面向对象基础

一、面向对象的本质

1、面向对象的世界观

世界观：是人们对世界的总的根本的看法。任何哲学问题的探讨，归其出发点和本源，都是世界观的问题。什么样的世界观决定了什么样的哲学观点。——马克思”

程序世界的两种对立。

过程论　　对象论

在软件开发设计方面不同的人也有着不同的世界观。而这其中最根本的对立便是过程论和对象论的对立，这个对立，衍生出了面向过程和面向对象两种方法论

注意：过程论和对象论，都承认程序世界本质上只有两种东西——数据和逻辑。数据天性喜静，构成了程序世界的本体和状态；逻辑天性好动，作用于数据，推动程序世界的演进和发展。

二、过程论的观点

1>数据和逻辑是分离的、独立的，各自形成程序世界的一个方面。所谓世界的演变，是在逻辑作用下，数据做改变的一个过程。
2>过程有明确的开始、结束、输入、输出，每个步骤有着严格的因果关系。
3>过程是相对稳定的、明确的和预定义的，小过程组合成大过程，大过程还可以组合成更大的过程。

结论：程序世界本质是过程，数据作为过程处理对象，逻辑作为过程的形式定义，世界就是各个过程不断进行的总体。

三、对象论的观点

1>数据和逻辑不是分离的，而是相互依存的。

2>相关的数据和逻辑形成个体，这些个体叫做对象，世界就是由一个个对象组成的。

3>对象具有相对独立性， 对外提供一定的服务。

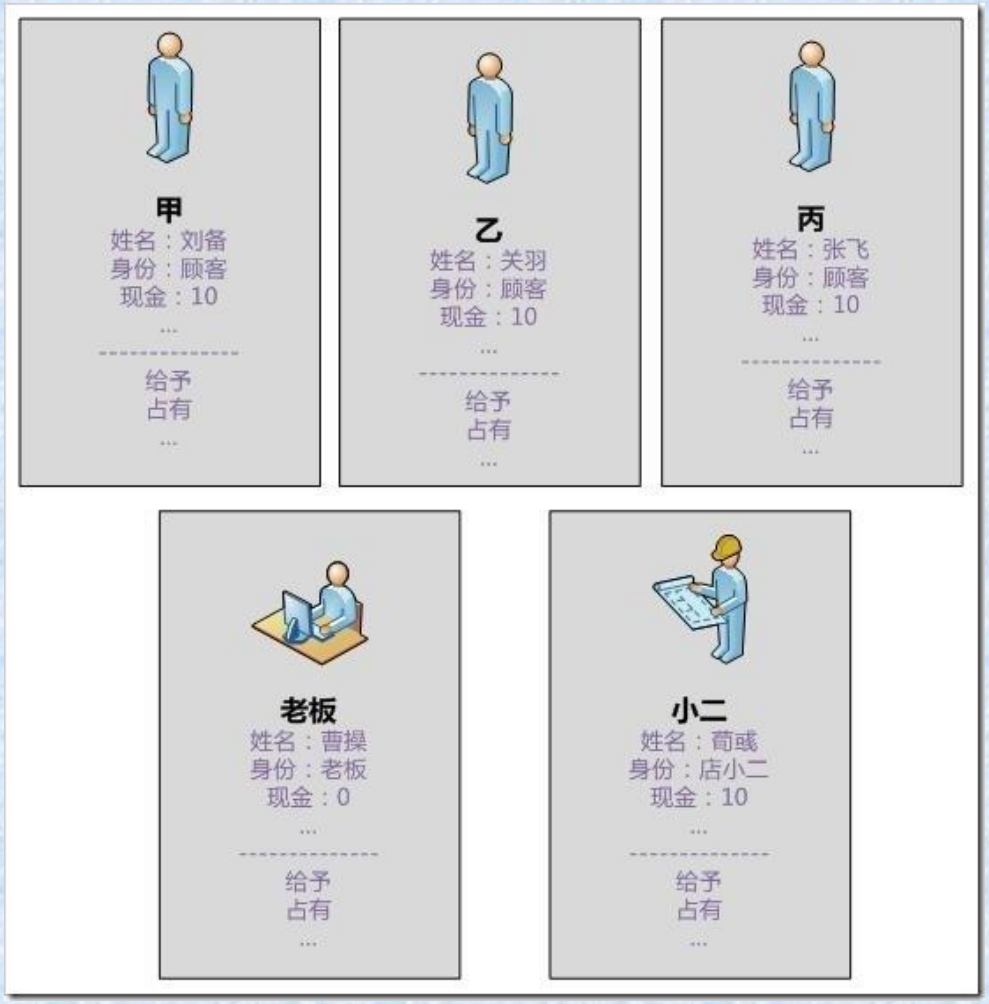
4>所谓世界的演进，是在某个“初始作用力”作用下，对象间相互调用而完成的交互，在没有初始作用力下，对象保持静止。

5>交互并不是完全预定义的，不一定有严格的因果关系，对象间交互是偶然的，对象间联系是暂时的。

结论：世界就是由各色对象组成，然后在初始作用力下，对象间的交互 完成了世界的演进。

问题：
有甲、乙、丙三人住店，一间房30。于是每人10元，共计给店老板30元住进一间房。后来店老板发现弄错了，房价应该是25元，于是给小二5元让小二退给 房客。小二黑心，贪污了2元，退给甲乙丙每人1元。这样房客每人付了10-1=9元，三九27，加上小二贪污的2元，共29元，问那1元哪里去了？

