# 软件工程
# 第三部分 面向对象方法学

8.1 面向对象方法学概述
8.2 面向对象的概念
8.3 对象建模技术
8.4 UML统一建模语言*



## 8.1 面向对象方法论概述

- 在20世纪60年代后期出现的面向对象编程语言Simula-67中首次引入了类和对象的概念。
- 20世纪80年代中期起，人们开始注重面向对象分析和设计的研究，逐步形成了面向对象方法学。
- 20世纪90年代，面向对象方法学已经成为人们在开发软件时首选的范型。
- 面向对象技术是当前最好的软件开发技术。



## 面向对象方法学的基本原则

- 尽可能模拟人类习惯的思维方式，使开发软件的方法与过程尽可能接近人类认识世界、解决问题的方法与过程，也就是使描述问题的问题空间与实现解法的解空间在结构上尽可能一致。
  - ✓ 客观世界的应用问题都是由客观世界中的实体及实体相互间的关系构成的。把实体抽象为问题域中的对象。
  - ✓ 问题空间中，实体具有作用、责任以及与其它实体协作。将实体的行为视对对象操作。



## 什么是面向对象

- Peter Coad和Edward Yourdon提出面向对象方法的概念:
  面向对象=对象+分类+继承+消息通信
- 用这四个概念开发的软件系统就是面向对象的软件系统。
- 如果仅使用对象和消息，称为基于对象的方法。
- 如果进一步把所有对象都划分为类，称为基于类的方法。



## 面向对象方法的要点

1. 面向对象的软件系统是由对象组成的，软件中的任何元素都是对象，复杂的软件对象由比较简单的对象组合而成。计算是通过对象的建立和对象之间的通信来执行的。面向对象方法用对象分解取代了传统方法的功能分解。
2. 把所有对象都划分成各种对象类，每个对象类都定义了一组数据和方法。数据用于表示对象的静态属性，是对象的状态信息。方法是允许施加于该类对象上的操作，为该类所有对象共享，并不需要为每个对象都复制操作的代码。



3. 采用继承机制把若干个对象类组成一个层次结构的系统。在这种层次结构中，通常下层的派生类（子类）具有和上层的基类（父类）相同的特性(包括数据和方法）。
4. 对象彼此之间仅能通过传递消息互相联系。对象与传统的数据有本质区别，它不是被动地等待外界对它施加操作，相反，它是进行处理的主体。必须发消息请求对象主动地执行它的某些操作，处理它的私有数据，而不能从外界直接对它的私有数据进行操作。

## 面向对象方法学的优点

1) 与人类习惯的思维方法一致：使用现实世界的概念抽象地思考问题，从而自然地解决问题。
   - ✓传统的软件开发方法以算法为核心，把数据和过程作为相互独立的部分。
   - ✓面向对象开发方法以对象为核心，开发出的软件系统由对象组成。对象是对现实世界实体的正确抽象，是数据和操作封装在一起所构成的统一体。对象之间通过传递消息互相联系，以模拟现实世界中不同事物彼此之间的联系。



2) 稳定性好：现实世界中的实体是相对稳定的，以对象为中心构造的软件系统也是比较稳定的。
   - ✓ 用传统方法所建立起来的软件系统的结构紧密依赖于系统所要完成的功能，当功能需求发生变化时将引起软件结构的整体修改。这样的软件系统是不稳定的。
   - ✓ 面向对象方法基于构造问题领域的对象模型，以对象为中心构造软件系统。



3) 可重用性好：对象是比较理想的模块和可重用的软件成分。
   - ✓ 传统的软件重用技术是利用标准函数库。
   - ✓ 面向对象方法中，数据和操作正是作为平等伙伴出现的对象中的。此外，对象固有的封装性和信息隐藏机制，使得对象的内部实现与外界隔离，具有较强的独立性。对象是比较理想的模块和可重用的软件成分。



4) 可维护性好
   - ✓ 稳定性比较好
   - ✓ 比较容易修改
   - ✓ 比较容易理解
   - ✓ 易于测试和调试
5) 较易开发大型软件产品



## 8.2 面向对象的概念

- 对象
- 类
- 实例
- 消息
- 方法

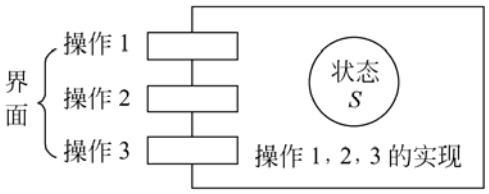
- 属性
- 封装
- 继承
- 多态性
- 重载



## 对象

- 对象是系统中描述客观事物的实体，是构成系统的一个基本单元，由一组属性值和一组对属性进行操作的服务组成。

  对象 = 对象名+数据(属性)+操作(行为)



- 什么可以成为对象?

