

面向对象技术

内容安排

- 面向对象技术概述
- 面向对象技术发展
- 面向对象与结构化
- 上升到面向对象

面向对象技术

- 是一种看待计算机软件系统的观点
- 是一种系统分析和设计的思想
- 是一种编程方法
- 是一组设计模式
- 是一种编程语言设计思路
- 是实践者的日常工作

面向对象技术定义

面向对象技术基于对象概念，以对象为中心，以类和继承为构造机制，充分利用接口和多态提供灵活性，来认识、理解、刻画客观世界和设计、构建相应的软件系统

面向对象 = 对象 + 类 + 消息 + 继承 + 多态

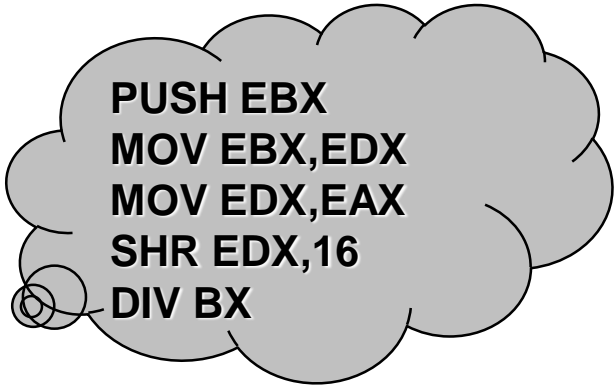
- **面向对象方法**是一种把面向对象的思想运用于软件开发过程，指导开发活动的系统方法，包括**分析、设计和实现**等活动

- 1.与人类习惯的思维方法一致。
- 2.稳定性好。
- 3.可重用性好。
- 4.较易开发大型软件产品。
- 5.可维护性好


面向对象技术利益-1

- 沟通

- 顺应人类思维习惯，让软件开发人员在解空间中直接模拟问题空间中的对象及其行为



```
PUSH EBX  
MOV EBX,EDX  
MOV EDX,EAX  
SHR EDX,16  
DIV BX
```



```
AHare.Run;  
ALion.Catch(AHare)  
;  
ALion.Kill(AHare);  
AHare.Dead;  
ALion.Eat;  
ALion.Happy;
```

在计算机中模拟现实世界的事和物

实例01：“东北一家人？”

- 东北人都是活雷锋
 - 人、东北人、雷锋
- 老张开车去东北…… 撞啦！
 - 老张、汽车、开车
 - 撞啦

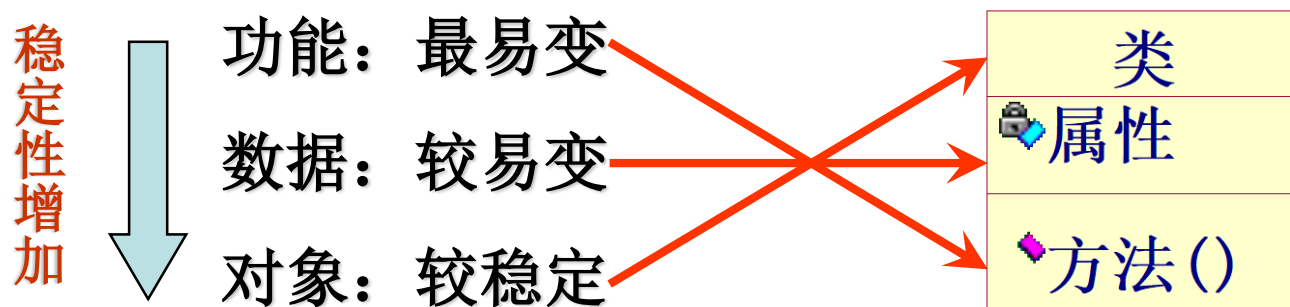
面向对象的表示

```
class 人 {  
    Region 籍贯;  
}  
  
class Region { }  
  
interface 雷锋 {  
    helpPeople(){ }  
}  
  
class 东北人 extends 人  
implements 雷锋 {  
    籍贯 = 东北;  
    helpPeople(){ }  
}
```

```
class Car{  
    DriveTo(Region) throws  
    Exception (撞车){}  
    人 Driver;  
}  
  
Main Program {  
    人 老张;  
    Car 夏利;  
    夏利.Driver = 老张;  
    try {  
        夏利.DriveTo(东北);  
    } catch (Exception) { }  
}
```


面向对象技术利益-2

- 稳定
 - 较小的需求变化不会导致系统结构大的改变
 - 当需求变化时.....



用较稳定把不稳定的包起来

面向对象技术利益-3

- 复用
 - 代码重用：类库、框架等重用机制
 - 能提高质量，减少由于编制新的系统代码而产生的成本
 - 通过继承、关联、封装等手段

面向对象技术利益-4

- 改善软件结构（模块化与封装），提高软件灵活性
- 增加可扩展性
- 支持增量式开发，支持大型软件开发
- ...

内容安排

- 面向对象技术概述
- 面向对象技术发展
- 面向对象与结构化
- 上升到面向对象

面向对象技术发展-1

- 里程碑1： Simula 67
(1962-1967, 挪威)
 - Ole-Johan Dahl和
Kristen Nygaard 在挪
威奥斯陆国家计算中心
(NCC) 设计实现. 公认
的世界上第一种面向对
象语言
 - 基本思想

The word "Simula" is rendered in a 3D, red, blocky font with a slight shadow underneath, giving it a three-dimensional appearance.

