# Ch. 6 面向对象

2017.5

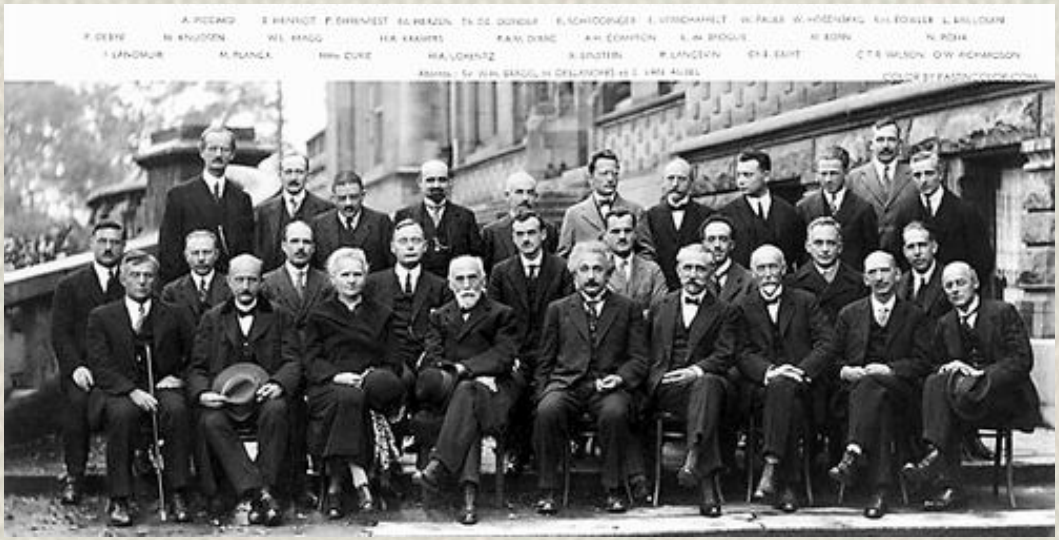
# §6.0 识“图”课

- 最简单有效的沟通方式

# 图片

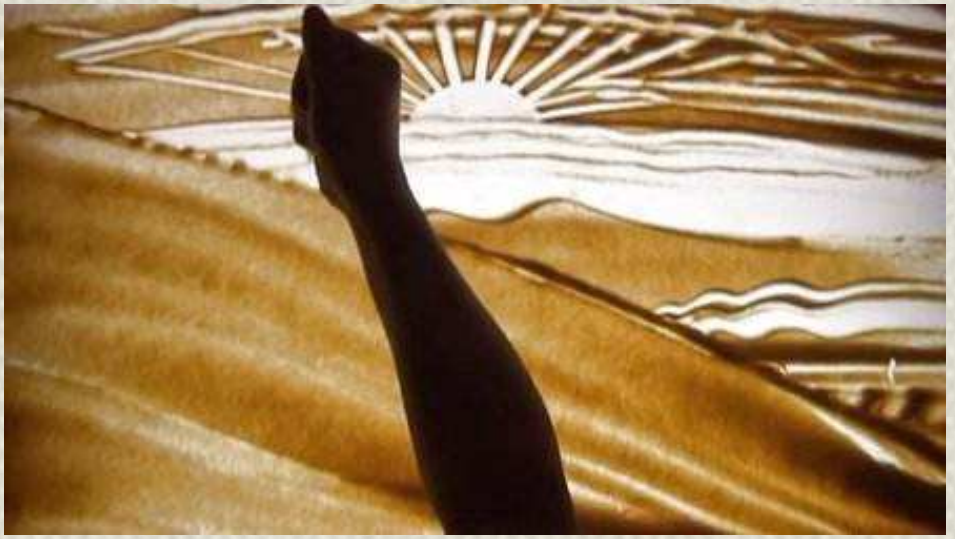
- Photo
- Picture
- Image 概念缩小

# Photo

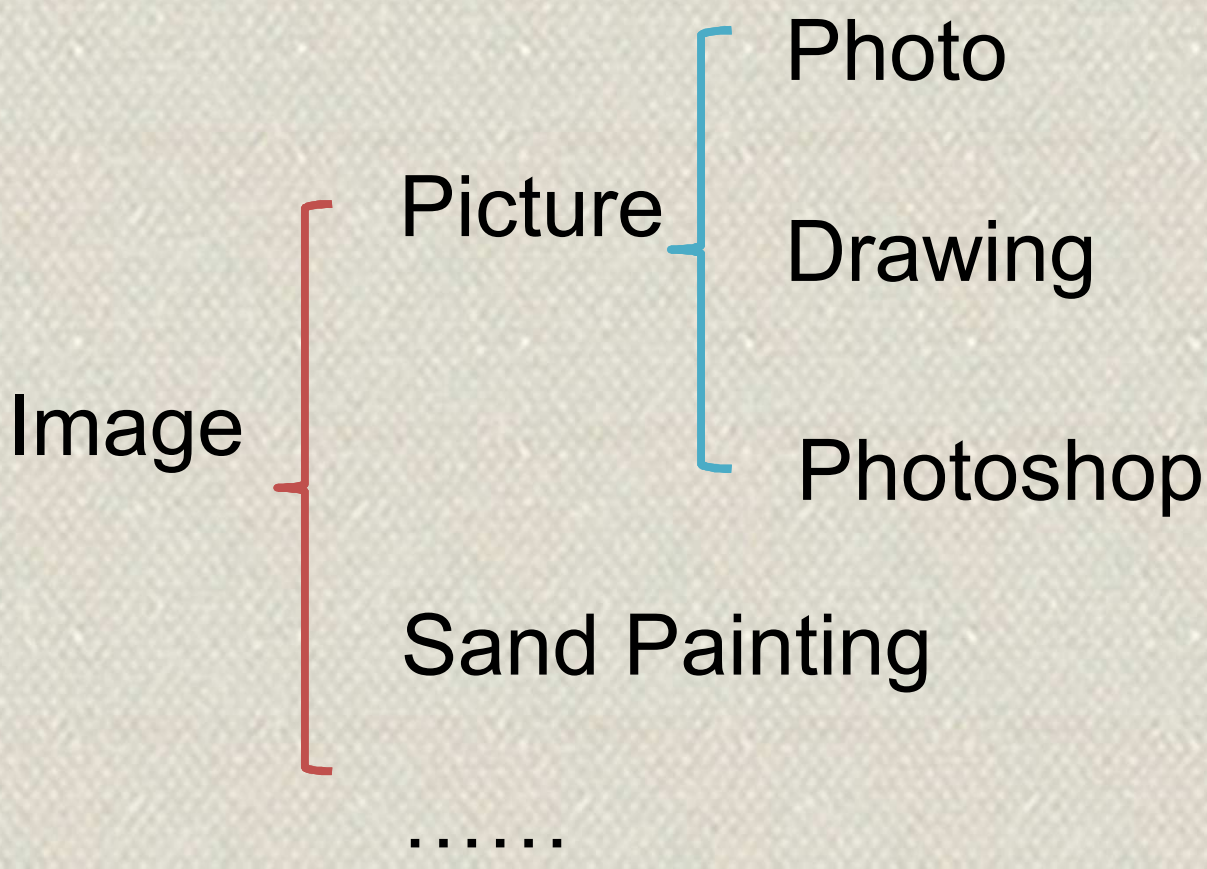


# Painting & Drawing

# 看内容



# 基本关系



# 图示

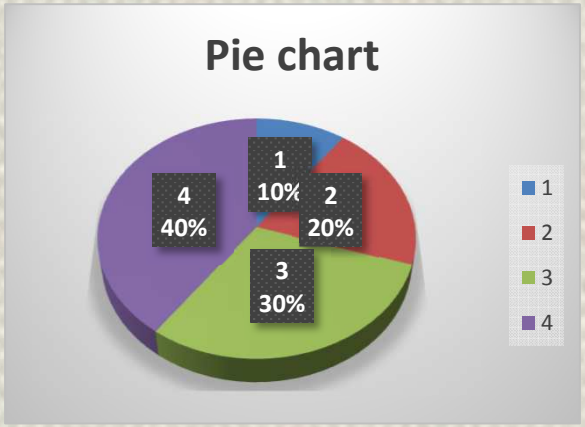
- Graph
- Chart
- Diagram
- Figure
- Illustration

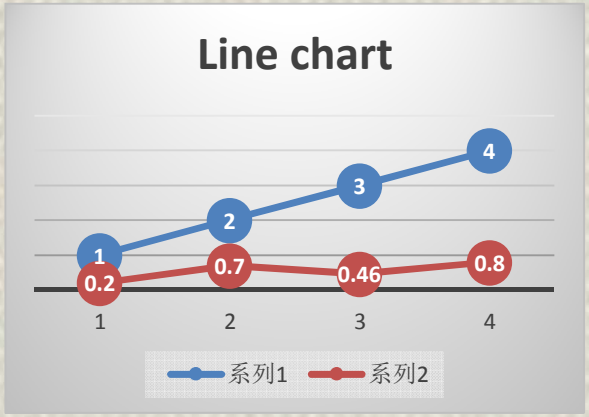
# Graph & Chart

- We usually use graphs to show analysis result, such as pie chart, bar chart, line chart and so on.
- Graph最初指函数变化关系的图，被泛用