如何用对象论的观点解决以上问题？



对象论眼中，有五个基本对象，每个对象有自己的一系列数据和逻辑没错，在对象论眼里，这就是这件事的本质模样，这件事所涉及的东西就是这么几个对象。

本来它们各自独立，老死不相往来。只不过在住店这个外部驱动力下，几个对象偶然、暂时互相联系，利用其他对象提供的公开服务，完成了一些交互。

对应考虑面向对象分析的模型

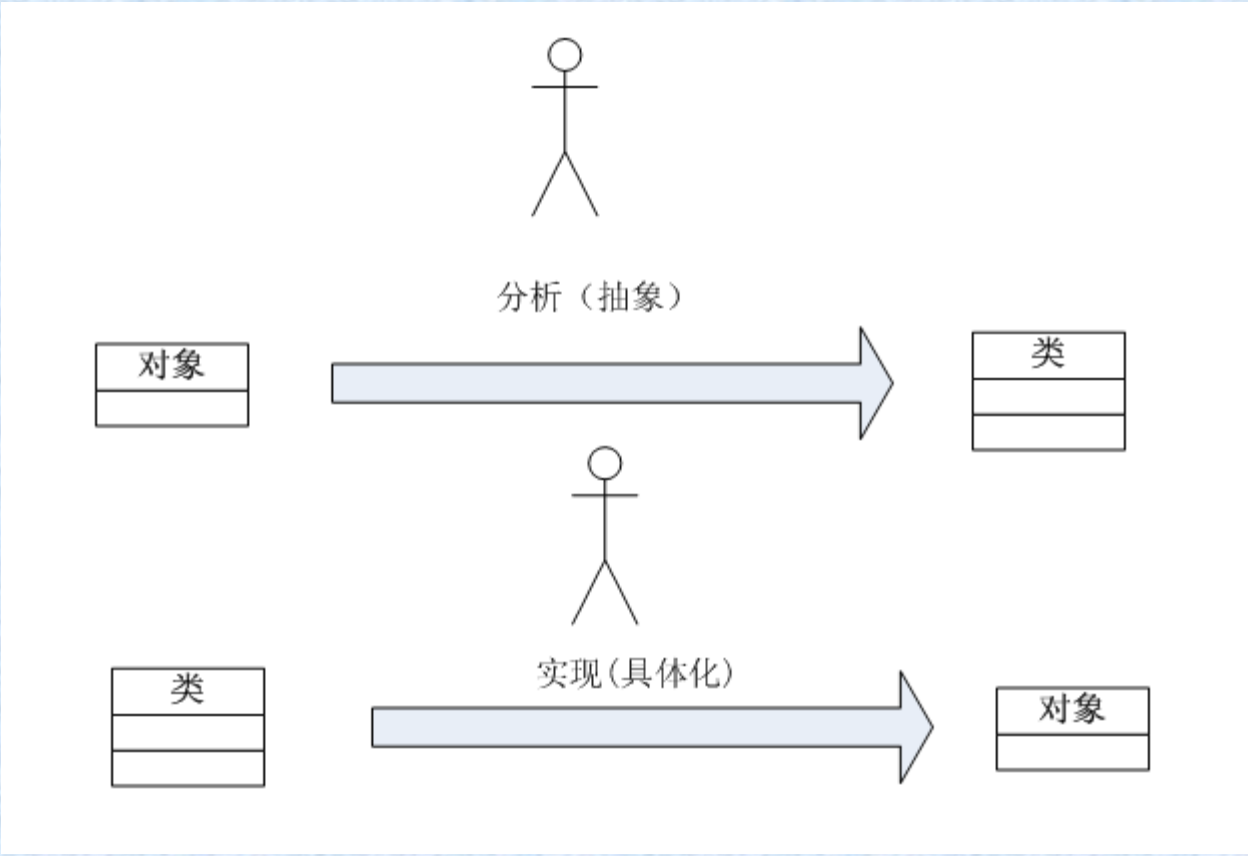
对象论观点

2、抽象

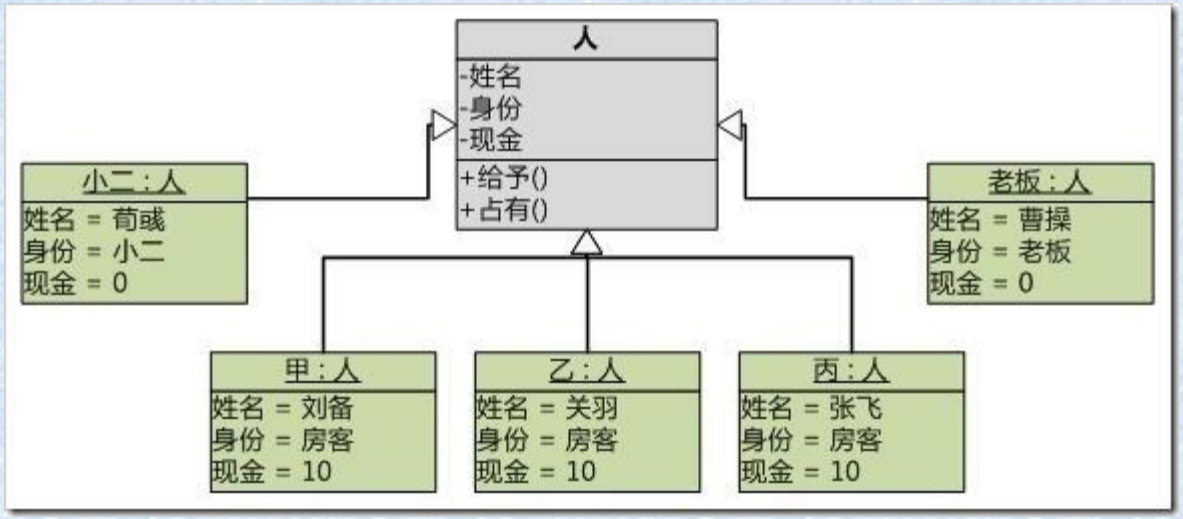
先有对象还是先有类？

从哲学角度说，先有对象，然后才有类，类和对象是“一般和特殊”这一哲学原理在程序世界中的具体体现

类其实是抽象认知能力作用于程序世界的基本元素——对象后所衍生出来的抽象概念，是抽象思维在程序世界中物化后的产物。现实世界中每个对象都有无数的数据和逻辑，但在具体到程序世界时，我们往往只关心具体场景中相关的数据和逻辑。


分析（抽象）
对象
类
实现(具体化)
类
对象

上面的例子中五个对象很相似，可以看做一类东西，于是，我们给出一个类，叫“人”，并且认为这五个对象都是“人”这个类的具体例子，我们叫其为实例。以后遇到类似的对象，我们都可以知道，这个对象属于“人”类。



