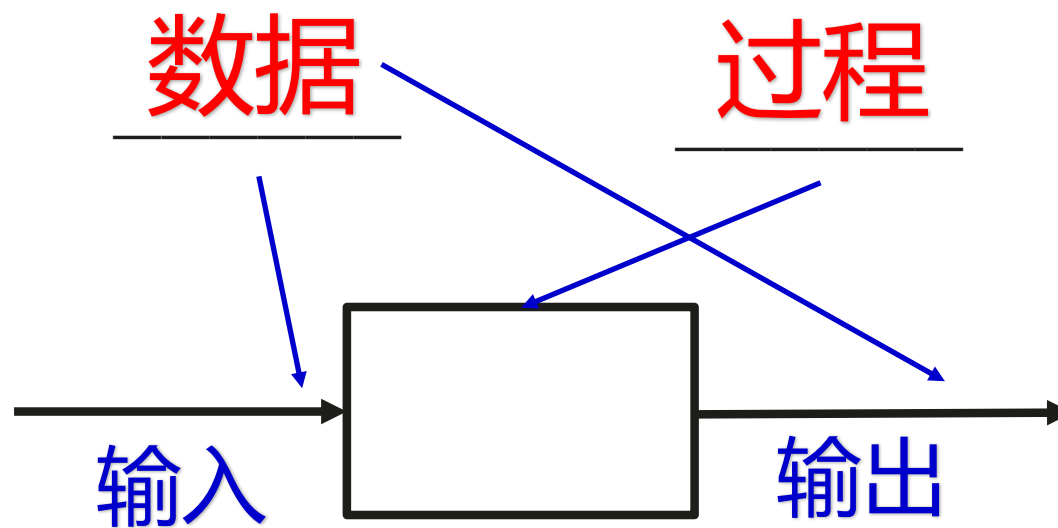


面向对象导入

面向对象导入



面向对象导入-非结构化编程

一个使用非结构化语言的程序经常包含按顺序排列的命令或声明，通常每个都占用一行。每一行都有编号或者标签，这样程序中的任意行都可以被执行。

```
10 i = 0
20 i = i + 1
30 PRINT i; " squared = "; i * i
40 IF i >= 10 THEN GOTO 60
50 GOTO 20
60 PRINT "Program Completed."
70 END
```



结构化编程（英语：Structured programming），一种编程典范。它采用子程序、代码区块、for循环以及while循环等结构，来取代传统的 goto。希望借此来改善计算机程序的明晰性、质量以及开发时间。

面向对象导入-过程式编程

过程式程序设计（英语：Procedural programming），又称过程式编程、过程化编程，一种编程典范，有时会被视为是指令式编程的同义语。派生自结构化编程（Structured programming），主要采取**程序调用**（procedure call）或**函数调用**（function call）的方式来进行流程控制。流程则由包涵一系列运算步骤的程序（Procedures），例程（routines），子程序（subroutines），方法（methods），或函数（functions）来控制。在程序运行的任何一个时间点，都可以调用某个特定的程序。任何一个特定的程序，也能被任意一个程序或是它自己本身调用。

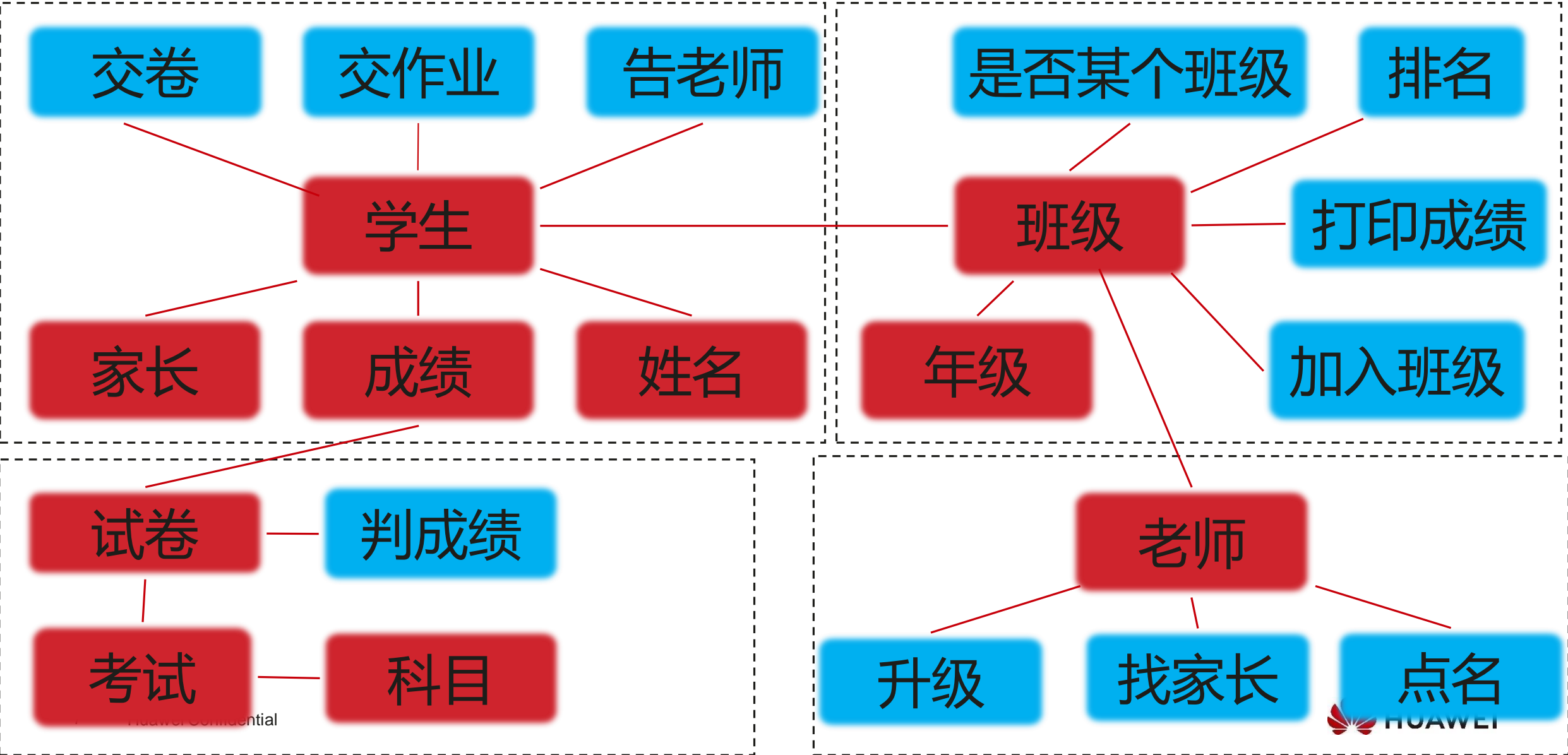


面向对象导入-为什么要使用面向对象

想象你有1000个数据，10000个过程，你怎么管理他们？



面向对象导入-为什么要使用面向对象



面向对象导入-面向对象式编程

面向过程的程序设计把计算机程序视为一系列的**命令集合**，即**一组函数的顺序执行**。为了简化程序设计，面向过程把函数继续切分为**子函数**，即把**大函数**通过**切割**成**小函数**来降低系统的复杂度。

面向对象的程序设计把计算机程序视为一组**对象的集合**，而每个对象都可以**接收**其他对象发过来的**消息**（包含信息的数据），并**处理**这些**消息**，计算机程序的执行就是一系列**消息在各个对象之间传递**。

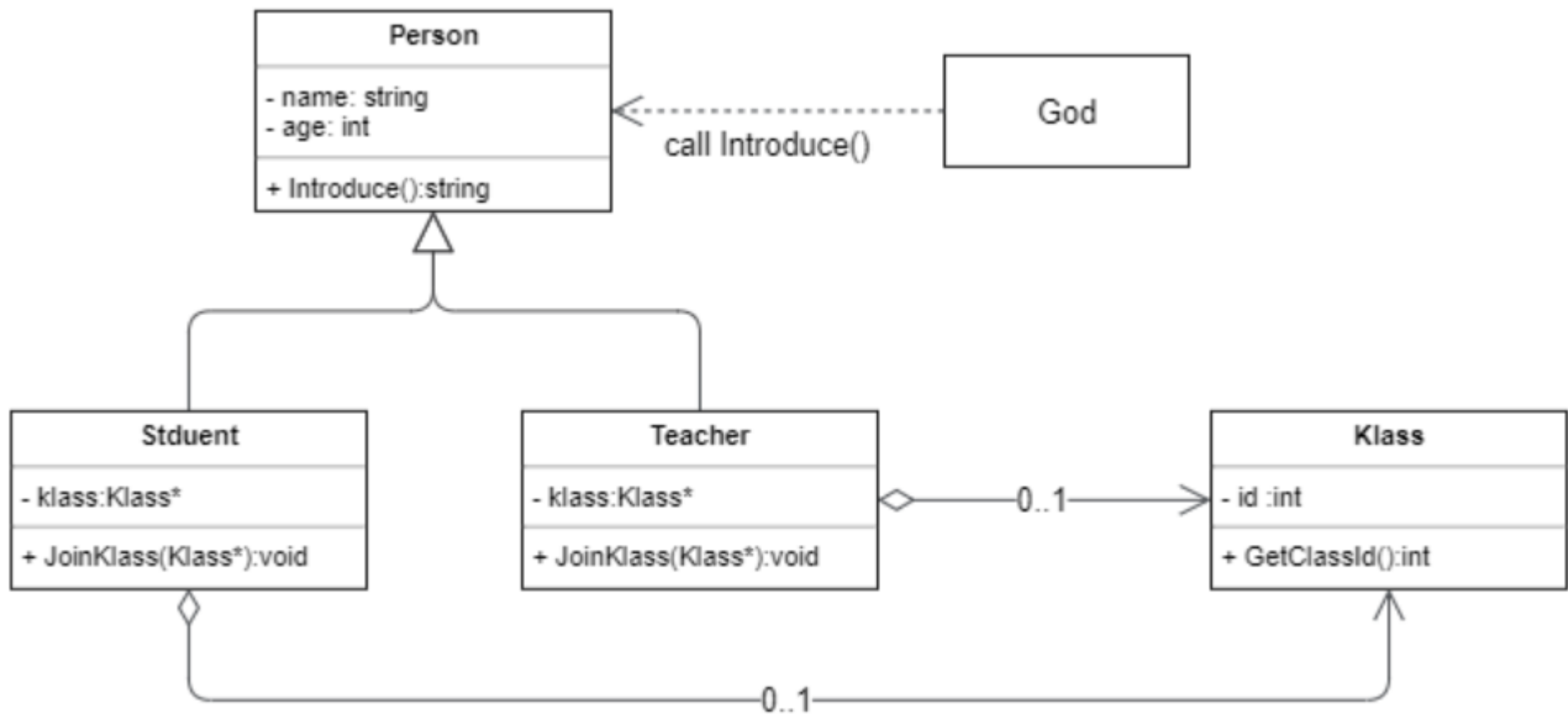
面向对象导入- UML类图

统一建模语言（英语：Unified Modeling Language，缩写 UML）是非专利的第三代建模和规约语言。

关 系	功 能	表 示 法
关联	类实例间连接的描述	——
依赖	两个模型元素间的关系	----->
流	在相继时间内一个对象的两种形式的关系	----->
泛化	更概括的描述和更具体的种类间的关系，适用于继承	——▷
实现	说明和实现间的关系	-----▷
使用	一个元素需要别的元素提供适当功能的情况	----->