面向对象技术发展-2

- 里程碑2:
Smalltalk(1970, 施乐
保罗阿托)
 - Alan Kay设计实现
 - 第一个成熟的面向对象语言, 为开发GUI而设计
 - 实用化



Smalltalk
all: objects
all: theTime

预言历史的最佳方式是创造历史

-- Alan Kay

面向对象技术发展-3

- 里程碑3: ADT, Ada 83, 基于对象(1977-1983)
 - 编程理论界在结构化运动中提出ADT思想, 以N. Wirth和Liskov为代表
 - 美国防部军用开发语言评选, Ada被指定为强制性军用编程语言, 实际上已经进入“基于对象”阶段



Ada Lovelace是英国著名诗人拜伦的女儿, 世界上第一个程序员。**Ada**语言以她的名字命名

面向对象技术发展-4

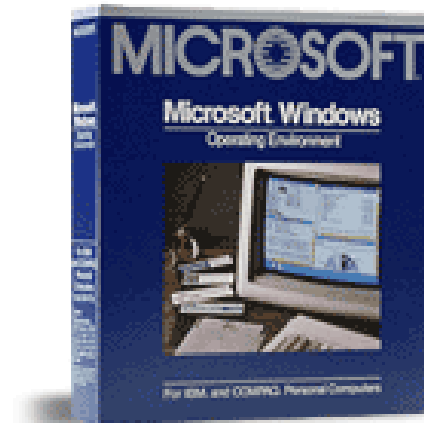
- 里程碑4：C++ (1982, 贝尔实验室)
 - 第一个被工业界广泛接受的支持面向对象能力的语言，创造者Bjarne Stroustrup
 - 动机是给C添加一些Simula特性，以完成当时刚刚出现的大规模复杂任务
 - 商业化