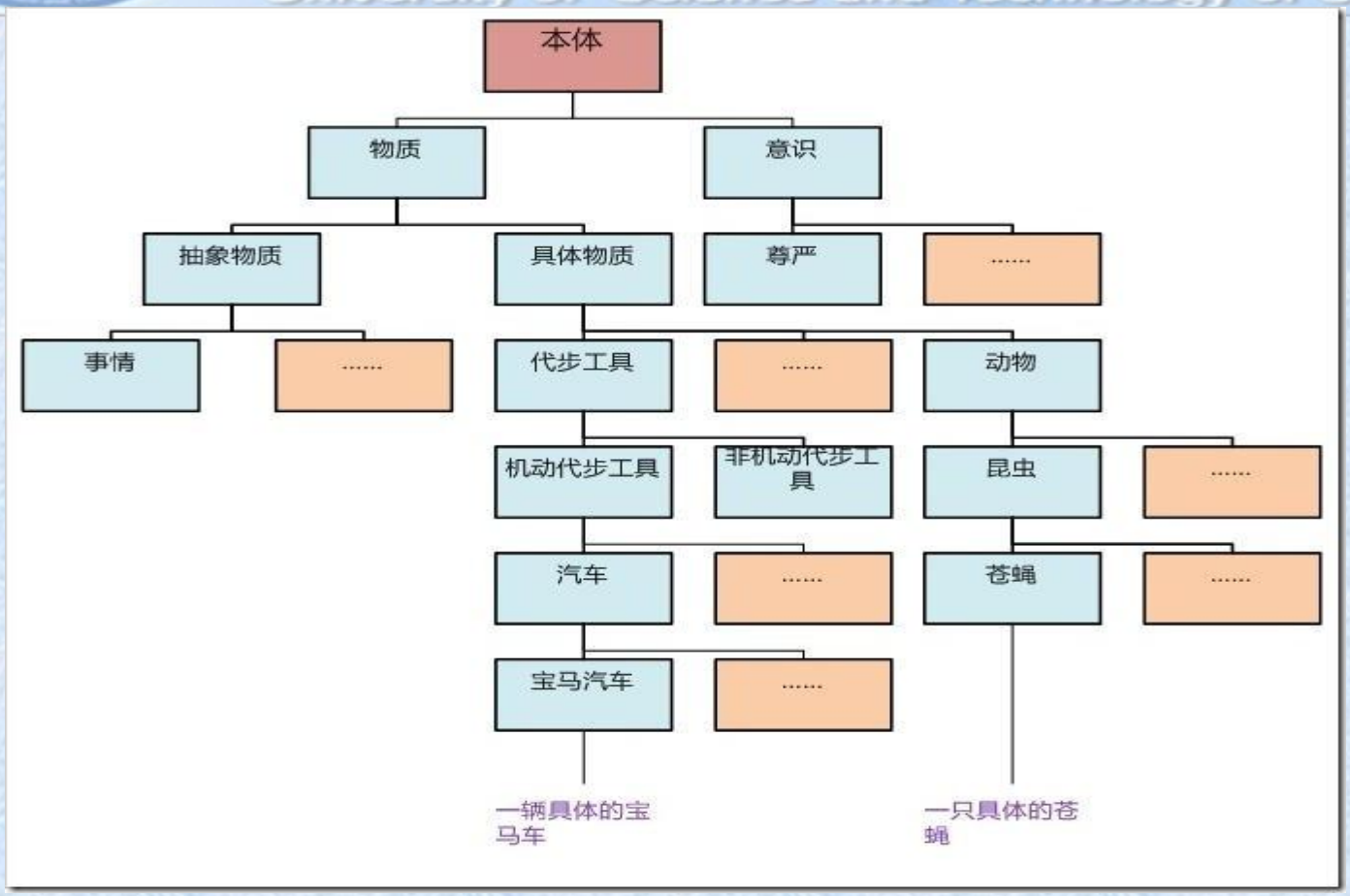
3、层次

“道生一，一生二，二生三，三生万物—老子”

抽象是有层次的

世界可并不是一层对象一层类那么简单，对象抽象出类，在类的基础上可以再进行抽象，抽象出更高层次的类。所以经过抽象的对象论世界，形成了一个树状结构

抽象层次树



本体即万物之源、万物之本，是哲学层面上最高层次的抽象

注意：

抽象层次树理论也是OO中许多内容的理论基础。

Eg:OO中重要的概念——继承和多态，如若探究其哲学本源，就是从这里来的。

- 这是一棵单根树，最顶层“本体”为唯一的根，最下层叶子节点为基本对象。一切中间节点都为类。
- 越往上的类抽象层次越高，具体度越低，其内涵越小，外延越大；越往下的类抽象层次越低，具体度越高，其内涵越大，外延越小

  内涵:指 类对属于自己的对象的说明力度

  外延:指类能包含的具体对象的总和。
- 抽象层次树不是从根部向下长的，而是从叶子节点向上归纳生成的。
- 某一个叶子节点所代表的对象可以归入所有其祖先结点所代表的类
- 直接问两个叶子节点属不属于一个类没有意义，而要指定抽象层次才有意义。例如在较低层，一辆宝马属于汽车，而一只苍蝇属于昆虫，不是一类。但如果指定在较高层比较，两个都属于具体物质，属于一个类。

- 我们定义，如果一个节点CNode非叶子节点也非根节点，那么在哲学意义上，这个节点继承于其父节点PNode，并且说PNode是CNode的泛化。
- 我们定义，如果一个节点CNode非叶子节点也非根节点，如果强行将它看成其任何一个祖先节点ANode，并当做ANode使用，那么在哲学意义上，叫做多态性

4、继承与泛化

抽象层次树上，除根节点和叶子节点外，任一节点CNode非严格继承其所有祖先节点所组成的集合中的任一元 素，而CNode严格继承其父节点PNode。泛化与继承相反。

注意：从哲学和认识论角度来说，是先有对象，然后有类；先有子类，然后有父类，是一种自底向上形成的体系。 而继承一词，就容易让人误解成是先有父类才有子类。所以推荐使用泛化来理解层次树上的关系

5、耦合

“一只蝴蝶在巴西轻拍翅膀，可以导致一个月后德克萨斯州的一场龙卷风——蝴蝶效应”