51CTO.com
技术成就梦想

面向对象导入- UML类图



面向对象

面向对象

三大特性

类的关系

六大原则

设计模式

面向对象-三大特性

封装

对象**内部**访问控制

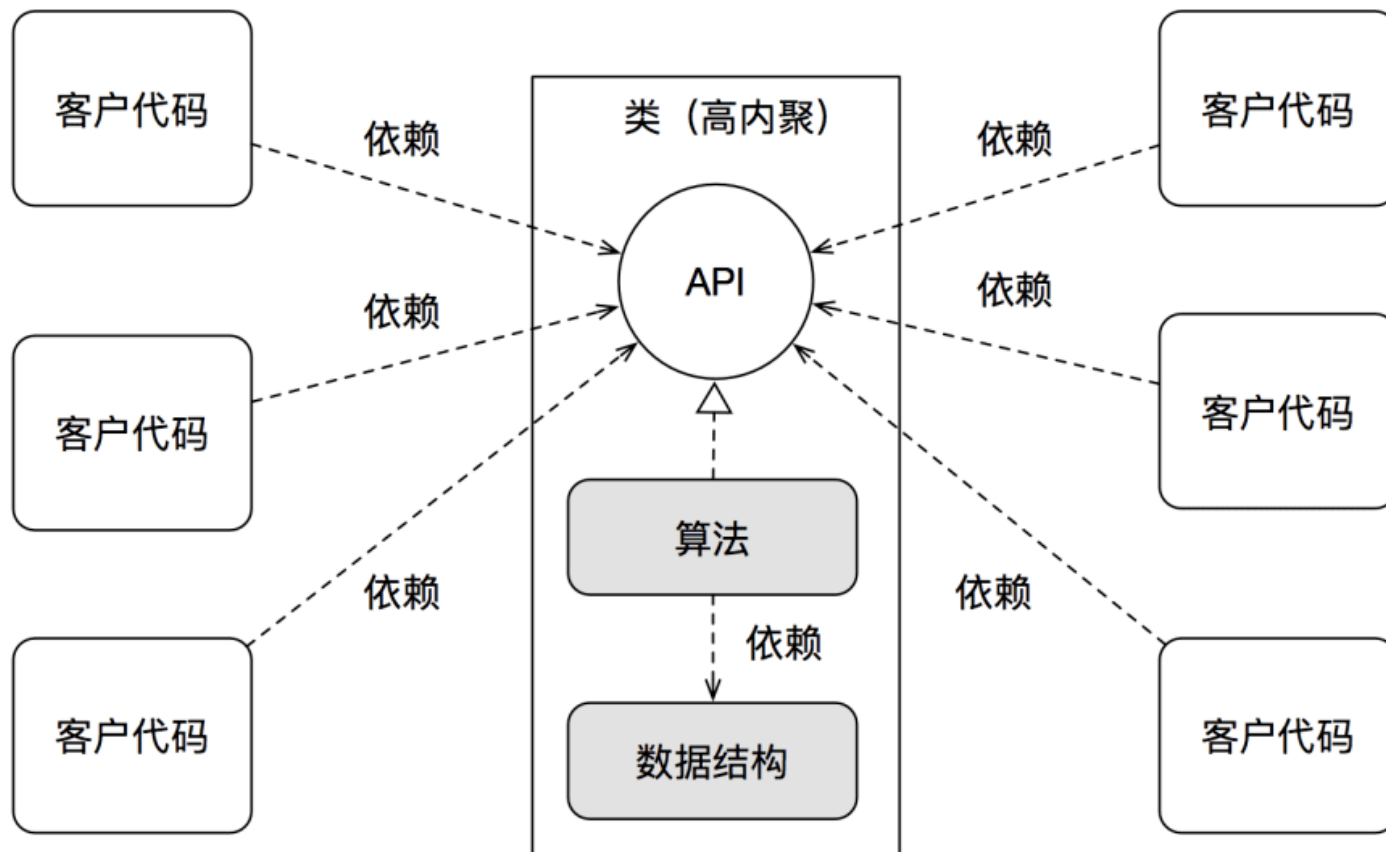
继承

复用实现 复用接口

多态

复用接口在**相同的业务行为约束**下体现**不同的细节**

面向对象-什么是封装



封装就是指隐藏对象的属性和实现细节，仅仅对外提供公共方法去访问它。

面向对象-CPP封装特性语法

【访问说明符】

public

protected

private

【默认访问属性】

struct

class

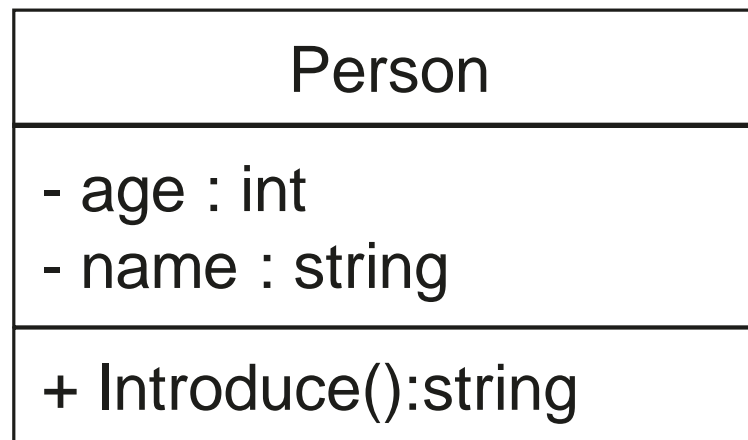
【友元】

friend

AC_1 封装

写一个Person类，要有name，age属性，要有一个Introduce方法，
introduce方法返回一个字符串形如：
My name is Tom. I am 21 years old.

面向对象-作业结果的类图展示



面向对象-封装的好处



简化用户逻辑

用户只需要知道有一个对象

(煎饼摊老板)

向该对象发送消息

(调用摊煎饼接口)

由接口返回一个结果

(煎饼)

而无需用户自己关注内部细节

(油、盐、炉子、配方、火候、 etc.)

掌握数据行为控制权



降低耦合，减少变化带来的影响



“亲，我们的数据结构变了”

面向对象-getter和setter

问题：getter、setter破坏封装吗？

```
public class Person {  
    // 属性 (数据)  
    private String name;  
  
    public person(string name){  
        this.name = name;  
    }  
  
    // 方法 (行为)  
    public void setName(String name) {  
            this.name = name;  
    }  
  
    public void Rename(String name) {  
        this.name = name;  
    }  
  
    public String getName() {  
        return this.name;  
    }  
}
```

面向对象-继承/多态

```
class Base {  
public:  
    void Bar()  
    {  
        cout << "BaseBar"<< endl;  
    }  
  
    virtual void Foo() = 0;  
  
    virtual void FooBar()  
    {  
        cout << "BaseFooBar"<< endl;  
    }  
};
```

```
class Derived : public Base {  
public:  
    virtual void Foo()  
    {  
        cout << "DerivedFoo"<< endl;  
    }  
    virtual void FooBar()  
    {  
        cout << "DerivedBar"<< endl;  
    }  
};
```

面向对象-继承/多态

【继承的内容】

继承实现

继承接口：函数多态 Polymorphism

输出什么？



```
class Base {  
public:  
    virtual void Foo(){cout << "Base"<< endl;}  
};  
class Derived : public Base {  
public:  
    virtual void Foo(){cout << "Derived"<< endl;}  
};  
Derived d;  
Base* p = &d;  
Base& r = d;  
  
p->Foo();  
r.Foo();
```

面向对象-继承的访问控制语法

【继承的访问控制】

显式制定

默认制定

【继承访问控制语法】

public

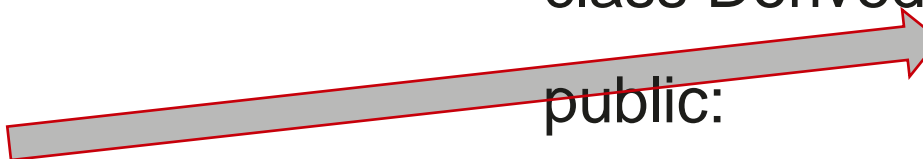
protected

private

替换会发生什么?

```
class Base {  
    public:  
        virtual void Foo();  
};
```

```
class Derived : public Base {  
    public:  
        virtual void Foo();  
};
```



面向对象-CPP继承与多态练习

AC_2 继承

再写一个Student类继承Person类，除了name，age属性，还有要有class属性。也有一个introduce方法，introduce方法返回一个字符串形如：

My name is Tom. I am 21 years old. I am a Student. I am at Class 2.

再写一个Worker类继承Person类，只有name，age属性。也有一个introduce方法，introduce方法返回一个字符串形如：

My name is Tom. I am 21 years old. I am a Worker. I have a job.

所有Person的子类的这段文字

My name is Tom. I am 21 years old.