# Chart ——图表

- 统计分析结果

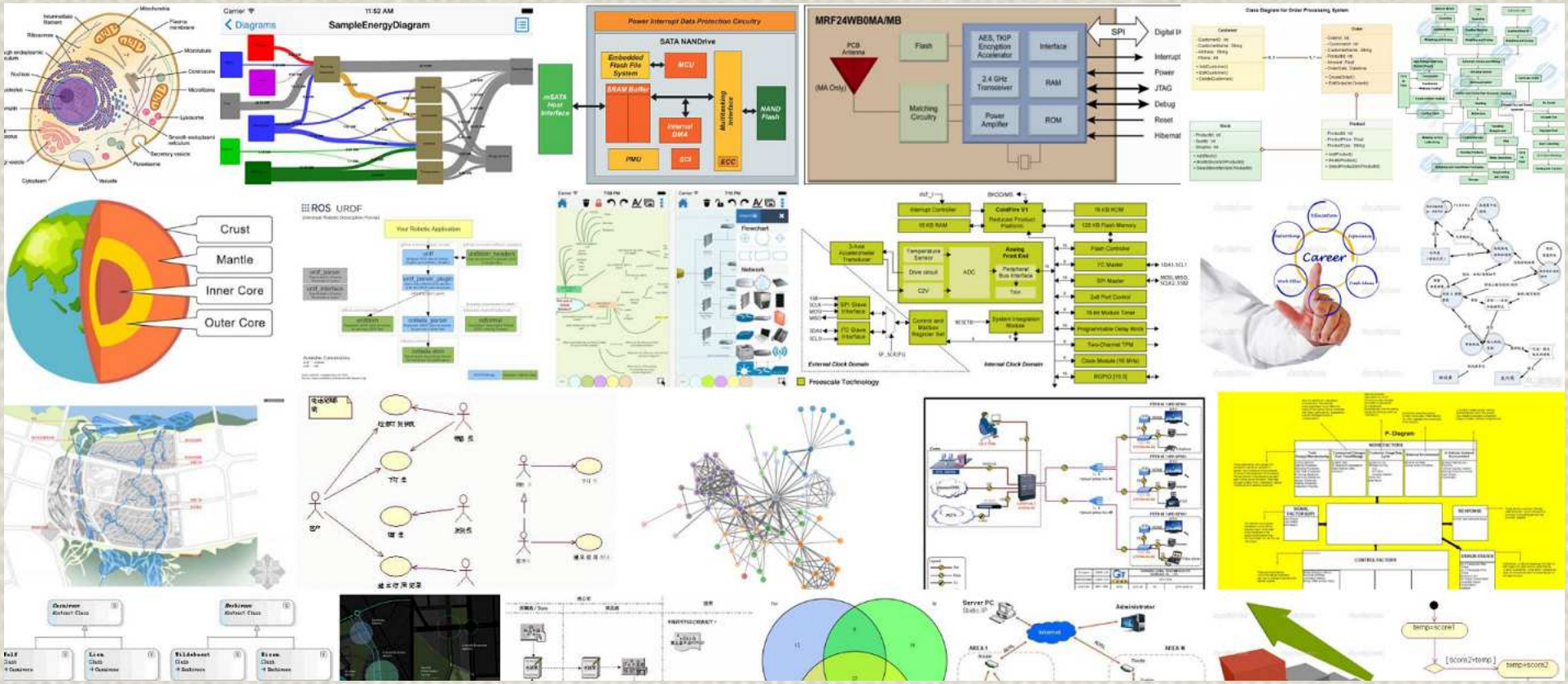



# Diagram

- A **diagram** is a symbolic representation of information according to some visualization technique.
- The word *graph* is sometimes used as a synonym for diagram.
- diagrams are pictorial, yet abstract, representations of information, and maps, line graphs, bar charts, engineering blueprints, and architects' sketches are all examples of diagrams, whereas photographs and video are not

# Diagram——图示


Crust
Mantle
Inner Core
Outer Core
SampleEnergyDiagram
Power Interrupt Data Protection Circuitry
SATA NANDrive
MRF24WB0MA/MB
ROS URDF
Career

# Figure & illustration

- There are 11 figures in my paper, and figure Fig.2 contains 10 illustrations.
- Fig.1 is a pie chart figure.