普遍联系： 世界上各种对象形成了一张复杂的耦合网，正因为有耦合的存在，世界才能演进。正如马克思主义哲学所说：联系是普遍的、客观的。所以，耦合的存在，有其深刻的哲学意义。

什么是耦合？

可以看成联系，耦合的存在是世界演进的途径，如果没有耦合，世界就变成了“死世界”，无法演进和发展。所以，世界需要耦合 。我们在开发设计中要降低耦合，但并不是要取消耦合

类之间的耦合关系

泛化耦合：

由于泛化（继承）关系的存在，在两个有祖孙、父子关系的类间形成的一种逻辑关联。

聚合：

一种弱的拥有关系，体现A对象可以包含B对象，但B对象不是A对象的一部分。

组合：

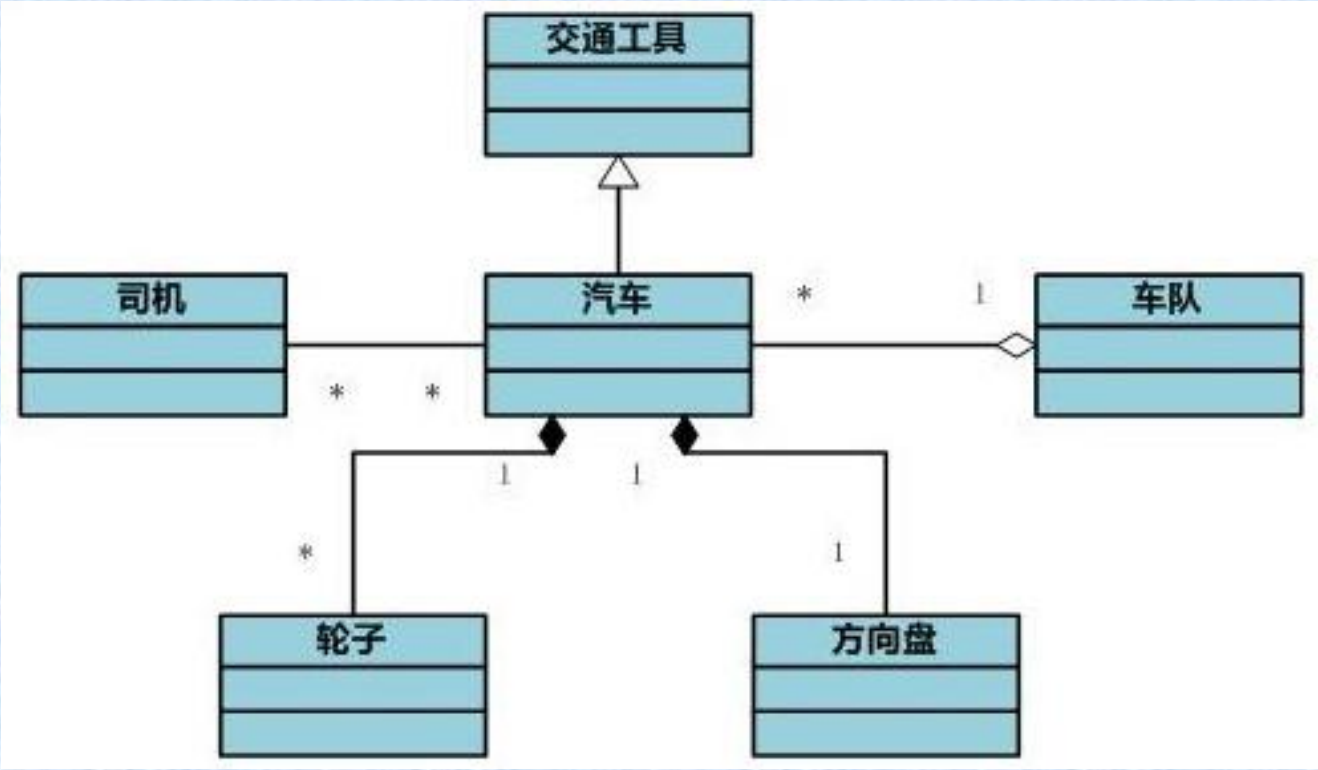
一种强的拥有关系，体现了严格的部分和整体的关系，部分和整体具有一样的生命周期。

依赖：

由于逻辑上相互协作可能，而形成的一种关系。

对象间的耦合

利用消息实现耦合



其中汽车和交通工具属于泛化耦合，轮子和方向盘组合于汽车，汽车聚合成车队，而汽车和司机具有依赖关系。

二、OO的具体概念

- 客观世界中的应用问题都是由实体及其相互关系构成的。
- 可以将客观世界中与应用问题有关的实体及其属性抽象为问题空间中的对象。
- 为应用问题寻求软件解，是借助于计算机语言对其提供的实体施加某些动作，以动作的结果给出问题的解。
- 面向对象方法包括：分析、设计和实现，它们是一脉相承的。

面向对象的方法是一种运用对象、类、继承、封装、聚合、消息传送、多态性等概念来构造系统的软件开发方法。

面向对象=对象+类+继承（泛化）+消息通信

可以说，采用这四个概念开发的软件系统是面向对象的。

面向对象方法成为主流开发方法。可以从下列几个方面来分析其原因：

- 从认知学的角度来看，面向对象方法符合人们对客观世界的认识规律。
- 面向对象方法开发的软件系统易于维护，其体系结构易于理解、扩充和修改。
- 面向对象方法中的继承机制有力支持软件的复用。

注意：并没有摒弃面向过程的方法

1. 对象（object）

具体到OO程序设计上，对象是指一组属性以及这组属性上的专用方法的封装体。

属性（attribute）通常是一些数据，有时它也可以是另一个对象。每个对象都有它自己的属性值，表示该对象的状态。对象中的属性只能通过该对象所提供的操作来存取或修改。