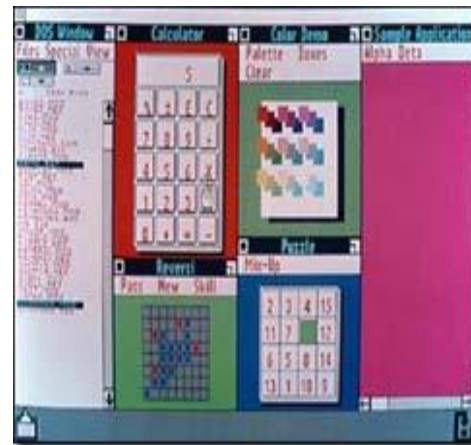
Bjarne Stroustrup

面向对象技术发展-5

- 里程碑5: MS-Windows(1985, 微软)
 - 第一个被广泛使用的GUI系统软件, 它使面向对象技术的使用不可阻遏



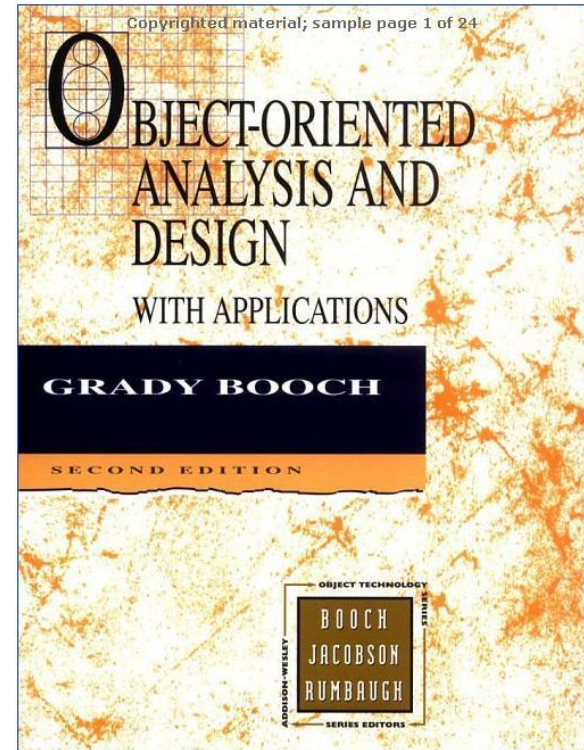
1985年出品的Windows 1.0产品



1987年出品的Windows 2.0界面

面向对象技术发展-6

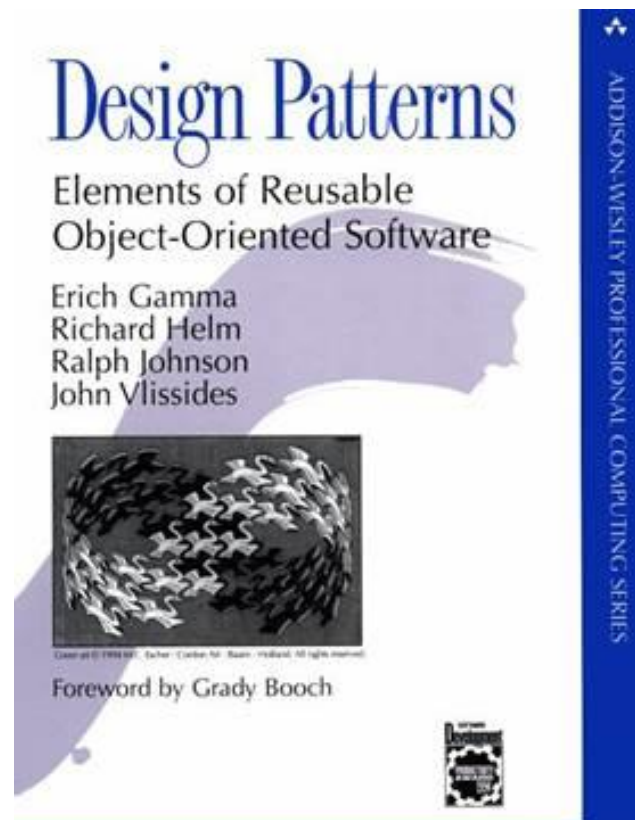
- 里程碑6: Booch Method(1991, G. Booch)
 - 第一个被广泛接受使用的面向对象建模方法



Booch代表作的第三版

面向对象技术发展-7

- 里程碑7：设计模式出版(1995, GoF)
 - 掀起模式运动



面向对象技术发展-8

- 里程碑8：Java语言推出(1995, Sun)
 - 第一个被广泛使用的面向对象语言，J2EE是目前最成功的面向对象框架。



面向对象技术发展-9

- 里程碑9：UML被OMG接纳为标准(1997)
 - 面向对象方法学之战结束



面向对象技术发展-10

- 里程碑10：微软.NET计划(2000)
 - 意义还难以评估



面向对象技术发展-总结

- 60年代后期: Simul67, 基本思想
- 70年代后期: Smalltalk80, 实用化
- 80年代: 理论基础, C++等, 商业化
- 90年代: 面向对象与设计方法学
 - B.H. Sellers等提出喷泉模型
 - G. Booch提出面向对象开发方法等
 - P. Coad和E.Yourdon提出OOA和OOD
 - Jacobson提出OOSE
 -

面向对象
程序设计语言

“方法大战”

1997年: UML

面向对象技术发展-现状

- 现状
 - OO成为最重要的软件开发方法
 - OO在GUI、模拟系统、游戏开发、应用框架、软件构件化领域大显身手
 - Java、UML 与 RUP
 - 构件技术 (CORBA、COM、EJB、.Net)
 - 类库与设计模式

面向对象技术发展-未来

- 未来
 - OO的形式化与自动化
 - OO构件、设计模式的丰富将进一步提高软件开发的效率和质量
 - 软件开发人员必须从 Think Procedurely 转变为 Think Object-Orientedly

内容安排

- 面向对象技术概述
- 面向对象技术发展
- 面向对象与结构化
- 上升到面向对象

面向对象 VS 结构化-1

- 扬弃，不是否定

数据结构+算法 = 程序设计	以对象为中心组织数据与操作
----------------	---------------

数据	对象属性
----	------

操作	对象的服务
----	-------

类型与变量	类与对象实例
-------	--------

函数（过程）调用	消息传递
----------	------

类型与子类型	一般类与特殊类，继承
--------	------------

构造类型	整体 - 部分结构，聚合
------	--------------

指针	关联
----	----

面向对象 VS 结构化-2

结构化方法(SA+SD+SP)

