

Typing Issues and LSP

David Rabinowitz

Typing

- Static typing
 - Reliability
 - Catching errors early
 - Readability
 - Efficiency
- Dynamic typing
 - Flexibility

The graph plots 'Correction cost' on the y-axis (log scale from 1 to 1000) against 'Time error found' on the x-axis (stages: Requirements, Design, Code, Development test, Acceptance test, Operation). Two curves are shown: 'LARGE PROJECTS' (steeper curve) and 'SMALL PROJECTS' (flatter curve). The area between the curves is shaded pink.

March 28, 2006 Object Oriented Design Course 2

In the real world

- How do we add flexibility to static typing?
 - Genericity - C++ templates, Java Generics
 - Inheritance (including multiple inheritance)

March 28, 2006 Object Oriented Design Course 3

Covariance

```

Class Skier {
  Skier roommate;
  void share(Skier s);
}
Class Girl {
  Girl roommate;
  void share(Girl g);
}
Class RankedGirl {
  RankedGirl roommate;
  void share(RankedGirl rg);
}
  
```

The diagram shows a hierarchy where Skier is the base class. Girl and Boy inherit from Skier. RankedGirl inherits from Girl, and RankedBoy inherits from Boy.

March 28, 2006 Object Oriented Design Course 4

Covariance

- What happens when we do the following

```

Skier s; Boy b; Girl g;
b = new Boy(); g = new Girl();
s = b;
s.share(g);
  
```

March 28, 2006 Object Oriented Design Course 5

Covariance

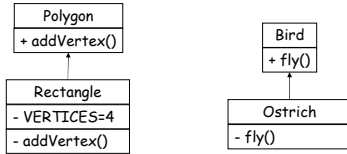
- What about multiple hierarchies?
- Does it solve the problem?

The diagram shows two parallel hierarchies. The first has Skier at the top, with Girl and RankedGirl below it. The second has Room at the top, with GirlRoom and RGRoom below it. Dotted arrows indicate relationships: Skier to Girl, Girl to RankedGirl, Room to GirlRoom, and GirlRoom to RGRoom.

March 28, 2006 Object Oriented Design Course 6

Descendant Hiding

- How can a class hide its parent's method?



March 28, 2006

Object Oriented Design Course

7

Solutions

- Some languages allow you to redeclare methods
 - No `Girl.share(Skier)`
- Java and C++ do not allow this
 - Need to check the validity at runtime
 - `if(skier instanceof Girl) {...}`
 - `if(skier instanceof Girl) {...} else throw...`

March 28, 2006

Object Oriented Design Course

8

The Liskov Substitution Principle

If for each object o_1 of type S there is an object o_2 of type T such that for all programs P defined in terms of T , the behavior of P is unchanged when o_1 is substituted for o_2 then S is a subtype of T .

Barbara Liskov, 1988

March 28, 2006

Object Oriented Design Course

9

LSP in plain English

Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it

March 28, 2006

Object Oriented Design Course

10

What's wrong with this?

```

void DrawShape(const Shape& s) {
    if (typeid(s) == typeid(Square))
        DrawSquare(static_cast<Square&>(s));
    else if (typeid(s) == typeid(Circle))
        DrawCircle(static_cast<Circle&>(s));
}
  
```

March 28, 2006

Object Oriented Design Course

11

Things Are Not Always That Simple

Consider the following class:

```

class Rectangle{
public:
    void SetWidth(double w) {_width=w;}
    void SetHeight(double h) {_height=h;}
    double GetHeight() const {return _height;}
    double GetWidth() const {return _width;}
private:
    double _width;
    double _height;
};
  
```

March 28, 2006

Object Oriented Design Course

12

Square

- We want to add a Square object
 - Naturally derives Rectangle
- And the trivial implementation is:


```
void Square::SetWidth(double w) {
    Rectangle::SetWidth(w);
    Rectangle::SetHeight(w);
}
void Square::SetHeight(double h) {
    Rectangle::SetHeight(h);
    Rectangle::SetWidth(h);
}
```
- Do you see any problem ?

March 28, 2006

Object Oriented Design Course

13

LSP is broken!

```
void g(Rectangle& r) {
    r.SetWidth(5);
    r.SetHeight(4);
    assert(r.GetWidth()*r.GetHeight()==20);
}
```

- A square object is not Rectangle object!
 - Their behavior is different

March 28, 2006

Object Oriented Design Course

14

The Liskov Substitution Principle

- Functions that use pointers or references to base classes must be able to use objects of derived classes without knowing it
- Use inheritance carefully!

March 28, 2006

Object Oriented Design Course

15