方法（method）规定了对象的行为，表示对象所能提供的服务。

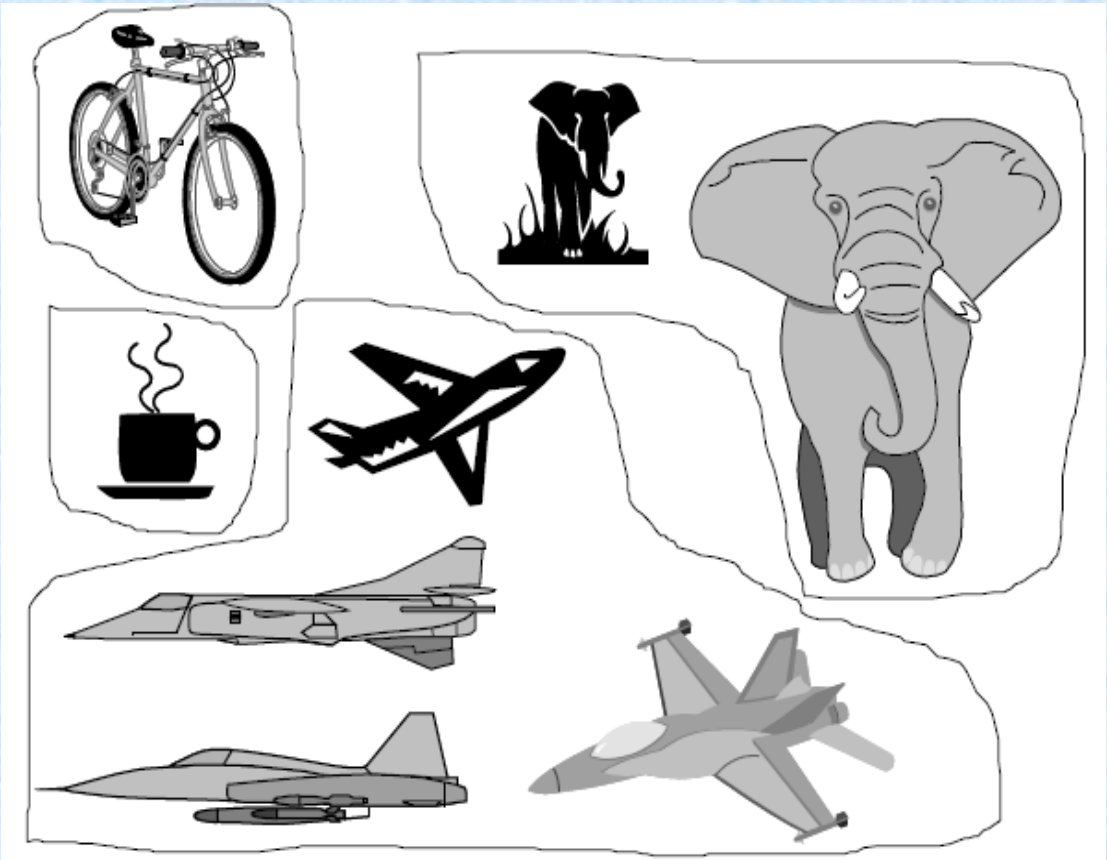
封装（encapsulation）是一种信息隐蔽技术，用户只能看见对象封装界面上的信息，对象的内部实现对用户是隐蔽的。

封装的目的是使对象的使用者和生产者分离，使对象的定义和实现分开。

2. 类（class）

具体到OO程序设计上，类可以看做是类是一组具有相同属性和相同操作的对象的集合。一个类中的每个对象都是这个类的一个实例（instance）。

类是创建对象的模板，从同一个类实例化的每个对象都具有相同的结构和行为。



Elephant
Color : text Number of tusks : Integer Location: text Weight: float Height: float
move_to (location) wash (date) feed (amount, date, time)

对象和类的描述

对象和类一般采用“对象图” 和 “类图” 来描述。

类图

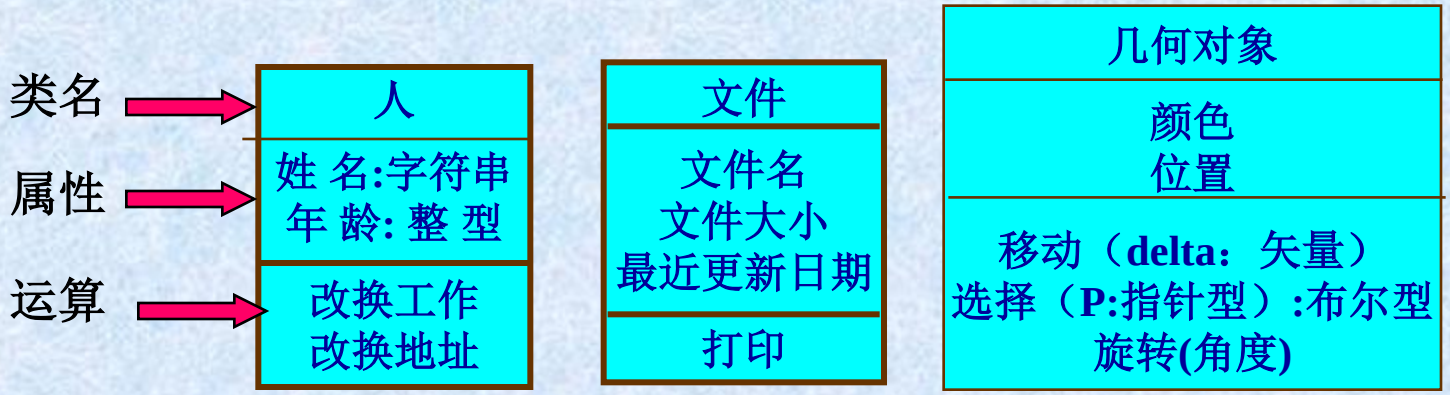
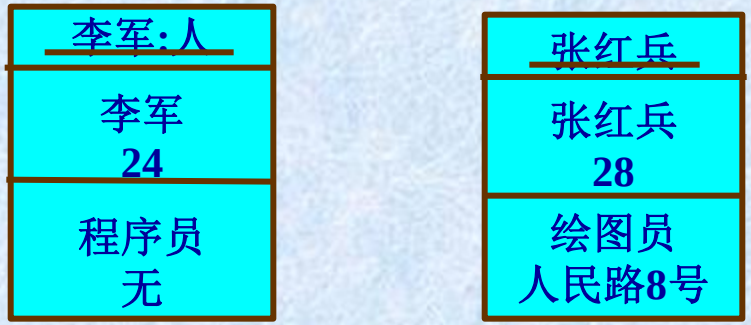