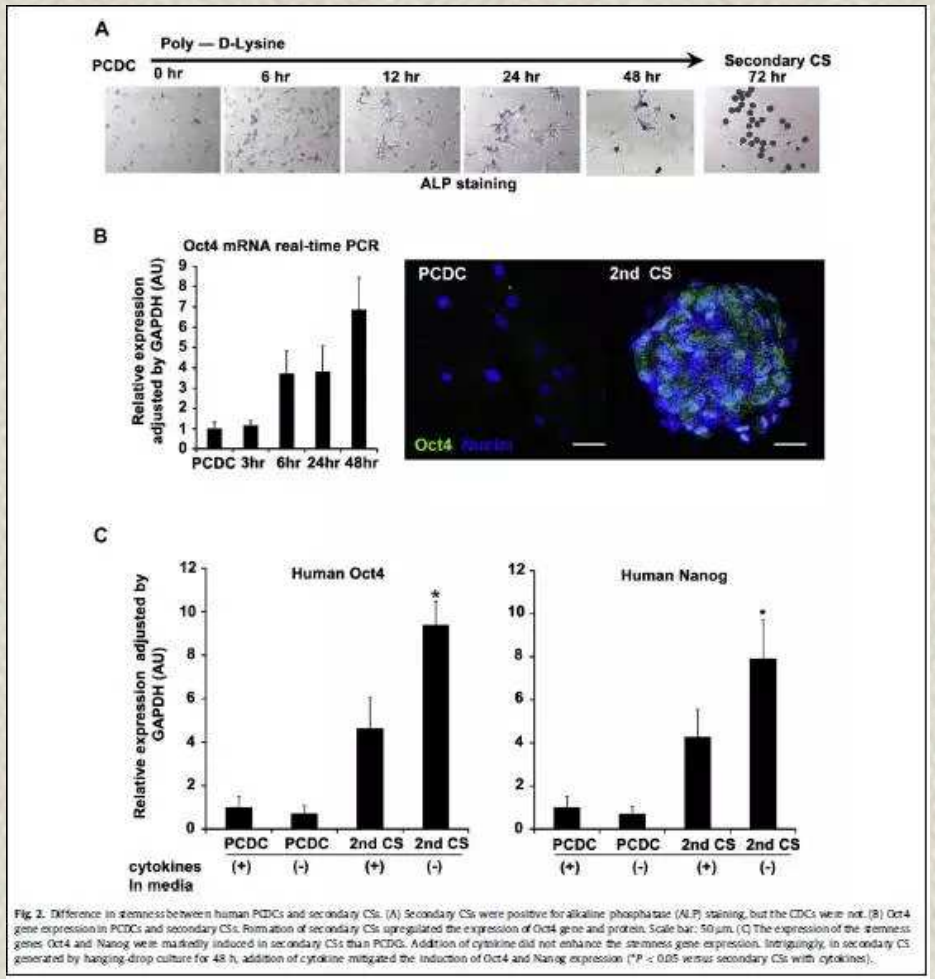


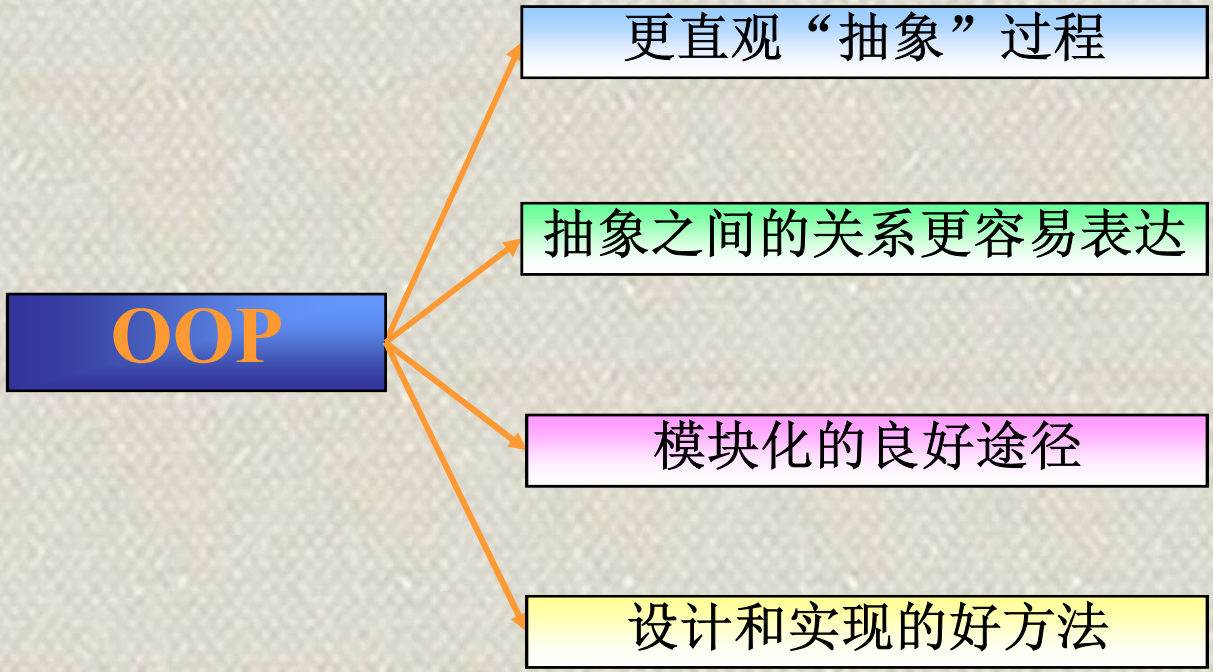

Fig 2. Difference in stemness between human PCDCs and secondary CSs. (A) Secondary CSs were positive for alkaline phosphatase (ALP) staining, but the CDCs were not. (B) Oct4 gene expression in PCDCs and secondary CSs. Formation of secondary CSs upregulated the expression of Oct4 gene and protein. Scale bar: 50 μm. (C) The expression of the stemness genes Oct4 and Nanog were markedly induced in secondary CSs than PCDCs. Addition of cytokine did not enhance the stemness gene expression. Intriguingly, in secondary CS generated by hanging-drop culture for 48 h, addition of cytokine mitigated the induction of Oct4 and Nanog expression (*$P < 0.05$ versus secondary CSs with cytokines).

# 特别说明

- 差别并不显著
- 上下文（ context）参照

# 目录

# 面向对象的编程思想



# 面向对象关键词

- 类
- 对象
- 封装
- 继承
- 多态

# 类

- 类是共性的抽象
- 共同的属性
- 共同的行为
- 访问控制

# 访问控制

- public
- protected
- private
- final
- static
- const/mutable

# 多态

- 编译时多态
- 运行时多态

# OOP & SP

- **Object Oriendted Programming**
- Procedure Oriented /Structured Programming

# 差别

| | SP | OOP |
|---|---|---|
| 设计思路 | 面向过程、注重效率 | 进一步抽象，注重封装 |
| 程序单元 | 结构、函数/子过程、指针<br>以函数调用为主体 | 对象、指针、引用<br>以对象为主体 |
| 设计方法 | 先解决关键问题 | 自顶向下、划分类与类的关系 |
| 用法 | **void use(something, way)** | **something.use(way)** |

# 哪个好？

- Kick(football); run(player)
- football.kick(); player.run()
- drink(coffee), coffee.drink()?
- cornerkick ?