都应该调用Person的introduce方法来获得

AC_3 更多继承

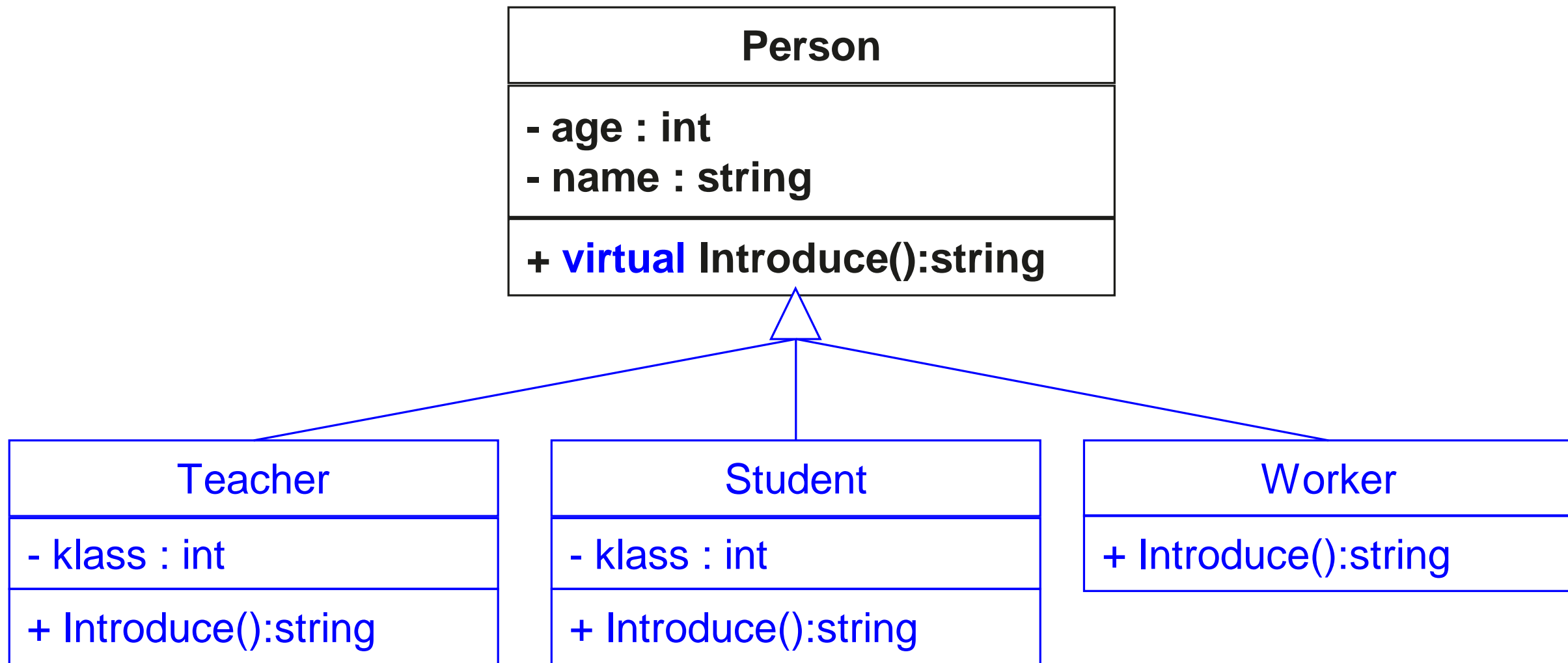
再写一个Teacher类继承Person类，除了name，age属性，也有class属性。也有一个introduce方法，introduce方法返回一个字符串形如：

My name is Tom. I am 21 years old. I am a Teacher. I teach Class 2.

如果class为0，就会返回

My name is Tom. I am 21 years old. I am a Teacher. I teach No Class.

面向对象-作业结果的类图展示



类的关系

类与类的关系

- **is-a**

什么是is-a

继承与is-a

程序世界的is-a和真实世界的is-a

- **has-a**

组合与聚合

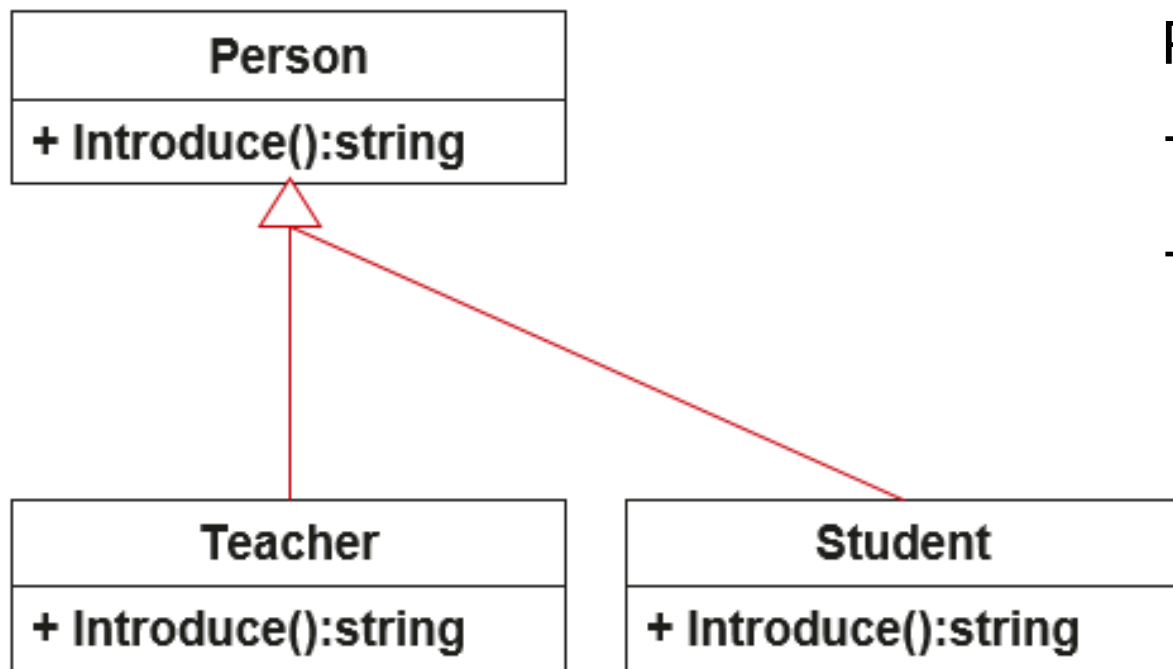
扩展：继承与has-a

类与类的关系-什么是is-a



这是____? 为什么你觉得它是____?

类与类的关系-继承与is-a的关系



含义一：

Person能表现出自我介绍

Teacher是（is-a）Person

Teacher能表现出自我介绍

含义二：

Person能表现出自我介绍

God不能表现出自我介绍

God必然不是Person

类与类的关系-现实生活的对象和建模的对象

```
class Bird {  
    public:  
        virtual std::string Fly() = 0;  
};
```

这是___?

此Bird和生活中的Bird是一回事吗?