图 对象类的描述

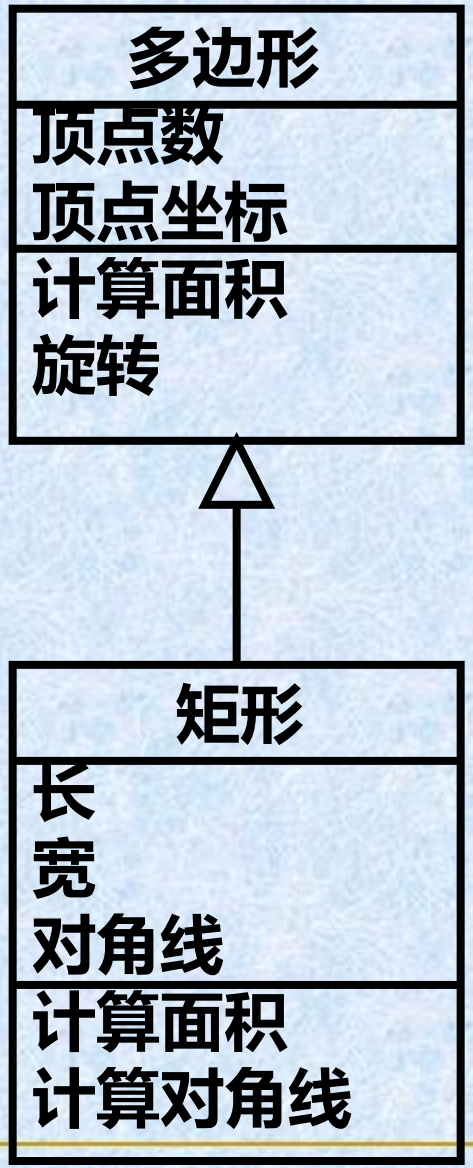
对象图



类　　实例对象

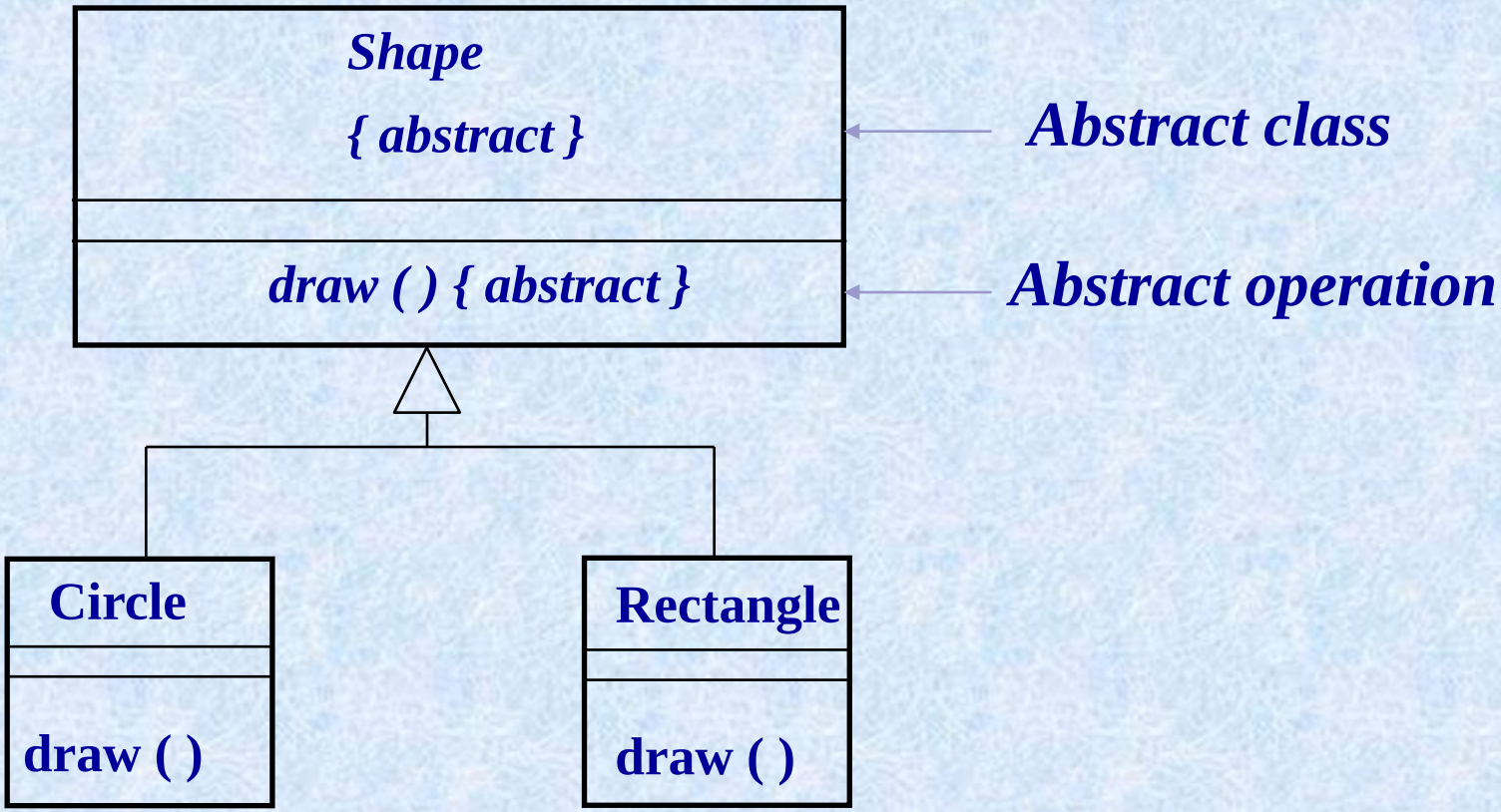
3.继承（inheritance）

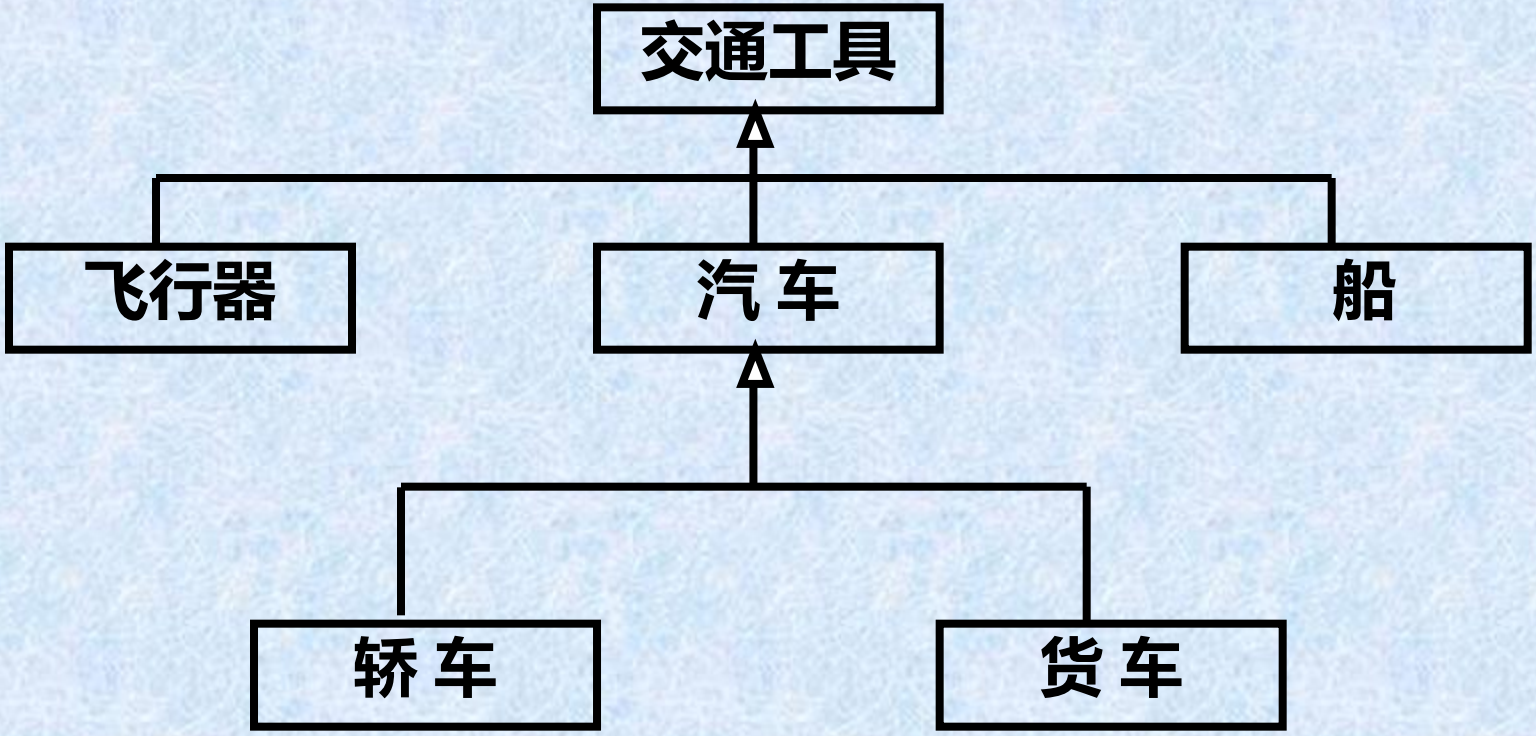
具体到OO程序设计上，继承指出了是类间的一种基本关系，它是基于层次关系的不同类共享数据和操作的一种机制。父类中定义了其所有子类的公共属性和操作，在子类中除了定义自己特有的属性和操作外，可以继承其父类（或祖先类）的属性和操作，还可以对父类（或祖先类）中的操作重新定义其实现方法。