# 注意

- 最终都要有过程实现
- 高度抽象化之后更利于“大家”理解
- 庞大的项目更容易规划

# 特别注意

- 我们曾经定义过一个类叫“Operation”
- 一类动作也可以被定义成“类”
- 动词的名词化

# 目录

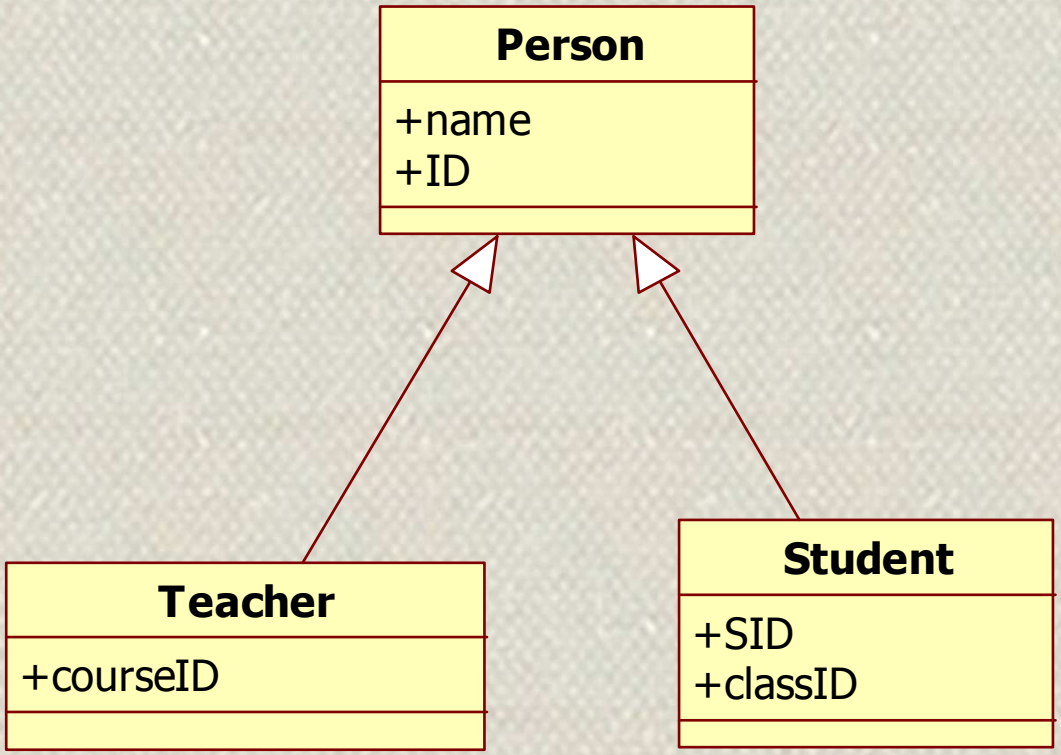
# 简要介绍UML图

- 用例图　需求过程
- 类图　　概要、详细设计
- 状态图　详细设计
- 时序图　详细设计
- ……

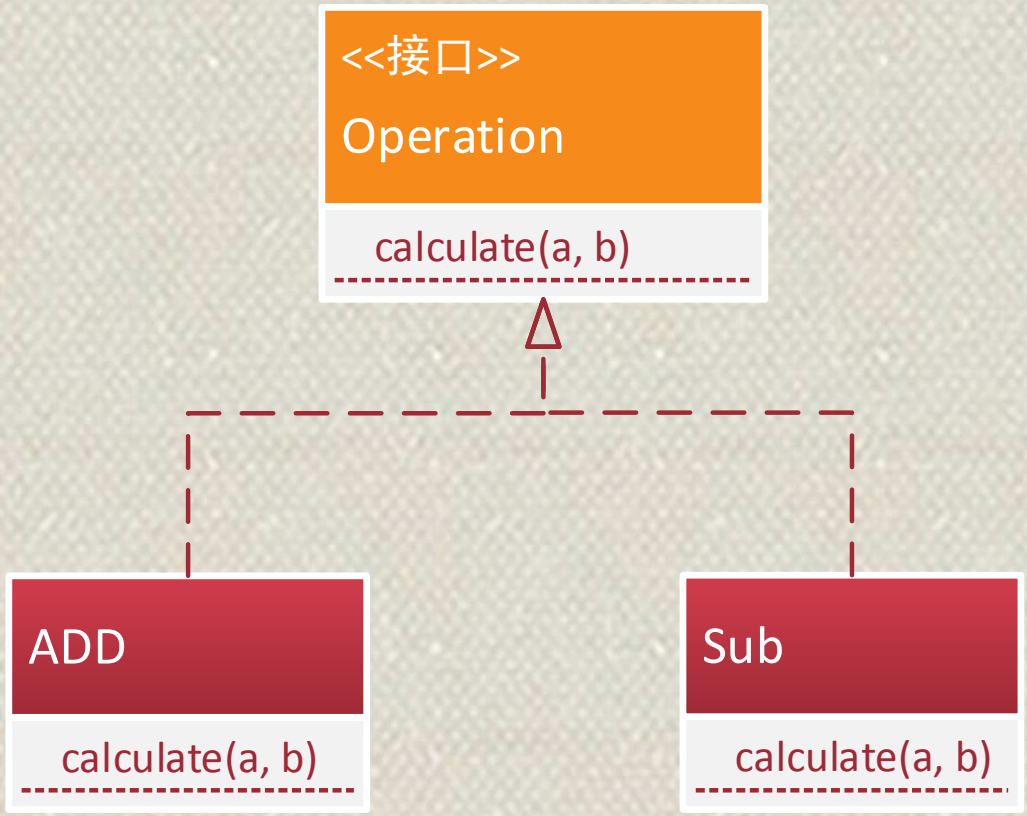
# UML类图

- 泛化（**Generalization**）
- 实现（**Realization**）
- 关联（**Association**）
- 聚合（**Aggregation**）
- 组合（**Composition**）
- 依赖（**Dependency**）