问题域

自然语言

结构化分析

数据流图
数据字典

分析与设计的鸿沟

结构化设计

模块和过程

编程语言

结构化编程, 如C语言

测试

计算机系统

面向对象的方法

问题域

自然语言

OO方法

需求工程

需求模型

OO建模语言

OOA&D

对象模型

OO编程语言

OOP, 如Java语言

测试

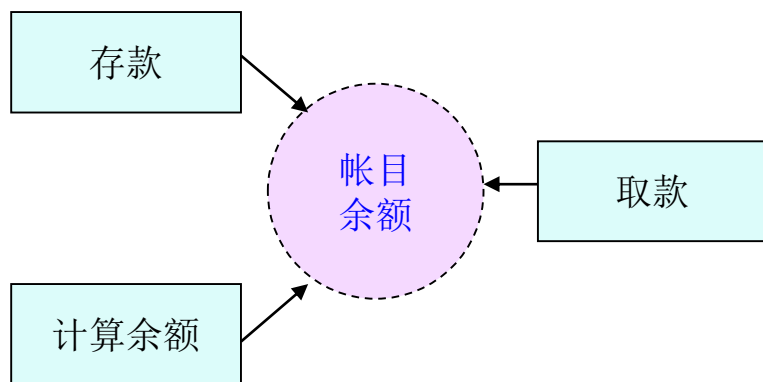
计算机系统



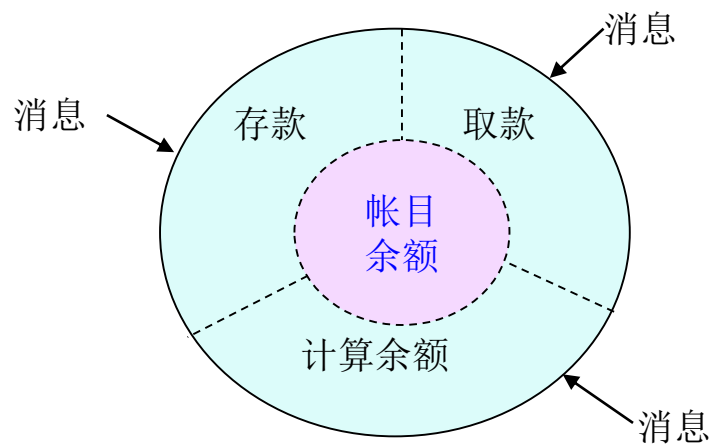
面向对象 VS 结构化-3

	传统结构化方法	面向对象方法(UML建模工具为例)
需求模型	输入I、处理P、输出O的视角, 面向功能的文档(用户需求规格说明书)需求变化, 其功能变化, 所以系统的基础不稳固	从用户和整体角度出发。 使用系统抽象出用例图、活动图, 获取需求; 如需求变化, 对象的性质相对功能稳定, 系统基础稳定
分析模型	面向过程的数据流图DFD、实体—关系图ERD、数据字典DD表示分析模型; 功能分解, 数据和功能/过程分开	把问题作为一组相互作用的实体,, 显式表示实体间的关系 数据模型和功能模型一致 类、对象图表示分析模型, 状态、顺序、协作、活动图细化说明
设计模型	功能模块(SC图), 模块之间的连接/调用是模块的附属形式	类和对象实现, 类/对象的关联、聚集、继承等连接、连接规范和约束作为显式定义
实施模型	体系结构设计	构件图, 配置图
测试模型	根据文档进行单元测试, 集成测试, 确认测试	单元测试采用类图, 集成测试用实现图和交互图, 确认测试采用用例图

实例02



结构化软件的三个模块



面向对象软件的一个类

结构化与面向对象

结构化

- 复杂世界 - > 复杂处理过程 (事情的发生发展)
- 设计一系列功能 (或算法) 以解决某一问题
- 寻找适当的方法存储数据

面向对象

- 任何系统都是由能够完成一组相关任务的对象构成
- 如果对象依赖于一个不属于它负责的任务, 那么就需要访问负责此任务的另一个对象 (调用其他对象的方法)
- 一个对象不能直接操作另一个对象内部的数据, 它也不能使其它对象直接访问自己的数据
- 所有的交流都必须通过方法调用

实例：五子棋

面向过程（事件）的设计思路就是首先分析问题的步骤：

- 1.开始游戏，初始化画面
- 2.黑子走，绘制画面，
- 3.判断输赢，如分出输赢，跳至步骤6
- 4.白子走，绘制画面，
- 5.判断输赢，如未分出输赢，返回步骤2，
- 6.输出最后结果。

面向对象的设计思路是分析与问题有关的实体：

- 1.玩家：黑白双方，这两方的行为是一模一样的，
- 2.棋盘：负责绘制画面
- 3.规则：负责判定诸如犯规、输赢等。

归纳总结

- 结构化设计用算法刻画数据的递归关系，而面向对象思想直接用对象表达递归关系——“模拟现实世界”
- 结构化设计中，数据是死的，全部依赖算法操作，而面向对象中，数据是活的，所谓的smart data
- 结构化设计更像是一个人在解决所有的问题，而面向对象设计更像是一个团队的分工协作

内容安排

- 面向对象技术概述
- 面向对象技术发展
- 面向对象与结构化
- 上升到面向对象

面向对象的程序开发

在结构化程序开发模式中优先考虑的是**过程抽象**，在面向对象开发模式中优先考虑的是**实体**(问题论域的对象)；

主要考虑对象的**行为**而不是必须执行的一**系列动作**；

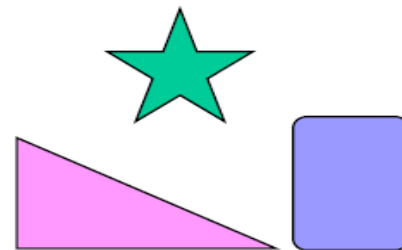
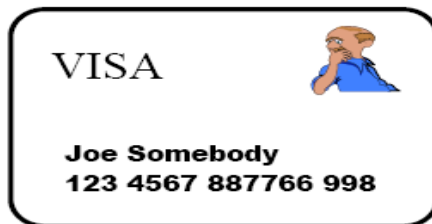
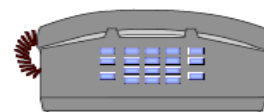
- 对象是数据抽象与过程抽象的综合；
- 算法被分布到各种实体中；
- 消息从一个对象传送到另一个对象；
- 控制流包含在各个对象的操作内；
- 系统的状态保存在各个对象所定义的数据抽象中；