类与类的关系—现实生活的对象和建模的对象

【三段论-1】

大前提：人都会死

小前提：苏格拉底是人

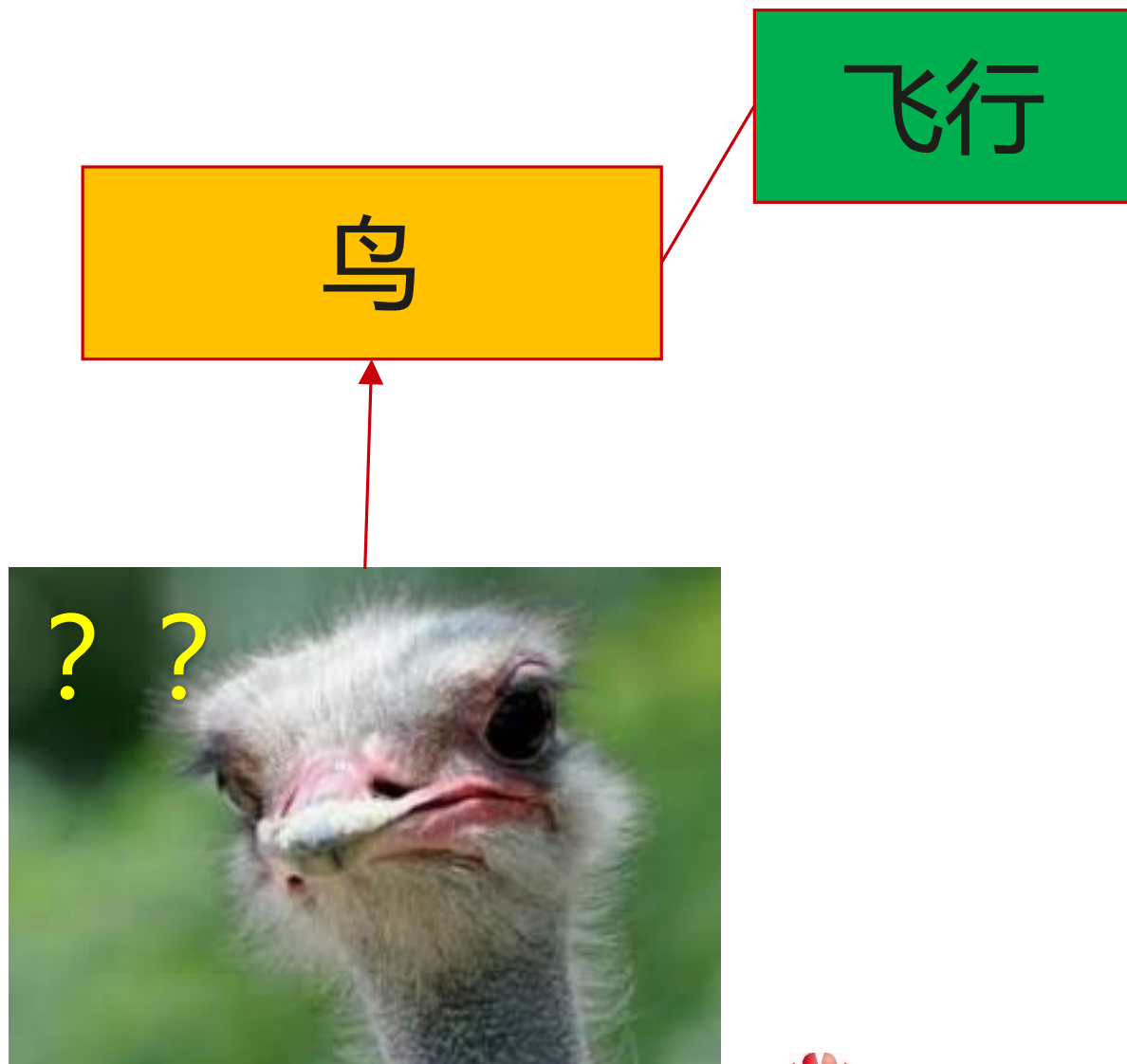
→ 苏格拉底会死

【三段论-2】

大前提：鸟会飞

小前提：鸵鸟是鸟

→ 鸵鸟会飞



类与类的关系—现实生活的对象和建模的对象

【三段论-1】

大前提：只要会死就是平凡的

小前提：苏格拉底会死

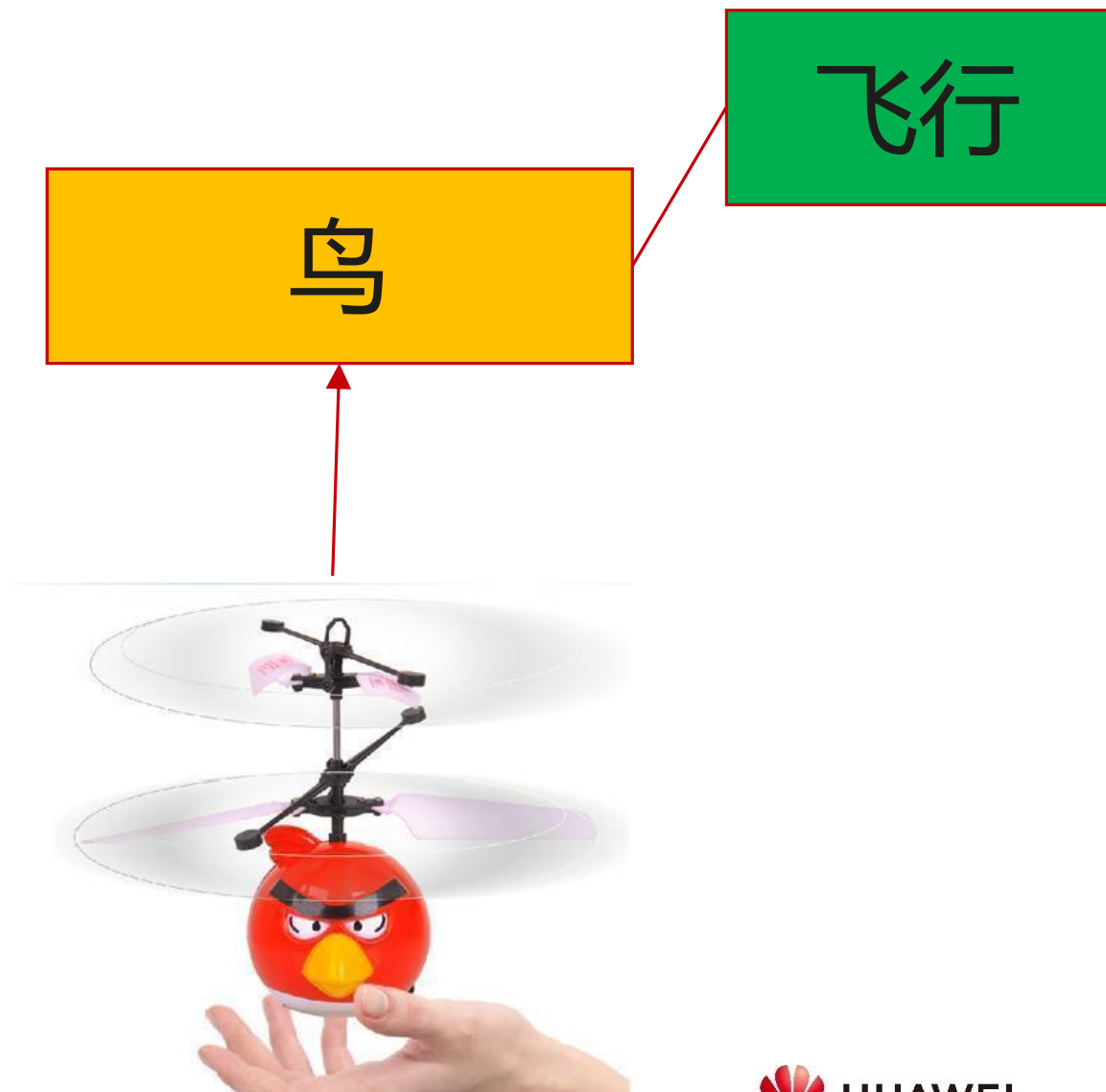
→ 苏格拉底是平凡的

【三段论-2】

大前提：只要会飞就可以是鸟

小前提：玩具鸟会飞

→ 玩具鸟可以是鸟

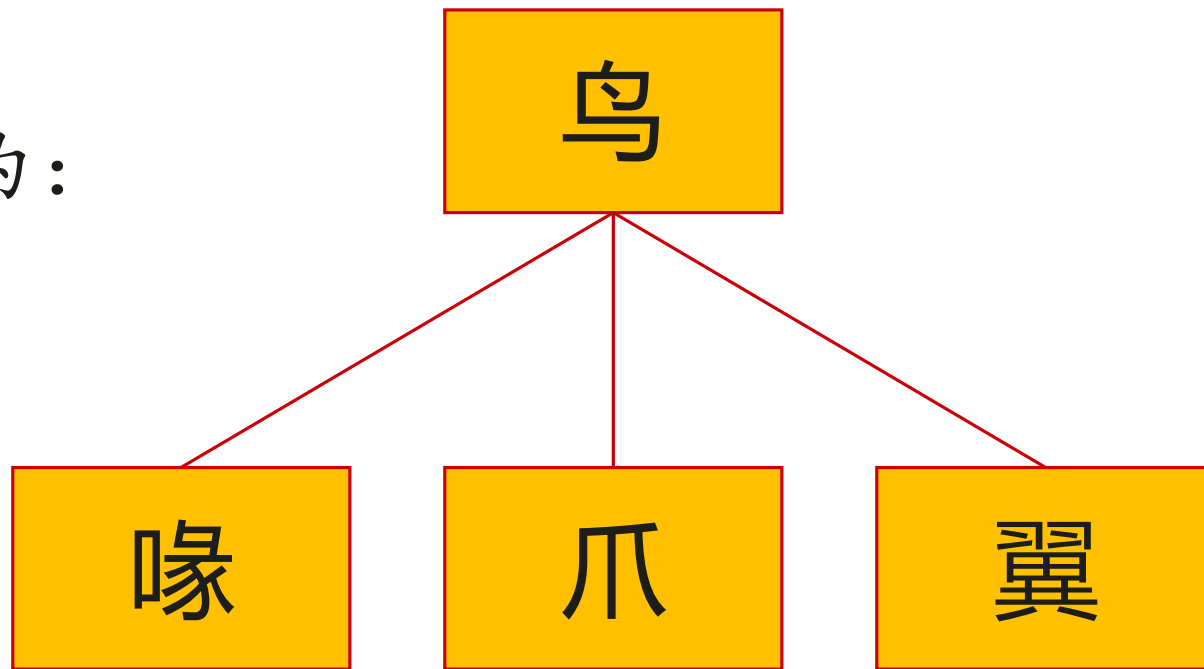


类与类的关系-组合

组合描述的是一种has的关系

组合根据具体细节不同，又可分为：

- 关联 (Association)
- 聚合 (Aggregation)
- 合成 (Composition)



类与类的关系-组合与生命周期

【选择题】

生命周期A和B相同的是__?

[I]

```
class A{  
public:  
    A() = default;  
    ~A() = default;  
private:  
    B b_;  
};
```

[II]

```
class A{  
public:  
    A(B& b):b_(b){}  
    ~A() = default;  
private:  
    B& b_;  
};
```

[III]

```
class A{  
public:  
    A(B*b):b_(b){}  
    ~A(){ b_ = nullptr;}  
private:  
    B* b_;  
};
```

【问答题】

II和III的区别是什么

类与类的关系-组合

AC_4 组合

Student的class属性不是一个数字，而是一个对象，写一个Class类，有number属性。Student构造的时候把Class的一个实例传给Student，Teacher一样。

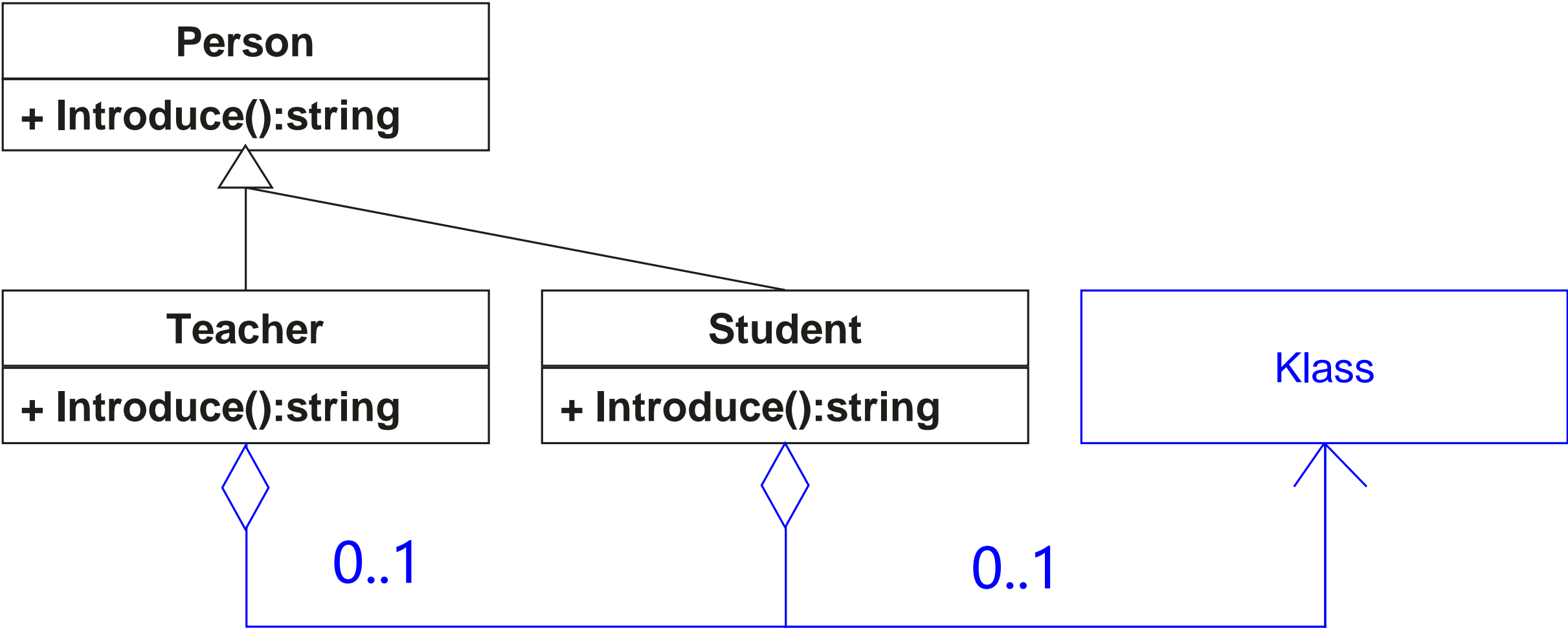
如果Teacher的class为null，introduce函数就会返回

My name is Tom. I am 21 years old. I am a Teacher. I teach No Class.