# 泛化



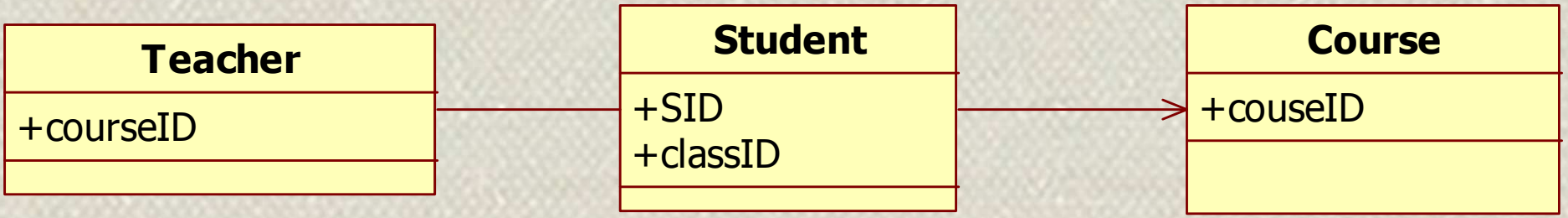
# 实现


<<接口>>
Operation
calculate(a, b)
ADD
calculate(a, b)
Sub
calculate(a, b)

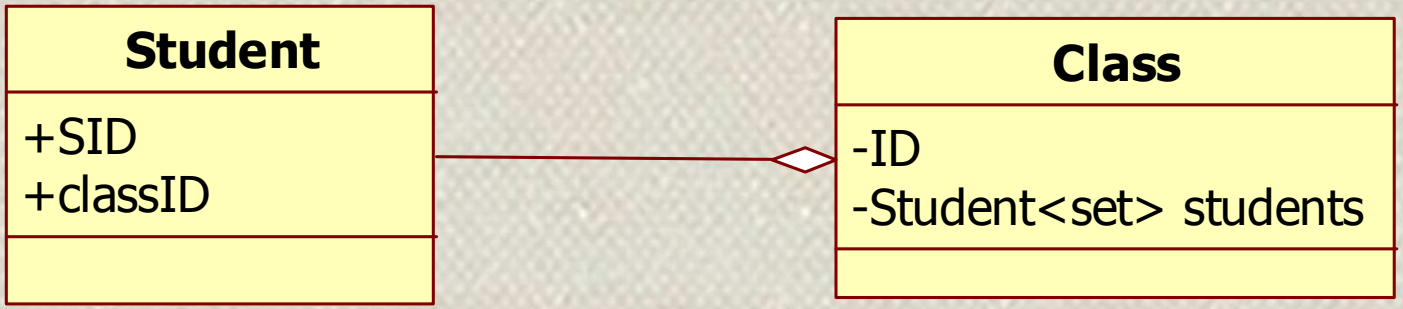
# Generalization OR Realization

- class
- interface / abstract class

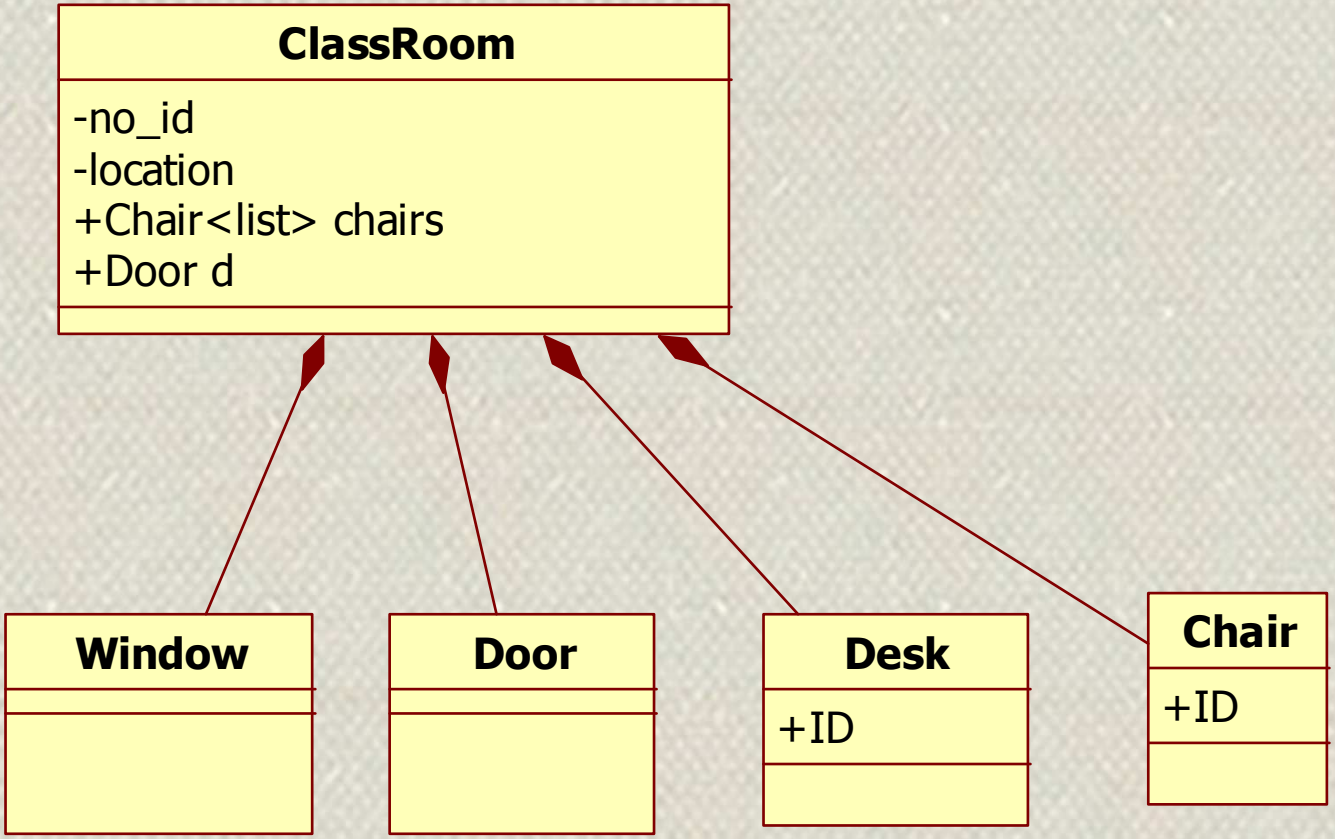