面向对象概念

- 对象
- 类
- 封装
- 继承
- 多态
- 消息

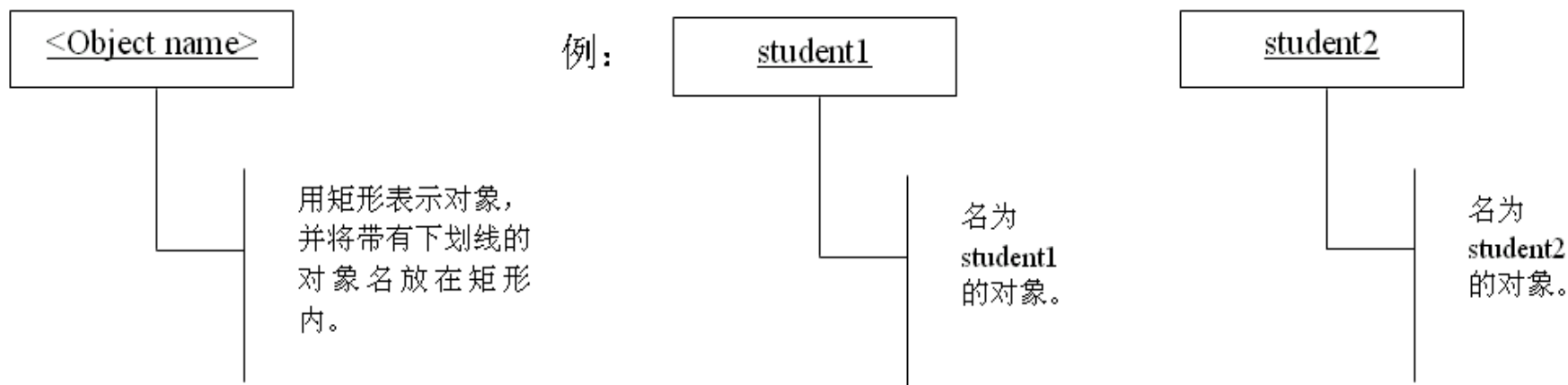
对象 (object)

- **对象**具有责任的实体。一个特殊的，自成一体的容器，对象的数据对于外部对象是受保护的。
- **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。
- **操作** (operation) 规定了对象的行为，表示对象所能提供的服务。也称方法或服务。



对象

- 对象是包含现实世界物体特征的抽象实体，它反映了系统为之保存信息和（或）与它交互的能力。
- 例如，Student对象的数据可能有姓名、性别、出生日期、家庭住址、电话号码等，其操作可能是对这些数据值的赋值及更改。



对象的图形表示

对象

- 对象与类具有几乎完全相同的表示形式，主要差别是对象的名字下面要加一条下划线。对象名有下列三种表示格式：

(1) 第一种格式是对象名在前，类名在后，中间用冒号连接。
形如：

对象名：类名

(2) 第二种格式形如：

：类名

这种格式用于尚未给对象命名的情况，注意，类名前的冒号不能省略。

(3) 第三种格式形如：

对象名

对象

- 对象有两个层次的概念：
 - (1) 现实生活中对象指的是客观世界的实体。可以是可见的有形对象，如人、学生、汽车、房屋等；也可以是抽象的逻辑对象，如银行帐号，生日。
 - (2) 程序中对象就是一组变量和相关方法的集合，其中变量表明对象的状态，方法表明对象所具有的行为。

对象

可以将程序中的对象分为5类：物理对象，角色，事件，交互，规格说明。

(1) 物理对象 (Physical Objects) —— 物理对象是最易识别的对象，通常可以在问题领域的描述中找到，它们的属性可以标识和测量。

例如，大学课程注册系统中的学生对象；一个网络管理系统中各种网络物理资源对象（如开关、CPU和打印机）都是物理对象。

对象

(2) 角色（Roles）—— 一个实体的角色也可以抽象成一个单独的对象。角色对象的操作是由角色提供的技能。

- 例如，一个面向对象系统中通常有“管理器”对象，它履行协调系统资源的角色。一个窗口系统中通常有“窗口管理器”对象，它扮演协调鼠标器按钮和其他窗口操作的角色。特别地，一个实际的物理对象可能同时承担几个角色。
- 例如，一个退休教师同时扮演退休者和教师的角色。

对象

(3) 事件（Events）—— 一个事件是某种活动的一次“出现”。

- 例如“鼠标”事件。一个事件对象通常是一个数据实体，它管理“出现”的重要信息。事件对象的操作主要用于对数据的存取。
- 如“鼠标”事件对象有诸如光标坐标、左右键、单击，双击等信息。

对象

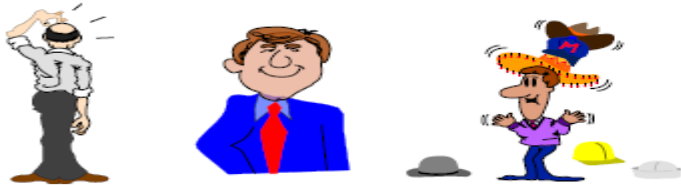
- (4) 交互（Interactions）—— 交互表示了在两个对象之间的关系，这种类型的对象类似于在数据库设计时所涉及的“关系”实体。
- 当实体之间是多对多的关系时，利用交互对象可将其简化为两个一对多的关系。
 - 例如，在大学课程注册系统中，学生和课程之间的关系是多对多的关系，可设置一个“选课”交互对象来简化它们之间的关系。