如果Stdudent的class为null，introduce函数就会返回

My name is Tom. I am 21 years old. I am a Student. I am at No Class.

类与类的关系-作业结果的类图展示



类与类的关系-封装与组合练习

AC_5 更多组合

给Teacher写一个introduceWith方法，传入一个student，比如Jerry，如果Jerry在Teacher教的班级则返回形如

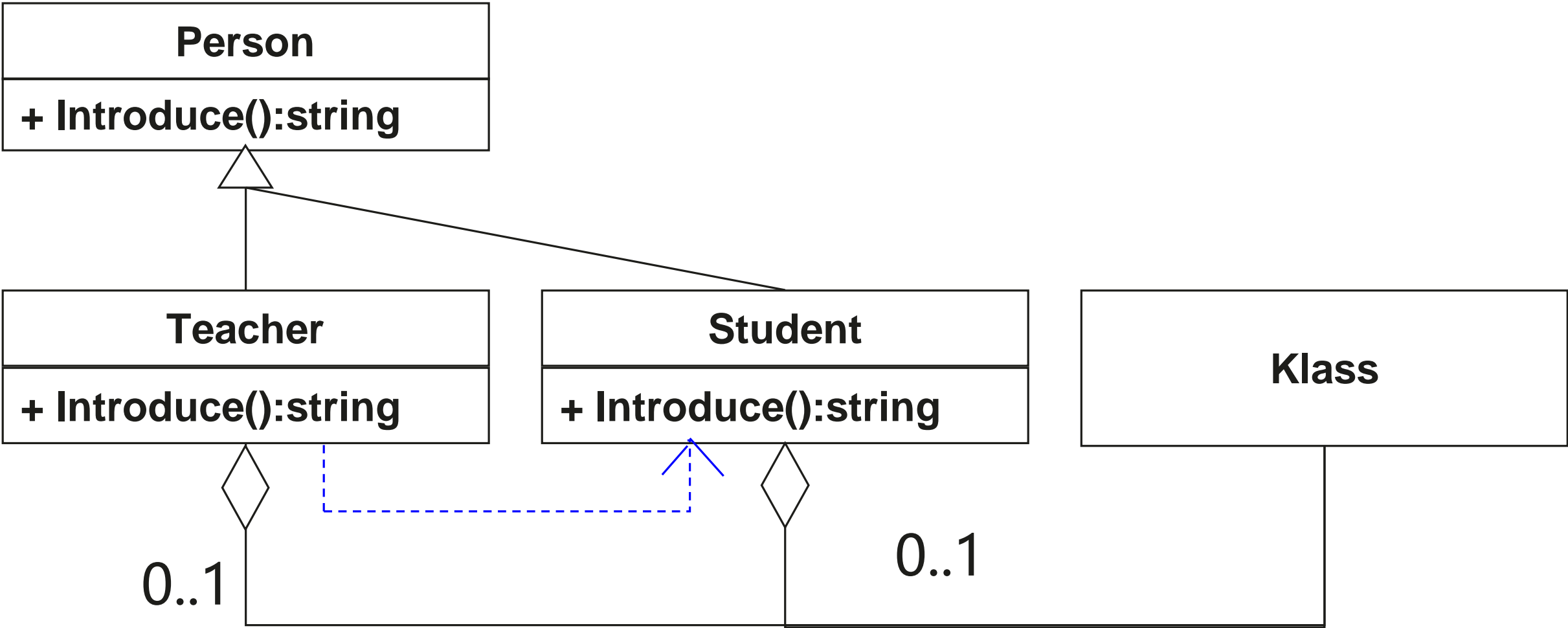
My name is Tom. I am 21 years old. I am a Teacher. I teach Jerry.

否则返回

My name is Tom. I am 21 years old. I am a Teacher. I don't teach Jerry.

抽取一个私有函数来复用字符串。

类与类的关系-作业结果的类图展示



类与类的关系-组合的循环依赖

AC_6 循环依赖

Person类，加入id属性，靠id来判断是否是同一个人。

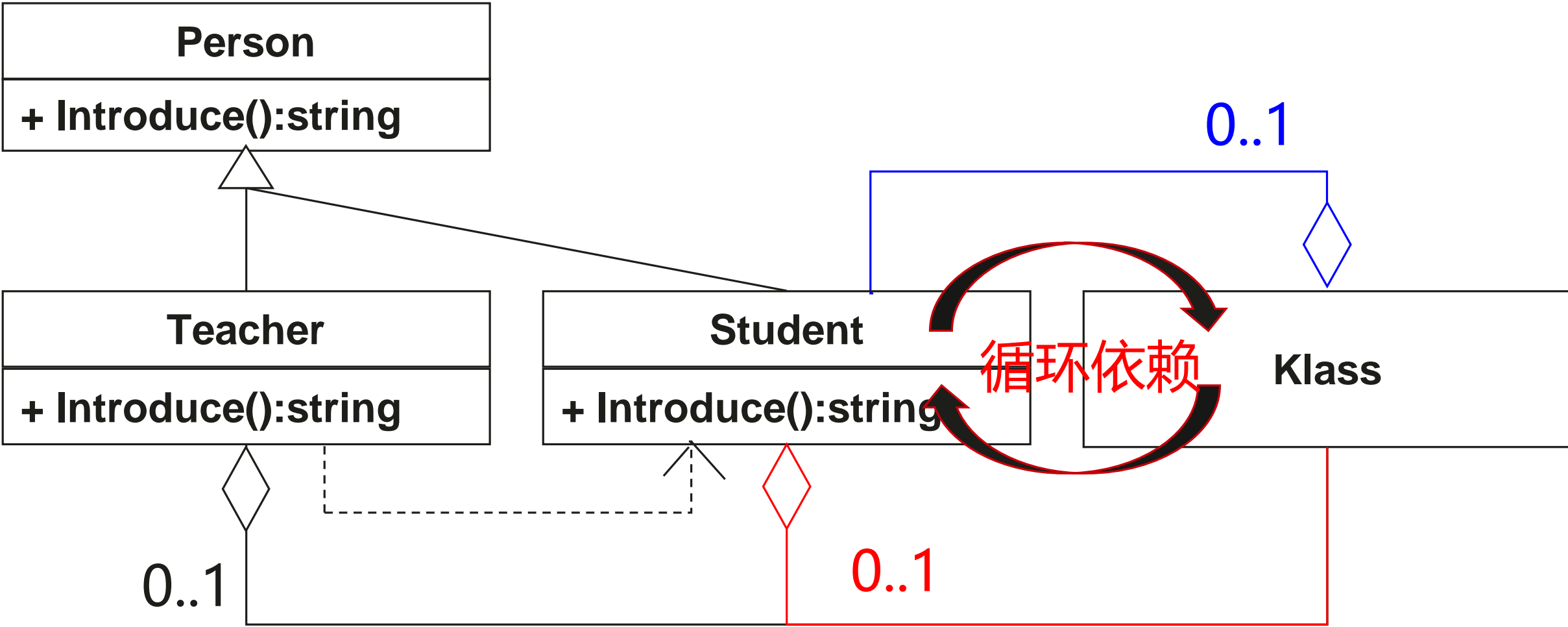
Class类，有number属性还有一个Student类型的leader属性。但是leader属性不在构造器里。

Student构造的时候把Class的一个实例传给Student。Class有一个assignLeader方法，接收一个Student实例。意为将一名学生设置为该Class的班长。如果Class的Leader是Tom，那么Tom调用introduce的方法就要返回：

My name is Tom. I am 21 years old. I am a Student. I am
Leader of Class 2.

如果没有就继续返回旧的字符串

类与类的关系-作业结果的类图展示



类与类的关系-组合进阶一对多

AC-7 一对多

Class还有一个appendMember方法，接受一个Student实例。意味将一名学生加入该班级。如果学生没有加入该班级，那么在调用assignLeader方法的时候，不会assign成功，会打印一句话：

It is not one of us.