# 关联



# 聚合



# 组合


ClassRoom
-no_id
-location
+Chair<list> chairs
+Door d
Window
Door
Desk
+ID
Chair
+ID

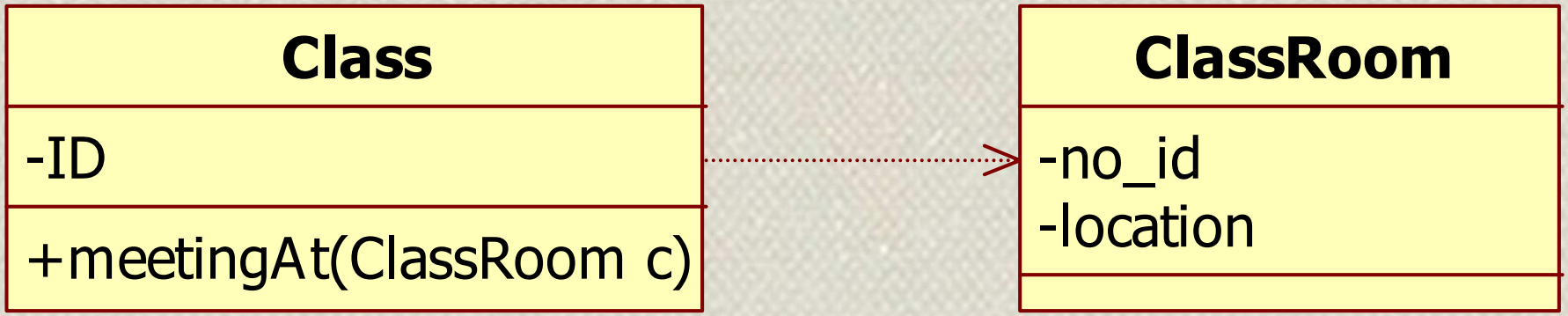
# 代码表达

```
class ClassRoom
{
        int roomID;
        Door front, back;
        BlackBoard mainboard
        list<Desk> desks;
};
```