类

类(Class): 具有相同属性和操作的一组对象的抽象, 它为属于该类的全部对象提供了统一的抽象描述。

- 类是概念定义, 抽象了同类对象共同的属性和操作
- 对象是类的一个实例

Person objects



abstracts to

Person class

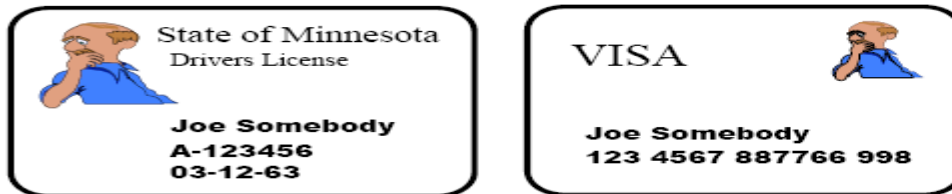
Attributes

name
age
height
weight

Operations

move
change-job

Card objects



Card class

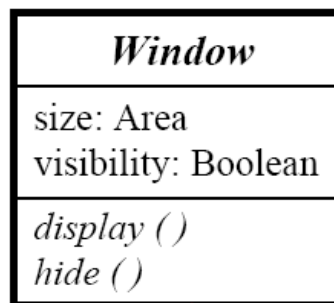
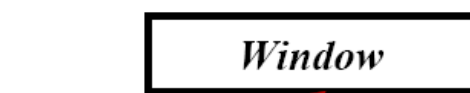
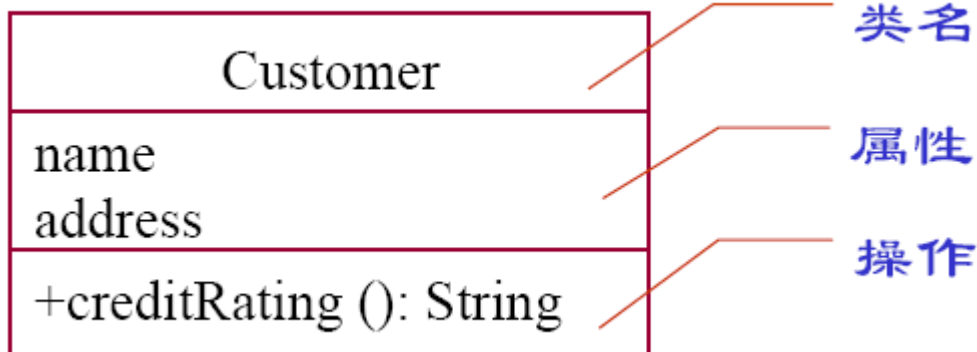
Attributes

height
width
id-number

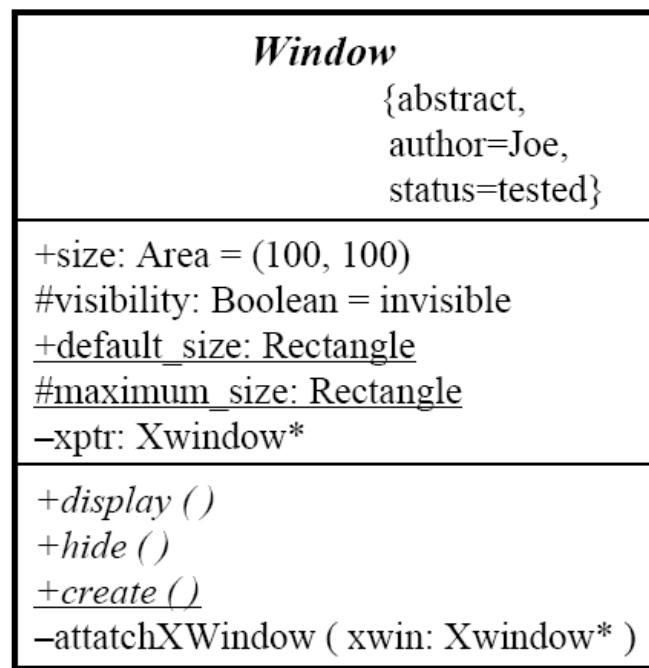
Operations

issue
change

类与封装



分析层次



设计层次

类与对象的对比

类与对象的比较

- “同类对象具有相同的属性和操作”是指它们的定义形式相同，而不是说每个对象的属性值都相同。
- 类是静态的，类的存在、语义和关系在程序执行前就已经定义好了。
- 对象是动态的，对象在程序执行时可以被创建和删除。

```
Class Student {  
    String sno;  
    String sname;  
    String dept;  
  
    public Student (String sno, string sname,  
                    string dept) { };  
    public boolean RegisterMyself() { };  
    public boolean SelectCourses() { };  
    private float QueryScore(int courseID) { };  
}
```

```
Student 张三 = new Student  
    ( "1080310501", "张三", "CS"  
    );  
  
Student 李四 = new Student  
    ( "1080310502", "李四", "CS"  
    );  
  
Student 王五 = new Student  
    ( "1080310503", "王五", "CS"  
    );
```