多边形
顶点数
顶点坐标
计算面积
旋转
矩形
长
宽
对角线
计算面积
计算对角线

抽象类(abstract class)：没有实例的类，它把一些类组织起来，提供一些公共的行为，但并不需要使用这个类的实例，而仅使用其子类的实例。

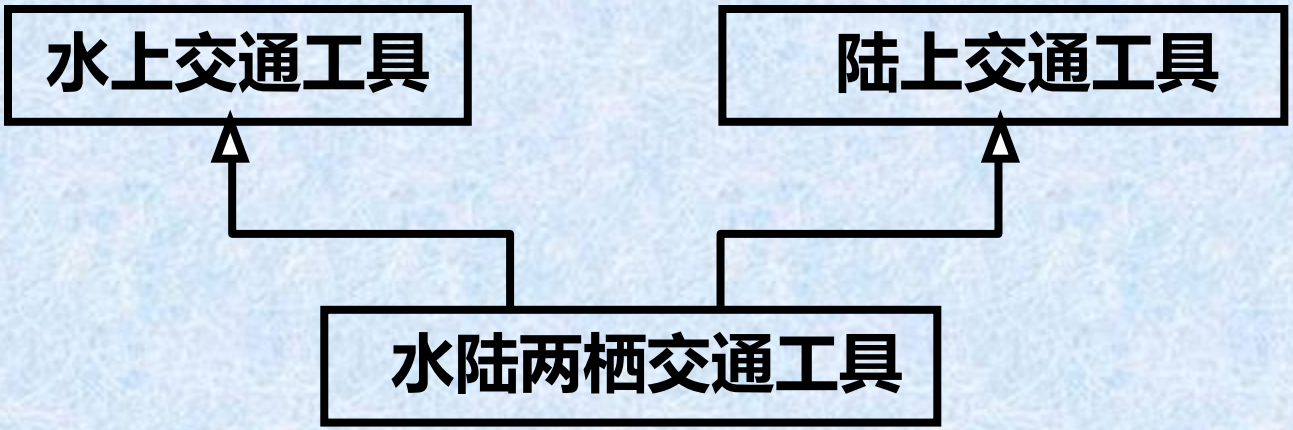
在抽象类中可以定义抽象操作，抽象操作指：只定义这个类的操作接口，不定义它的实现，其实现部分由其子类定义。抽象操作操作名用斜体字表示，也可以在操作特征（signature）后面加上特征字符串{abstract}。