  在应用领域中有意义的、与所要解决的问题有关系的任何事物都可以作为对象，它既可以是具体的物理实体的抽象，也可以是人为的概念，或者是任何有明确边界和意义的东西。

  对象可以是外部实体、信息结构、事件、角色、组织结构、地点或位置、操作规程等。



## 对象的特点

- 抽象性：对象是数据抽象与过程抽象的综合体。
  - ✓过程抽象是指当使用某个过程时，无需关心过程内部的实现细节，只要知道如何调用该过程以及该过程完成什么功能即可。
  - ✓数据抽象是指使用结构或记录等方式把某个实体的数据集中起来，使得使用者能够以单元为单位使用数据。

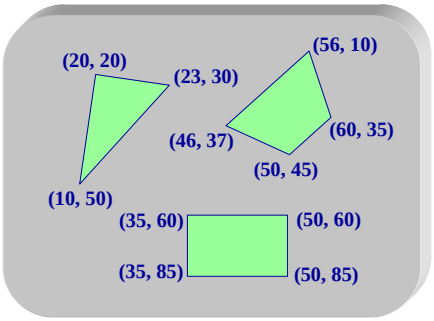


- 封装性：信息隐蔽（使用与实现分开）。把私有数据完全被封装内部，对外不可见。私有数据的访问或处理只能通过公有的操作进行。
- 共享性：同一个类中所有对象共享相同的数据结构和行为；同一应用中的对象通过继承关系，共享共同的数据结构和行为；不同应用中的对象通过复用，共享数据结构和行为。
- 以数据为中心：操作围绕对其数据所需要做的处理来设置，不设置与这些数据无关的操作。面向对象范型在数据抽象中组织过程抽象。



- 模块独立性好：对象内部各种元素彼此结合得很紧密，内聚性相当强；由于完成对象功能所需要的元素(数据和方法)基本上都被封装在对象内部，它与外界的联系自然就比较少，因此，对象之间的耦合通常比较松。
- 分布性：系统的状态分别保存在各个对象的数据存储中；解决问题的控制流包含在各个对象中的操作内；算法被分布到各种实体中。






计算机窗口中的三个多边形对象



| triangle | quadrilateral1 | quadrilateral2 |
|---|---|---|
| (10, 50)<br>(20, 20)<br>(23, 30) | (46, 37)<br>(50, 45)<br>(60, 35)<br>(56, 10) | (35, 60)<br>(35, 85)<br>(50, 85)<br>(50, 60) |
| draw<br>move(Δx, Δy)<br>contains?(aPoint) | draw<br>move(Δx, Δy)<br>contains?(aPoint) | draw<br>move(Δx, Δy)<br>contains?(aPoint) |

表示多边形的三个对象

## 类与实例

- 现实世界中，分类是人类认识客观世界的基本方法。把有相似特征的事物归为一类。
- 面向对象方法中，把具有相同属性和服务的对象归在一起就形成了类。类的定义包括一组数据属性和在数据上的一组合法操作。
- 属于某一个类的各个对象都是该类的实例，它们都可使用类中的操作。
- 类定义了各个实例所共有的数据结构，使用类的构造函数，可以在创建该类的实例时初始化这个实例的状态。



| quadrilateral1 | quadrilateral2 | quadrilateral |
|---|---|---|
| (46, 37)<br>(50, 45)<br>(60, 35)<br>(56, 10) | (35, 60)<br>(35, 85)<br>(50, 85)<br>(50, 60) | point1<br>point2<br>point3<br>point4 |
| draw<br>move(Δx, Δy)<br>contains?(aPoint) | draw<br>move(Δx, Δy)<br>contains?(aPoint) | draw<br>move(Δx, Δy)<br>contains?(aPoint) |

由两个四边形对象导出一个类



## 消息

- 消息是一个对象向另一个对象传递的信息，要求某个对象执行在定义它的那个类中所定义的某个操作的规格说明。
- 一个消息由下述三个部分组成：
  - ✓ 接收消息的对象；
  - ✓ 消息选择符(也称为消息名)；
  - ✓ 零个或多个变元（参数）。



- 消息的使用类似于函数调用，消息中指定了某一个实例，一个操作名和一个参数表（可能是空的）。

quadrilateral1.move(15, 20)