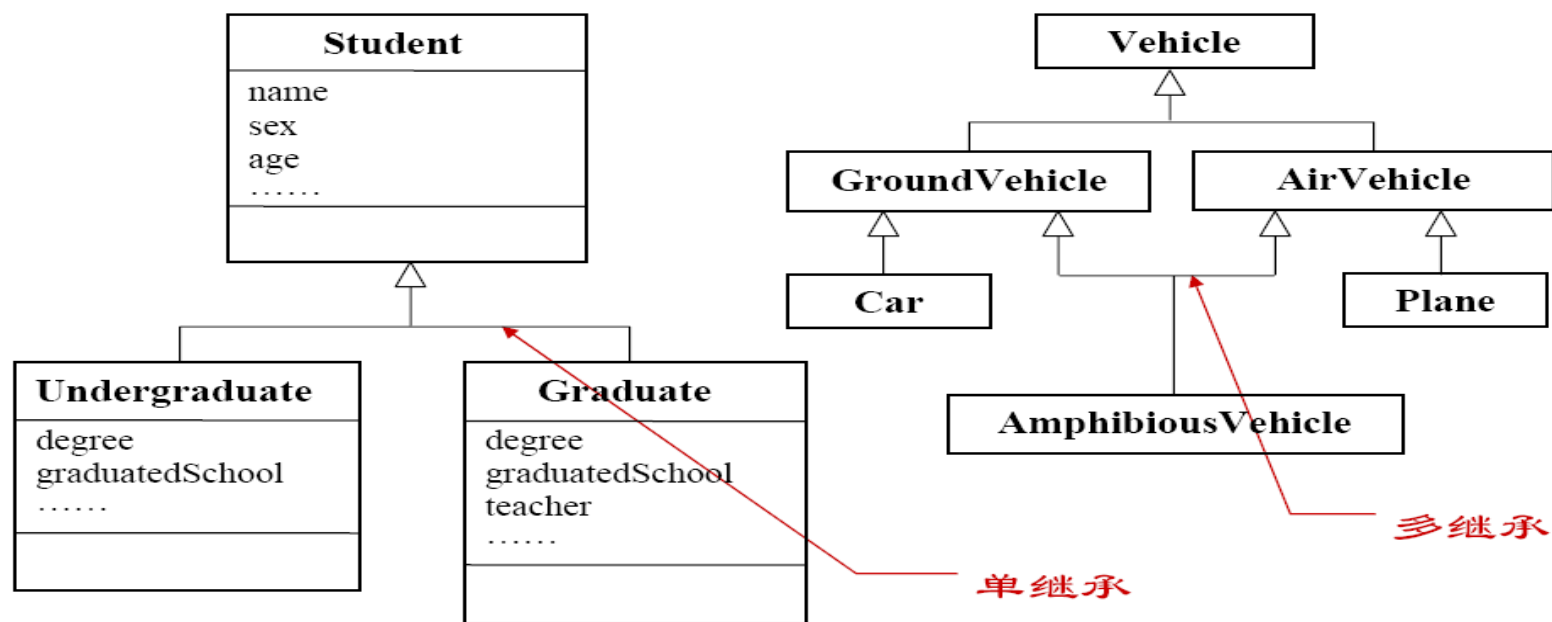
类与封装

- **封装** 是一种信息隐蔽技术，就是利用抽象数据类型将数据和基于数据的操作封装在一起。用户只能看到对象的封装界面信息，对象的内部细节对用户是隐蔽的。
- 封装的定义是：
 - (1) 清楚的边界，所有对象的内部信息被限定在这个边界内；
 - (2) 接口，即对象向外界提供的方法，外界可以通过这些方法与对象进行交互；
 - (3) 受保护的内部实现，即软件对象功能的实现细节，实现细节不能从类外访问。

例如：对“汽车”对象来说，“司机”对象只能通过方向盘和仪表来操作“汽车”，而“汽车”的内部实现机制则被隐藏起来。

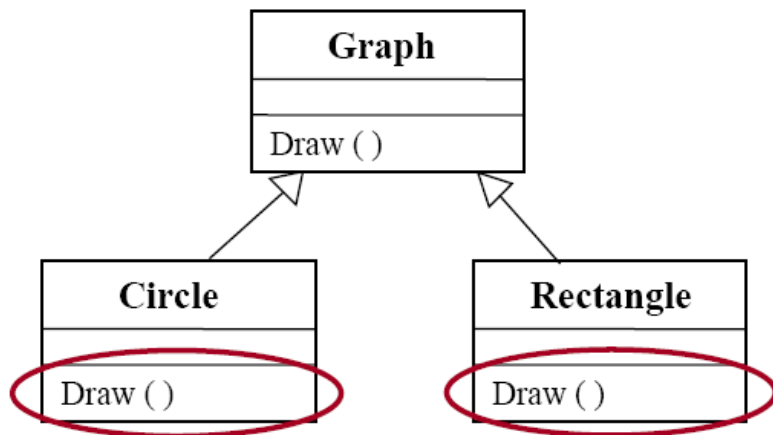
继承

- 继承是一种联结类的层次模型，为类的重用提供了方便，它提供了明确表述不同类之间共性的方法。
- 我们将公共类称为超类(superclass)、父类 (father class)、祖先 (ancestor) 或基类 (base class)，而从其继承的类称为子类 (subclasses)、后代 (deslendane) 或导出类 (derived class)。



多态

- 根据为请求提供服务的对象不同可以得到不同的行为，这种现象称为多态。
- 在运行时对类进行实例化，并调用与实例化对象相应的方法，称为**动态绑定**、**后期绑定**或**运行时绑定**。相应地，如果方法的调用是在编译时确定的，则称为是**静态绑定**、**前期绑定**或**编译时绑定**。
- 通过在子类中覆盖父类的方法实现多态。