Shape
{ abstract }
draw ( ) { abstract }
Abstract class
Abstract operation
Circle
draw ( )
Rectangle
draw ( )



一般-特殊关系

如果一个子类只有唯一一个父类，这个继承称为单一继承。如果一个子类有一个以上的父类，这种继承称为多重继承。（Java能否多继承？OC呢？）



多重继承

4.消息，方法，属性

- 消息就是某个操作的规格说明，其组成：
  - ○ 接收消息的对象
  - ○消息名（消息选择符）
  - ○零个、一个或多个数据

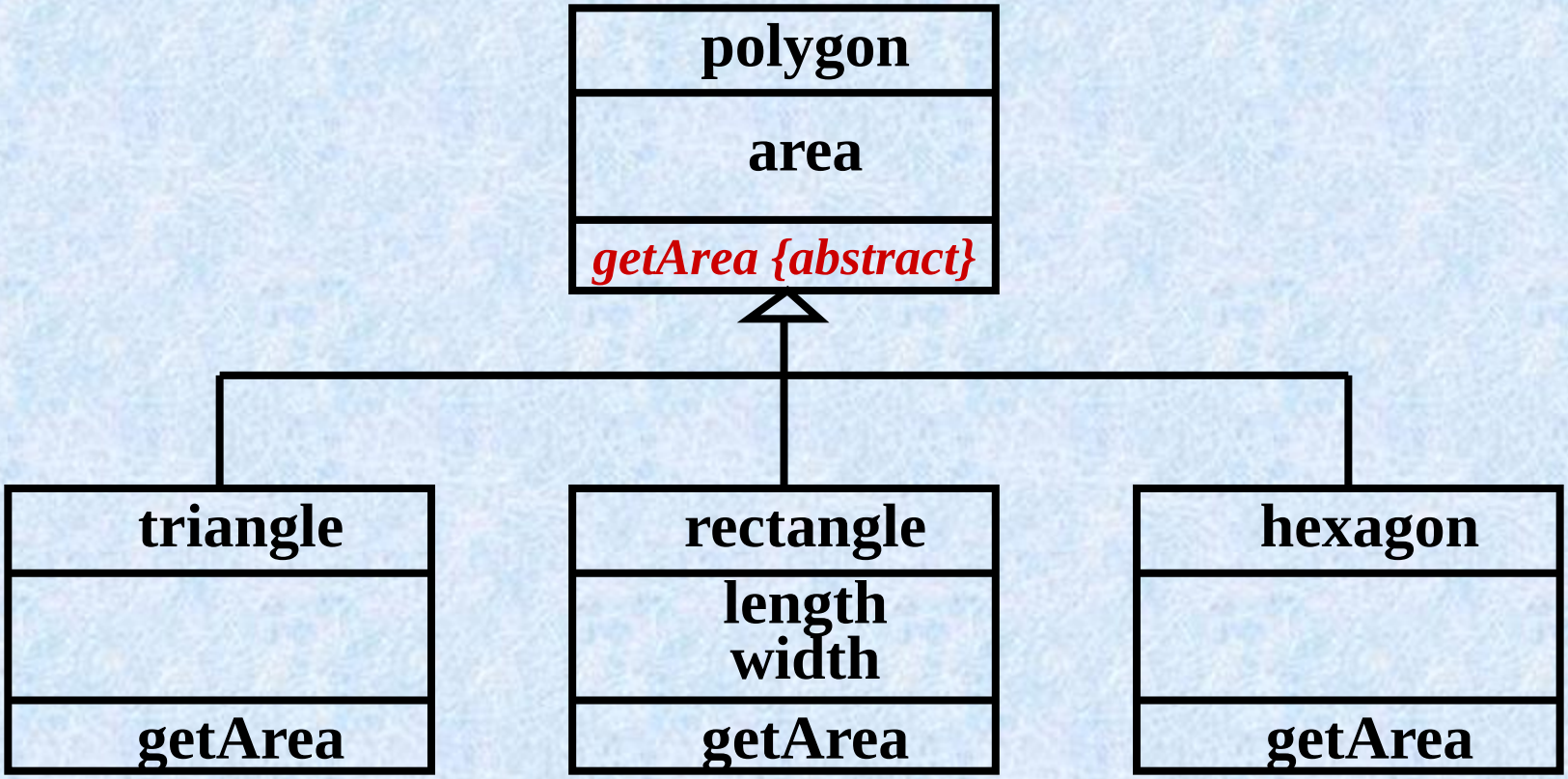
  例如：对于类Circle的一个实例MyCircle，如果使其以绿色在屏幕上显示，MyCircle . Show (GREEN);
- 方法（操作、服务）
  - ○对象所能执行的操作，即类中所定义的服务。它是对操作算法和响应消息办法的描述。
  - ○在类Circle中给出成员函数Show (int color)的定义。
- 属性是类中所定义的数据，是实体性质的抽象
  - ○类实例都有其特有的属性值，如类Circle定义的圆心、半径和颜色。

5. 多态性与动态绑定

具体到OO程序设计上， 多态性是指同一个操作作用于不同的对象上可以有不同的解释，并产生不同的执行结果。

例如“画”操作，作用在“矩形”对象上，则在屏幕上画一个矩形，作用在“圆”对象上，则在屏幕上画一个圆。

在一般与特殊关系中，子类是父类的一个特例，所以父类对象可以出现的地方，也允许其子类对象出现。因此在运行过程中，当一个对象发送消息请求服务时，要根据接收对象的具体情况将请求的操作与实现的方法进行连接，即**动态绑定**。


polygon
area
getArea {abstract}
triangle
getArea
rectangle
length
width
getArea
hexagon
getArea

6、永久对象(Persistent object)

所谓永久对象是指生存期可以超越程序的执行时间而长期存在的对象。

目前，大多数OOPL不支持永久对象，如果一个对象要长期保存，必须依靠于文件系统或数据库管理系统实现，程序员需要作对象与文件系统或数据库之间数据格式的转换，以及保存和恢复所需的操作等烦琐的工作。