相应的调用Student的introduce方法也只会返回旧的字符串。

Teacher的class属性改为classes属性，也就是可以教多个班。introduce方法返回一个字符串形如：

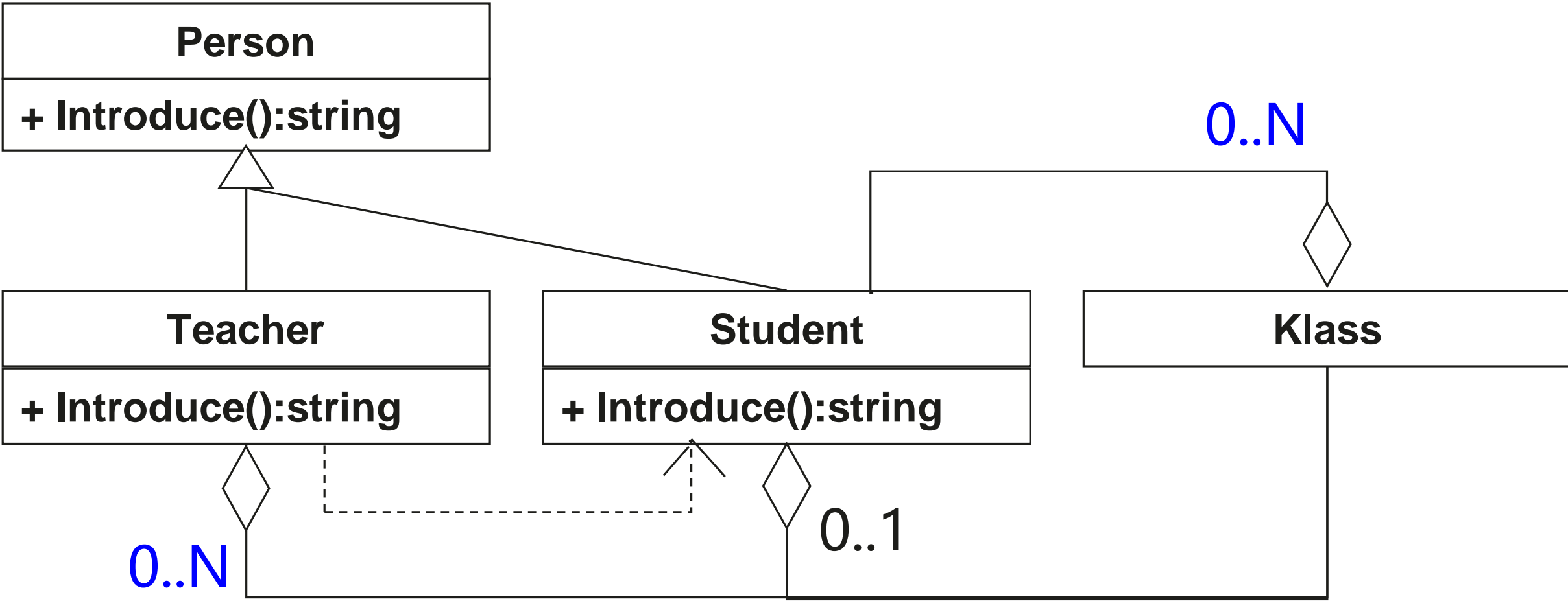
My name is Tom. I am 21 years old. I am a Teacher. I teach Class 2, 3.

如果classes的长度为0，就会返回

My name is Tom. I am 21 years old. I am a Teacher. I teach No Class.

Teacher还有一个isTeaching方法，传入一个学生，返回true/false。只要学生在classes中的任一个class中，就是在教他。而学生是否在class中这件事情，应该是Class有一个方法hasMember来判断。

类与类的关系-作业结果的类图展示



设计原则

设计原则

SOLID原则

单一职责原则 (The Single Responsibility Principle, 简称 SRP)

开放-封闭原则 (The Open-Close Principle, 简称 OCP)

里氏替换原则 (The Liskov Substitution Principle, 简称 LSP)

接口隔离原则 (The Interface Segregation Interface, 简称 ISP)

依赖倒置原则 (The Dependency Inversion Principle, 简称 DIP)

迪米特法则 (Law of Demeter, 简称 LoD)