# Aggregation OR Composition

- Member
- Component/Part

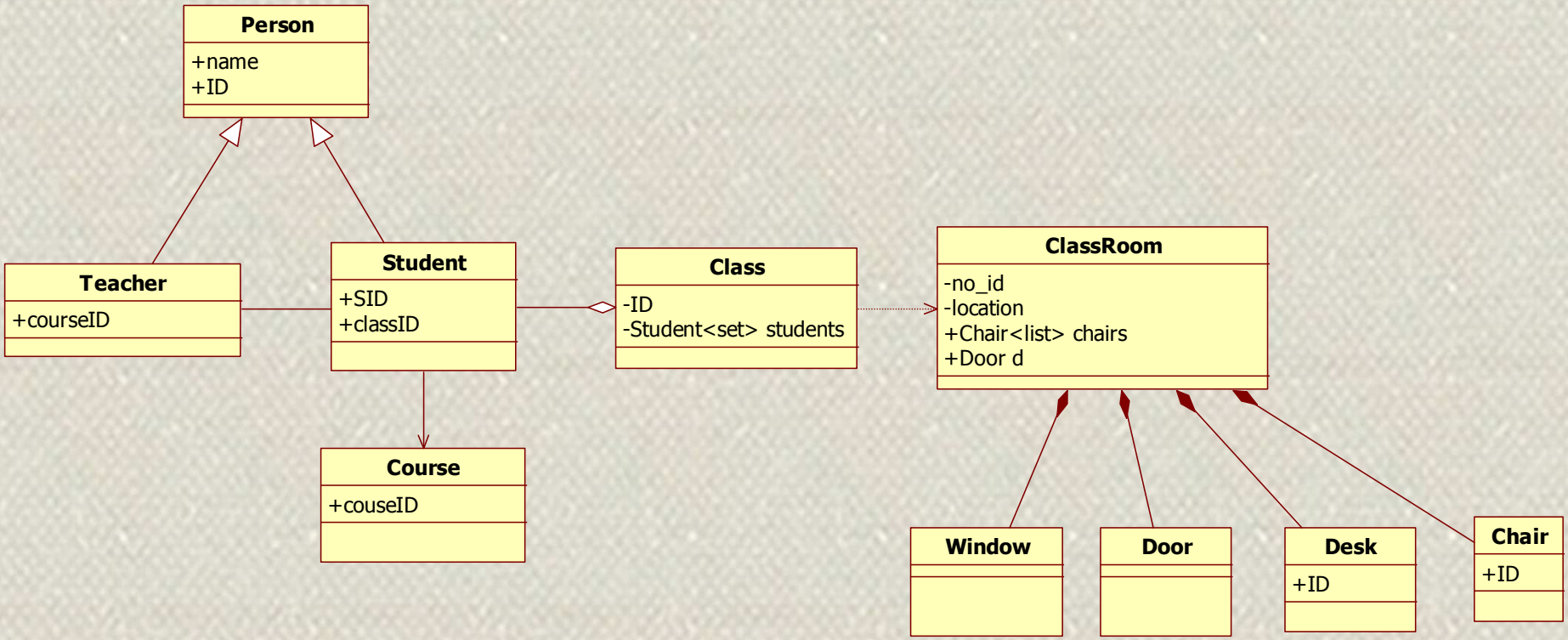
# 依赖

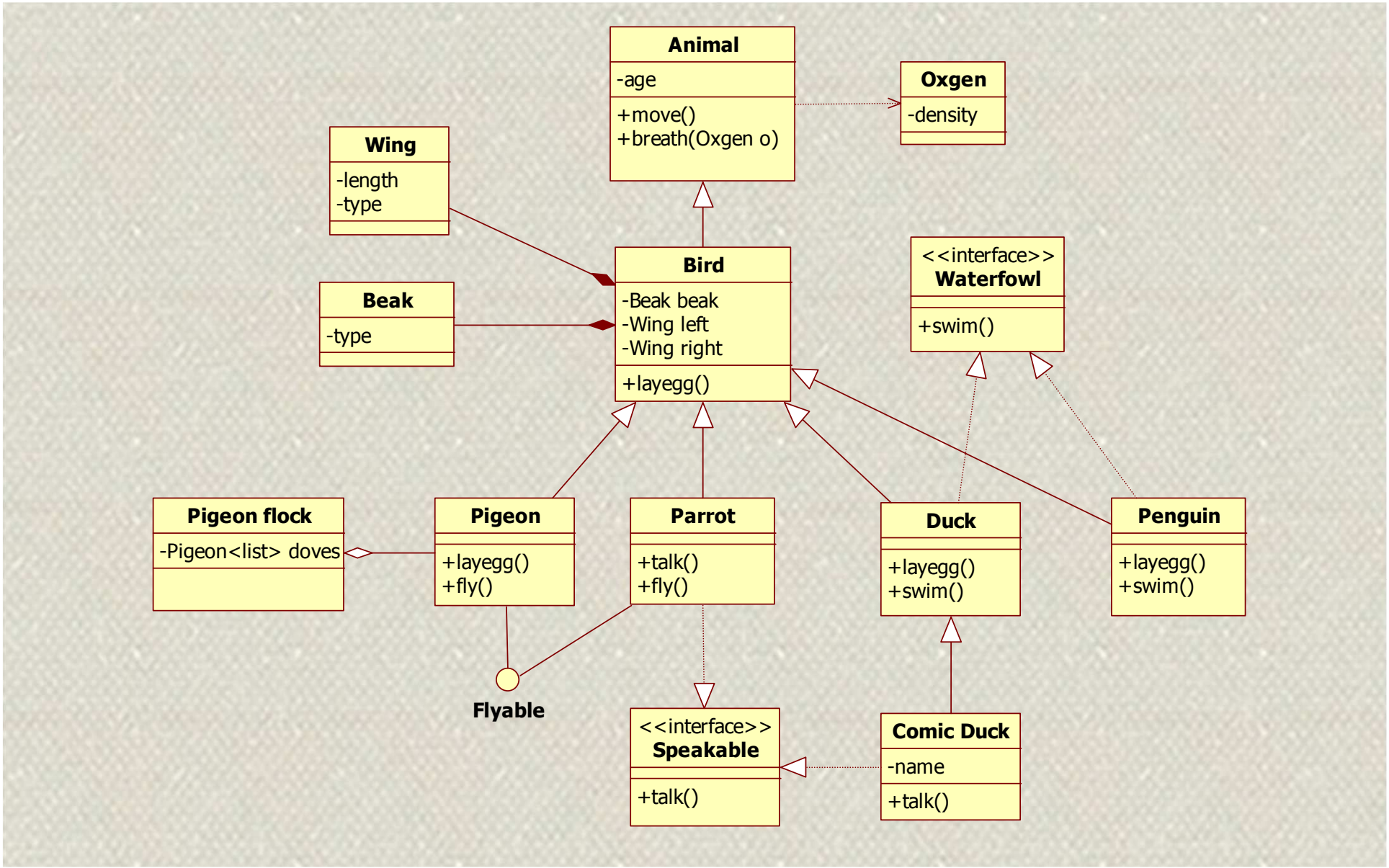


# Dependency OR Association

- Use or not

# 示例


Person
+name
+ID
Teacher
+courseID
Student
+SID
+classID
Class
-ID
-Student<set> students
ClassRoom
-no_id
-location
+Chair<list> chairs
+Door d
Course
+couseID
Window
Door
Desk
+ID
Chair
+ID


Animal
-age
+move()
+breath(Oxgen o)
Oxgen
-density
Wing
-length
-type
Beak
-type
Bird
-Beak beak
-Wing left
-Wing right
+layegg()
<<interface>>
Waterfowl
+swim()
Pigeon flock
-Pigeon<list> doves
Pigeon
+layegg()
+fly()
Parrot
+talk()
+fly()
Duck
+layegg()
+swim()
Penguin
+layegg()
+swim()
Flyable
<<interface>>
Speakable
+talk()
Comic Duck
-name
+talk()