消息通信

- 消息是一个对象与另一个对象的通信单元，是要求某个对象执行类中定义的某个操作的规格说明。
- 发送给一个对象的消息定义了一个方法名和一个参数表（可能是空的），并指定某一个对象。
- 一个对象接收到消息，则调用消息中指定的方法，并将形式参数与参数表中相应的值结合起来。

发送消息



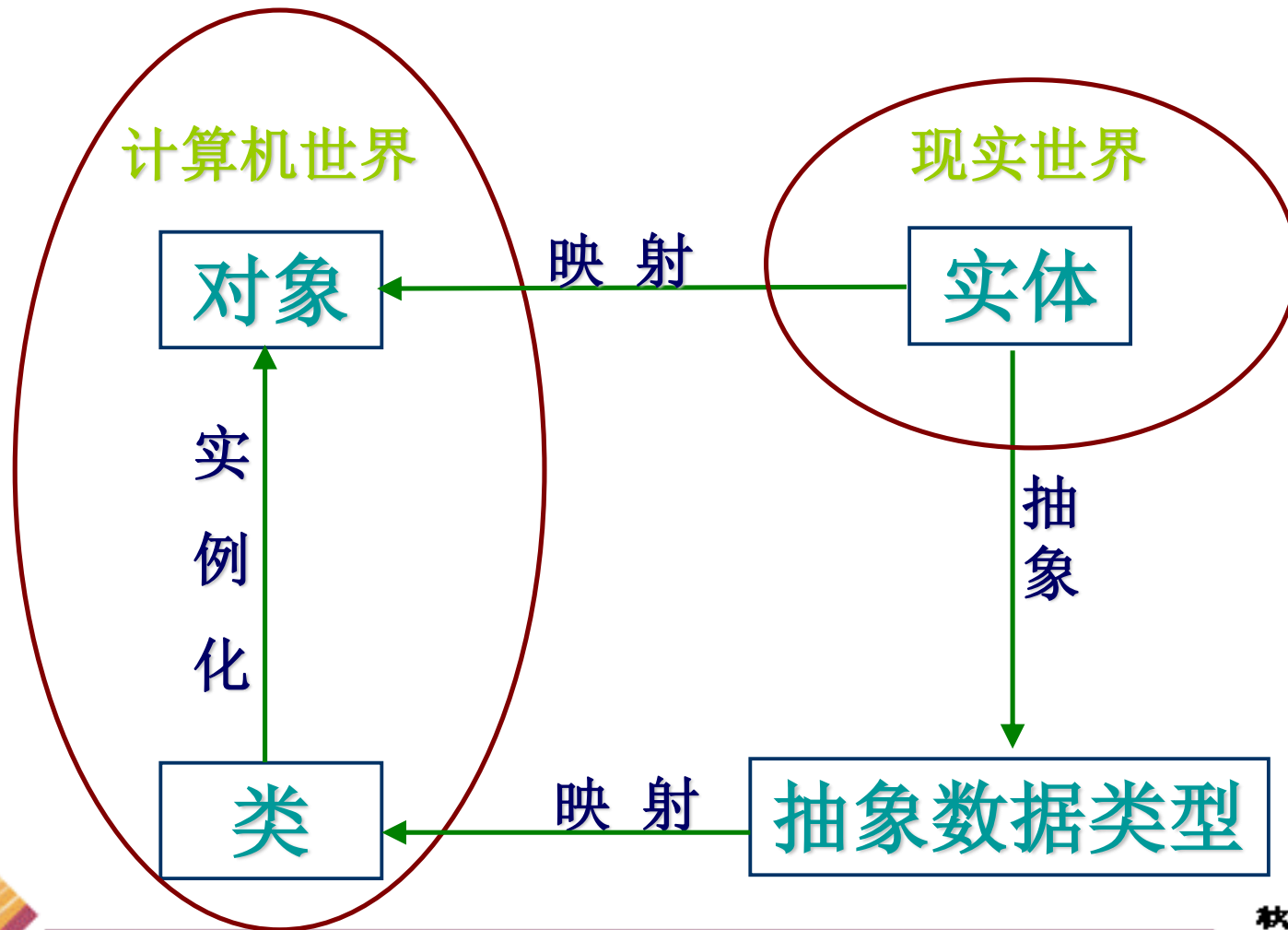
加速

`speed=myCar.speedup(1)`

接收并响应消息



类和对象



面向对象技术

- 客观世界是由**对象**组成的，任何客观事物或实体都是对象；复杂对象可以由简单对象构成；
- 具有相同数据和相同操作的对象可以归并为一个统一的“**类**”，对象是类的实例；
- 类可以派生出子类，子类**继承**父类的全部特性(数据和操作)，同时加入了自己的新特性；子类和父类形成层次结构；
- 对象之间通过**消息**传递相互关联；
- 类具有**封装**性，其数据和操作对外是不可见的，外界只能通过消息请求某些操作。
- 具体的**计算**则是通过新对象的建立和对象之间的通信来执行的。

小结

- 1、面向对象与结构化
- 2、面向对象概念
- 3、面向对象方法