rectangle rectangle = createRectangle(new Rectangle());

        // When
        int area = rectangle.area();

        // Then
        assertThat(area, is(width * height));
    }

    private Rectangle createRectangle(Rectangle rectangle) {
        rectangle.setWidth(width);
        rectangle.setHeight(height);
        return rectangle;
    }
}
```

Rectangle 예제

- Rectangle은 시스템의 여러곳에 퍼져있다.
- 정사각형(Square)을 서브 타입으로 추가하려고 한다.
- Square IS-A Rectangle

```
public class Square extends Rectangle {
    @Override
    public void setWidth(int width) {
        super.setHeight(width);
        super.setWidth(width);
    }

    @Override
    public void setHeight(int height) {
        super.setHeight(height);
        super.setWidth(height);
    }
}
```

Rectangle 예제

- failed RectangleTest

```
@Test
public void
should_return_area_using_width_and_height() {
    Rectangle rectangle = buildRectangle(new Rectangle());
    assertArea(rectangle);

    rectangle = buildRectangle(new Square());
    assertArea(rectangle);
}

private void assertArea(Rectangle rectangle) {
    assertThat(rectangle.area(), is(width * height));
}

private Rectangle buildRectangle(Rectangle rectangle)
    rectangle.setWidth(width);
    rectangle.setHeight(height);
    return rectangle;
}
```

```
@Test
public void
should_return_area_using_width_and_height() {
    Rectangle rectangle = buildRectangle(new Rectangle());
    assertArea(rectangle);

    rectangle = buildRectangle(new Square());
    assertArea(rectangle);
}

private void assertArea(Rectangle rectangle) {
    if(rectangle instanceof Square)
        assertThat(rectangle.area(), is(height * height));
    else
        assertThat(rectangle.area(), is(width * height));
}

private Rectangle buildRectangle(Rectangle rectangle) {
    rectangle.setWidth(width);
    rectangle.setHeight(height);
    return rectangle;
}
```

The Representative Rule

- 대리인은 자신이 대리하는 어떤 것들에 대한 관계까지 대리(공유)하지는 않는다.
- 이혼 소송 변호사들(대리인)이 이혼하는 부부들의 관계(부부)를 대리(공유)하지 않는 것 처럼
- 따라서 기하학에 따르면 Square IS-A Rectangle이지만
- 이들을 표현/대리(represent)하는 SW는 그들의 관계(IS-A)를 공유하지 않는다.