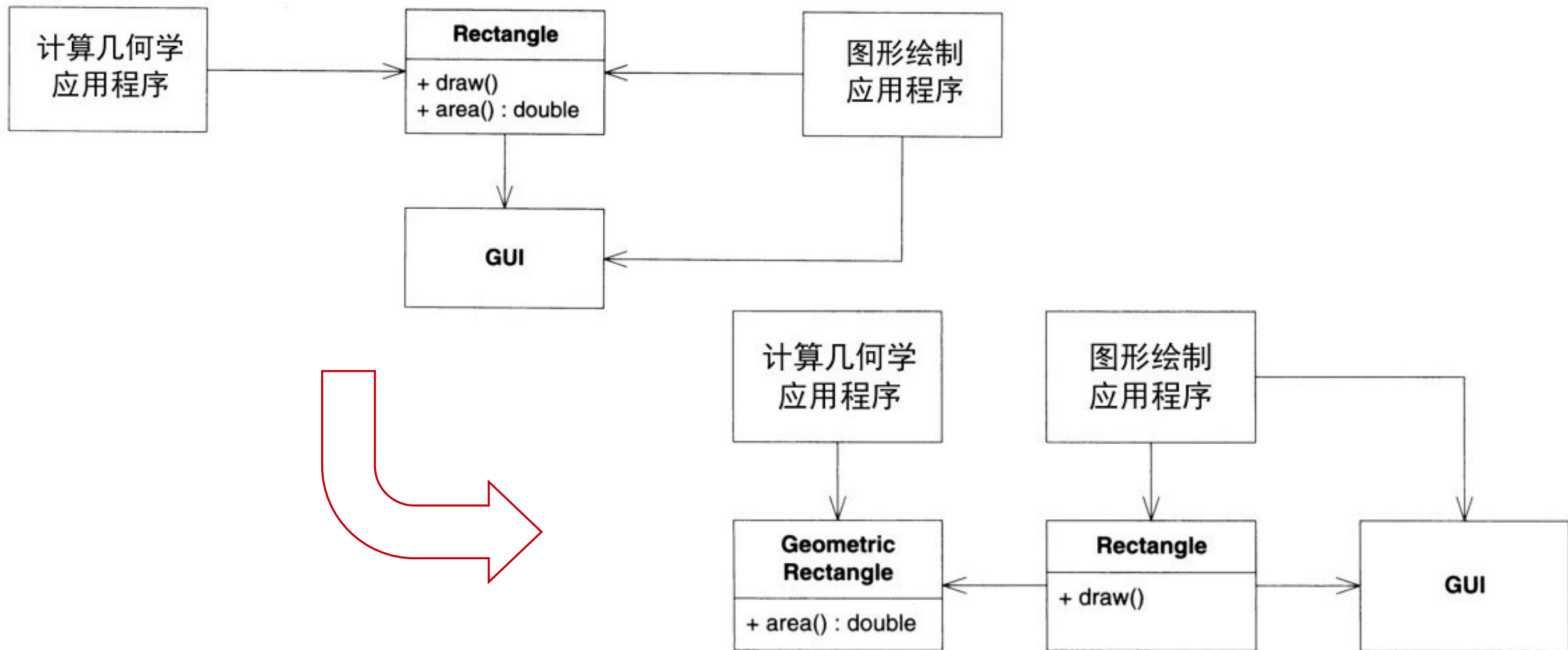
DRY原则

设计原则-单一职责原则



一个类应该只有一个发生变化的原因。

设计原则-单一职责原则

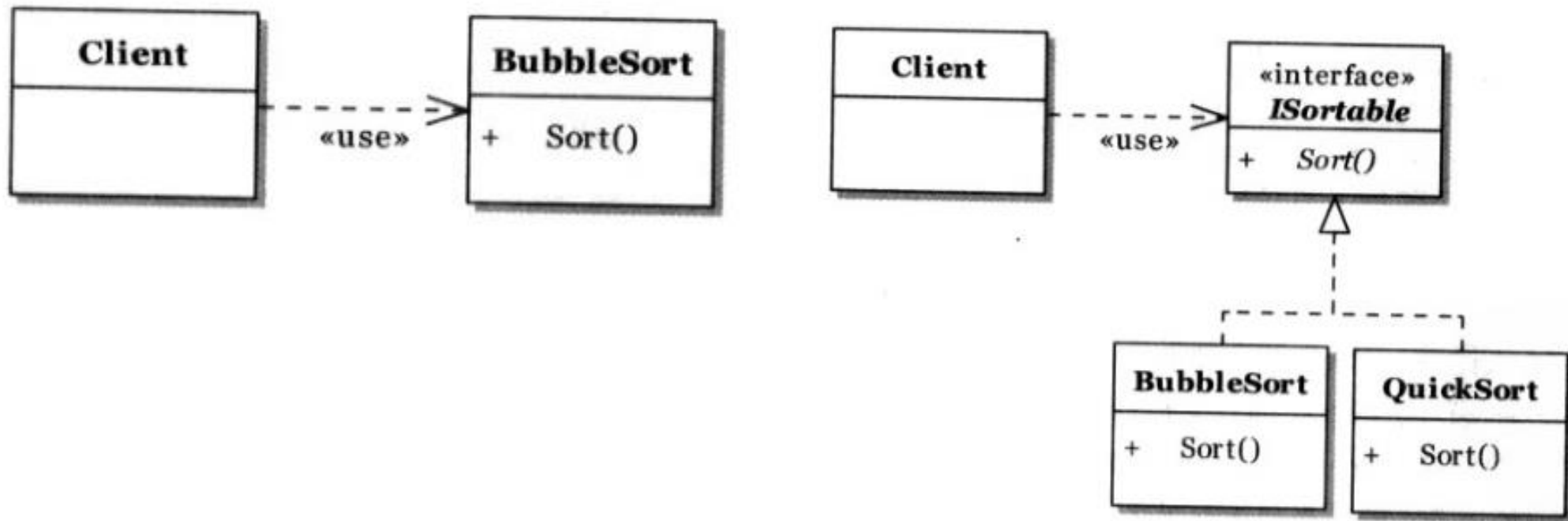


设计原则-封闭原则

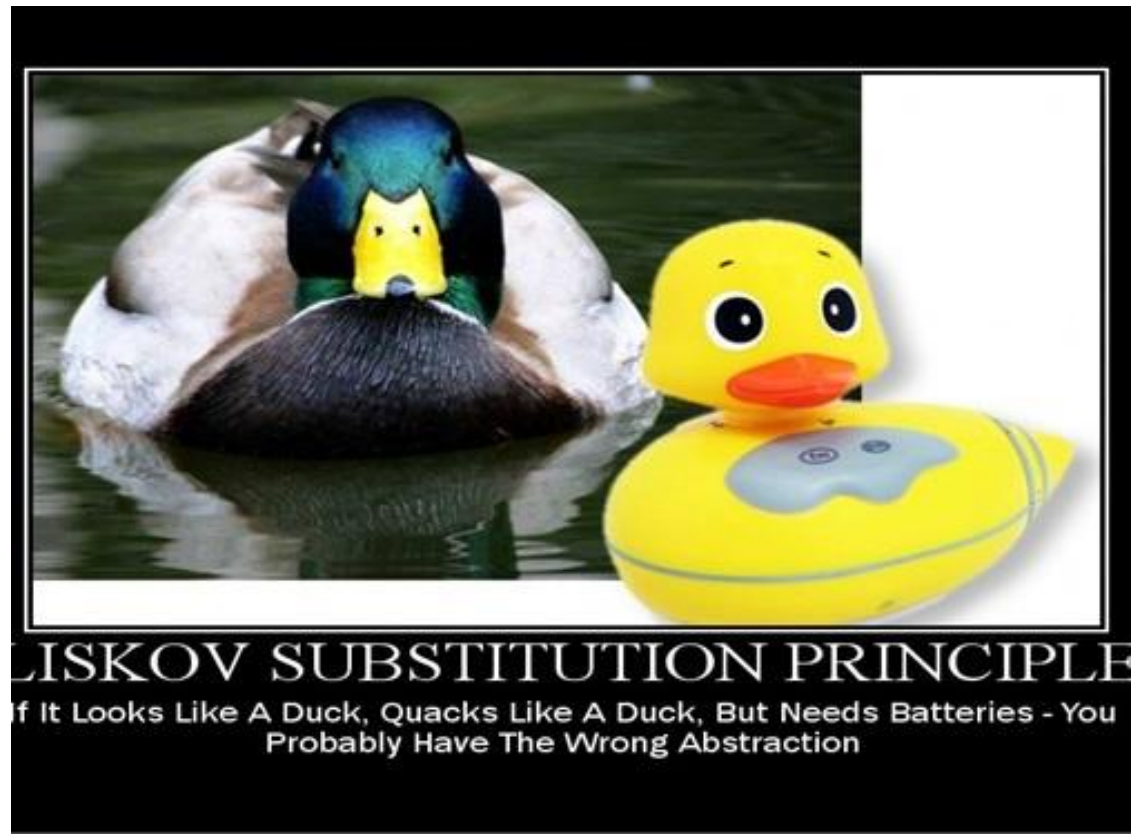


软件实体（类、模块、函数等）应该对扩展开放，对修改封闭。

设计原则-封闭原则



设计原则-里氏替换原则



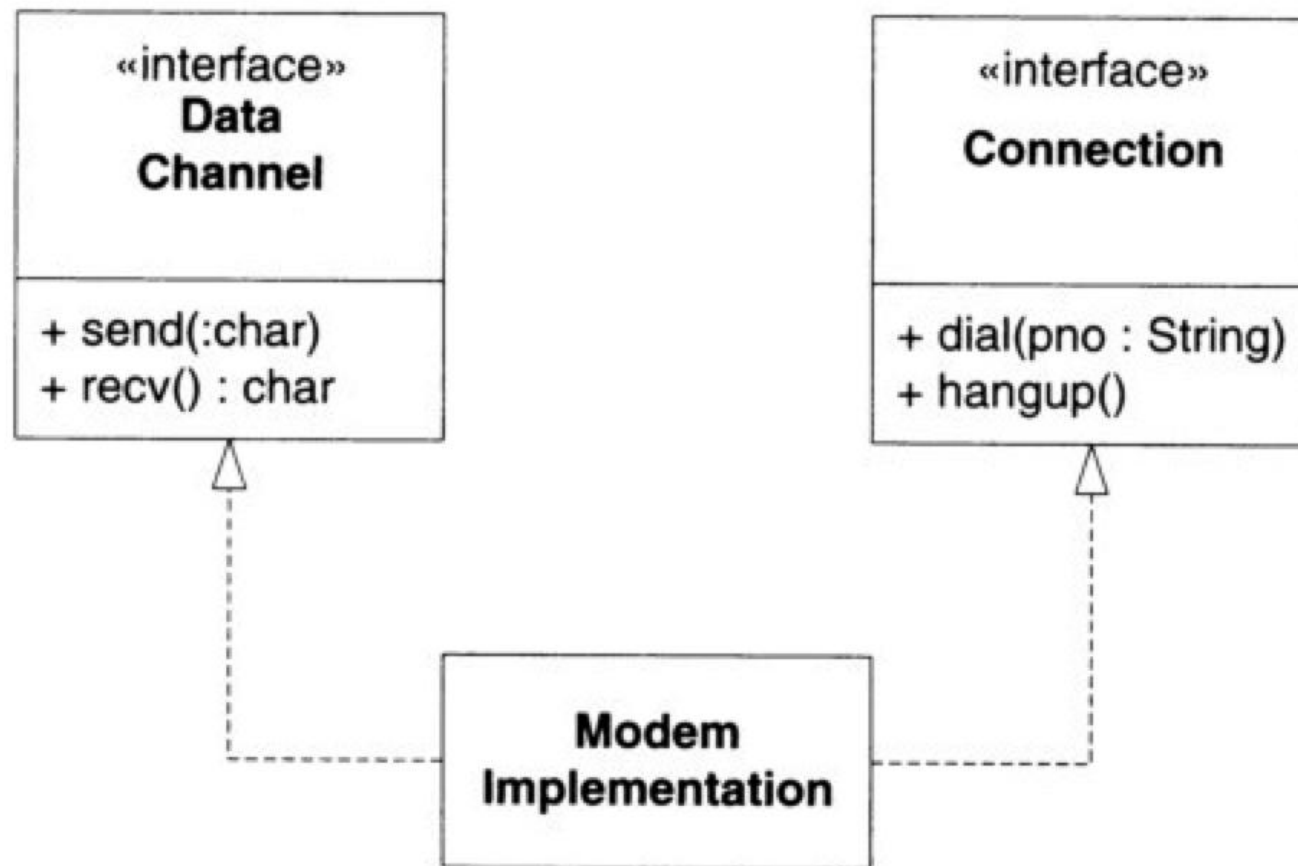
派生类（子类）对象可以在程式中代替其基类（超类）对象。

设计原则-接口隔离原则

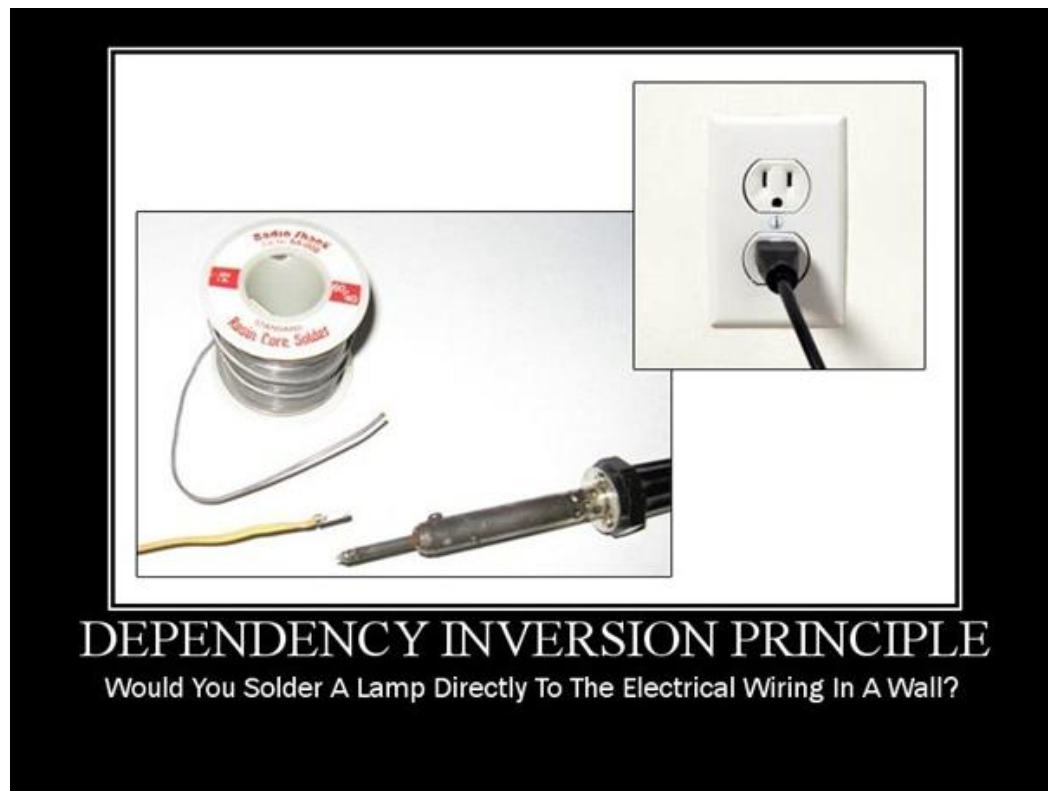


客户端不应该被迫依赖于它不使用的方法

设计原则-接口隔离原则

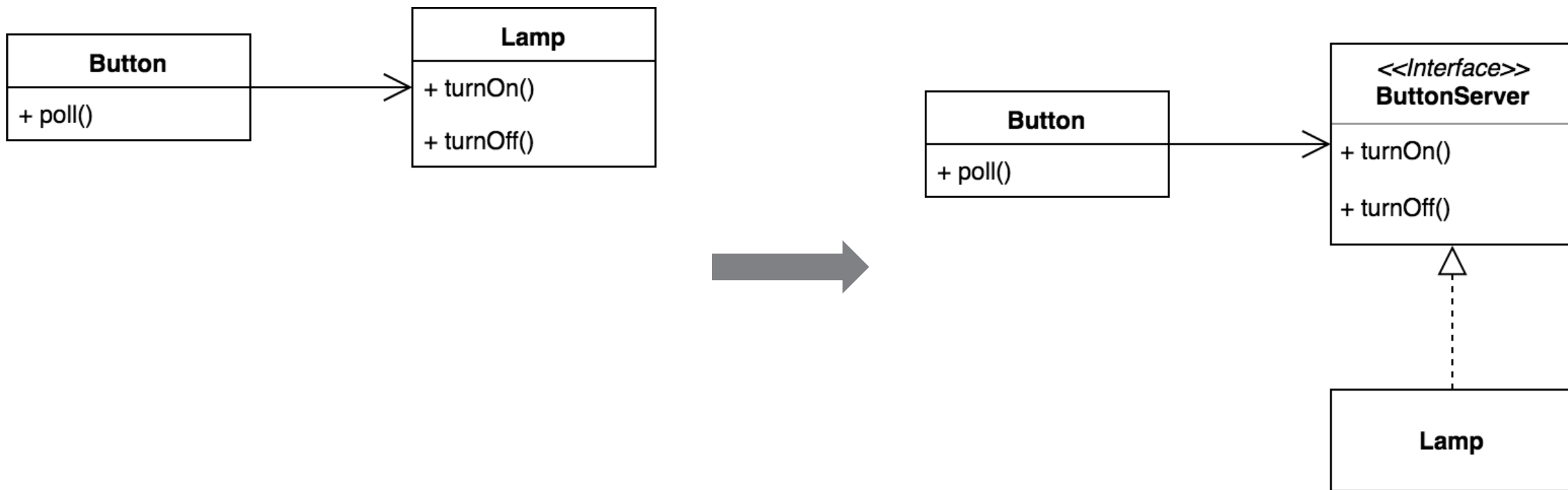


设计原则-依赖倒置原则

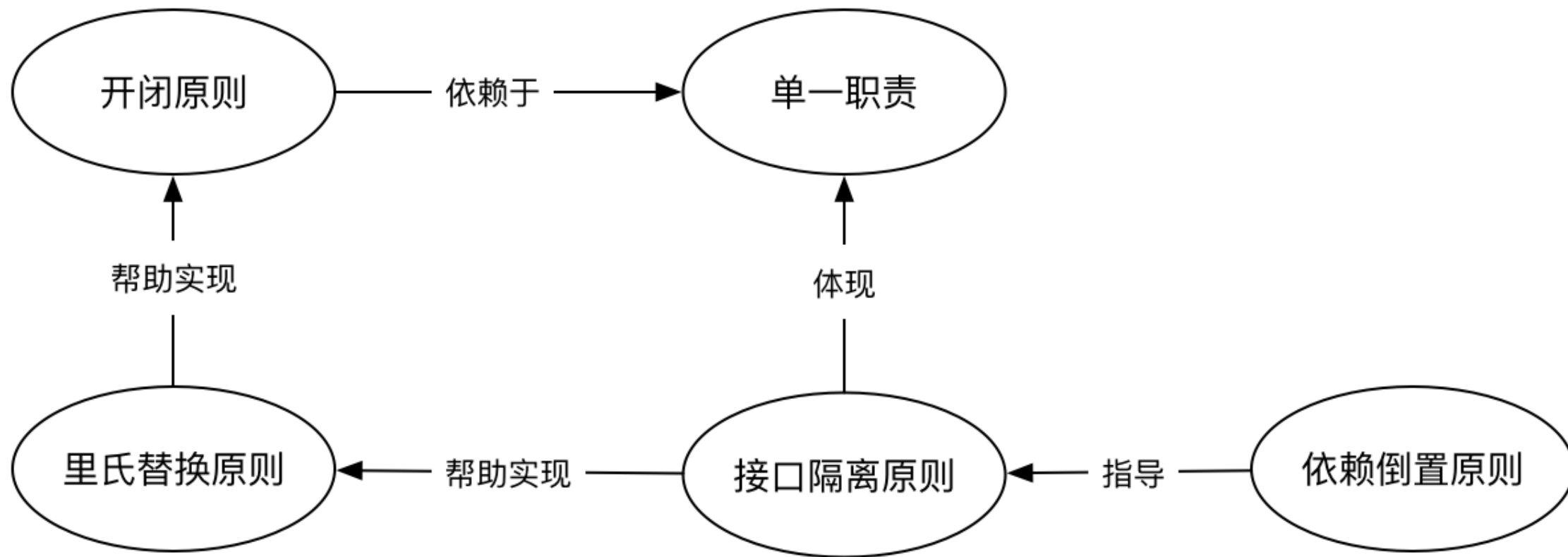


高层模块不应该依赖底层模块，两者都应该依赖底层模块的抽象。

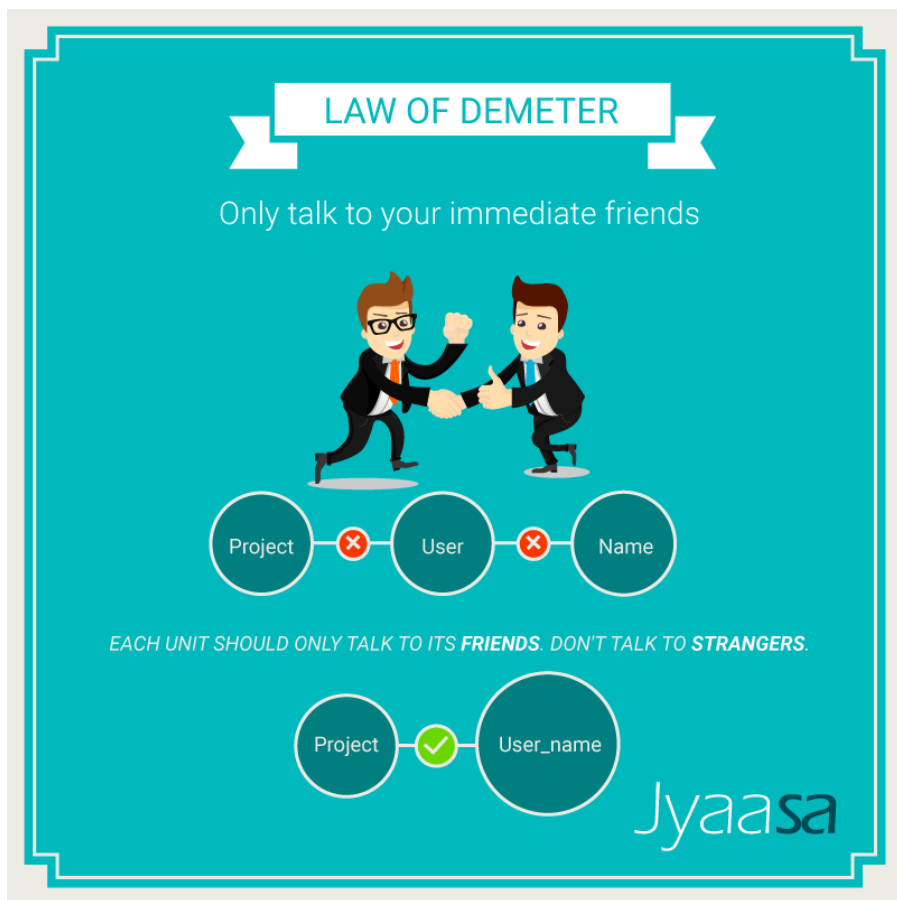
设计原则-依赖倒置原则



设计原则-五个原则的关系



设计原则-迪米特法则



* 原则： 迪米特法则（Law of Demeter）
或者称之为 最少知识原则LKP（Least Knowledge Principle）

* 多种表述

- * 每个软件单元应该只知道关于别的单元的有限的知识，并且应该只知道紧密关联的单元的知识。
- * 只跟朋友说话，不跟陌生人说话。
- * 只跟你直接的朋友说话。

设计原则-DRY > KISS > YAGNI



DRY : Don' t Repeat Yourself

KISS : Keep it Simple and Stupid

YAGNI: You Aren' t Gonna Need It

设计模式

设计模式-观察者模式

AC-8 观察者

当学生加入Teacher教的班级的时候，Teacher会打印一个句话，形如：

I am Tom. I know Jerry has joined Class 2.

当学生成为Teacher教的班级的班长的时候，Teacher会打印一个句话，形如：

I am Tom. I know Jerry become Leader of Class 2.

再引入一个类，叫Computer，Computer有一个name属性，Computer不是Person的子类。

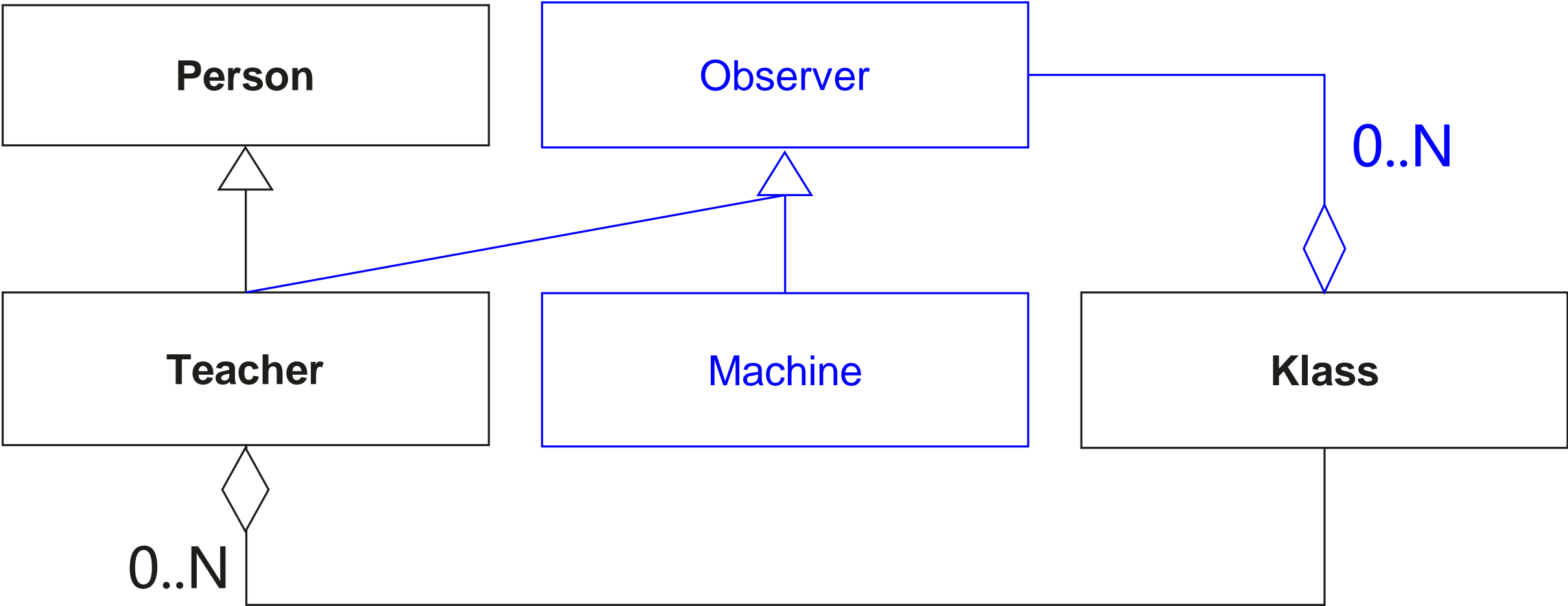
当学生加入Computer关心的班级的时候，不仅Teacher会打印，Computer也会打印一个句话，形如：

I am the Machine. I know Jerry has joined Class 2.

当学生成为Computer关心的班级的班长的时候，不仅Teacher会打印，Computer也会打印一个句话，形如：

I am the Machine. I know Jerry become Leader of Class 2.

设计模式-作业结果的类图展示



Thank You

面向对象-继承背后的含义

【继承访问控制语法】

public

private

protected

【继承的意义】

is-a

has-a/implemented-by

child and me both has-a or
both can implemented-by