- 接收消息的对象执行消息中指定的操作，并将形式参数与消息中相应的变元值结合起来。



## 属性与方法

- 属性就是类中所定义的数据，它是对客观世界实体所具有的性质的抽象。类的每个实例都有自己特有的属性值。

  例quadrilateral1的四个点

  (46, 37)，(50, 45)，(60, 35)，(56, 10)

- 方法就是对象所能执行的操作，也就是类中所定义的服务。方法描述了对象执行操作的算法，响应消息的方法。

  例quadrilateral.move( ， )



## 封装

- 所谓封装就是把某个事物包起来，使外界不知道该事物的具体内容。面向对象方法中，一个对象是一个不透明的黑盒子，封装了数据和实现操作的代码。
- 使用一个对象只需知道它向外界提供的接口形式，无须知道它的数据结构细节和实现操作的算法。通过封装对外界隐藏了对象的实现细节。

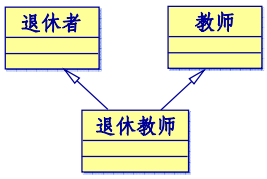
## 继承

- 继承是指能够直接获得已有的性质和特征。面向对象技术中，继承是子类自动地共享基类中定义的数据和方法的机制。
- 继承是使用已有的类定义做为基础建立新类的定义技术。
- 建立继承结构的好处:
  - ✓易编程、易理解、代码短、结构清晰
  - ✓易修改：共同部分只要在一处修改即可
  - ✓易增加：新类只须描述不同部分



## 多继承

- 如果一个类需要用到多个现有类的特征，可以从多个类中继承，称为多继承。
- 例如退休教师是继承退休者和教师这两个类的某些特征或行为而得到的一个新类。






## 重载

- 两种重载形式:
  - ✓函数重载是指在同一作用域内的若干个参数特征不同的函数可以使用相同的函数名字;
  - ✓运算符重载是指同一个运算符可以施加于不同类型的操作数上面。
- 当参数特征不同或被操作数的类型不同时，实现函数的算法或运算符的语义是不相同的。
- 在C++语言中，编译时决定到底使用函数的哪个实现代码或算符的具体语义。



- 运算符重载举例
  - a) 整数“+”：整数加法
  - b) 浮点数“+”：浮点数加法
  - c) 字符串“+”：字符串连接
  - d) 点“+”：两个点的坐标位置分别叠加

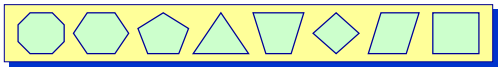


## 多态性

- 多态性是在类等级的不同层次中可以共享一个方法的名字，然而不同层次中的每个类却各自按自己的需要来实现这个行为。
- 具有多态的函数或操作在运行时才根据实际的对象类型，执行相应实现程序的连接，此即动态绑定。



- 例如，想要在屏幕上画一系列多边形，多态性允许发送消息draw，根据消息接收对象的类型不同，画出不同的多边形。draw针对的是一系列的类型（类族）而不仅仅是一个类型。



- C++语言中，多态性是通过虚函数来实现的。在类等级不同层次中可以说明名字、参数特征和返回值类型都相同的虚拟成员函数，而不同层次的类中的虚函数实现算法各不相同。
- 动态绑定把函数调用与目标代码块的连接延迟到运行时进行（滞后联编），保证在程序执行时实施与对象P连接的操作。如果P是矩形类的实例，则执行与矩形连接的操作，如果P是三角形类的实例，则执行与三角形连接的操作。



## 8.3 对象建模技术

- 模型是为了理解事物而对事物作出的一种抽象。模型由一组图示符号和组织这些符号的规则组成，可用来定义和描述问题域中的术语和概念。
- 建模的目的主要是为了减少复杂性。
  - 开发复杂的软件系统时，从不同角度抽象出目标系统的特性，使用精确的表示方法构造系统的模型；
  - 验证模型是否满足用户对目标系统的需求；
  - 在设计过程中逐渐把和实现有关的细节加进模型中，直至最终用程序实现模型。(模型驱动)



- 面向对象建模时，通常需要建立3种形式的模型。从不同侧面描述了所要开发的系统:
  - ✓对象模型:描述系统的静态结构，定义了做事情的实体。
  - ✓动态模型:描述系统的主要行为，明确规定了在何种状态下接受了什么事件的触发。
  - ✓功能模型:描述如何实现系统功能，指明了系统应该“做什么”。
- 上述3种模型互相补充，是大系统开发必不可少。对象模型始终都是最重要、最基本、最核心的，为其他两种模型奠定了基础。



## 对象模型

- 对象模型是对模拟客观世界实体的对象以及对象彼此间的关系的映射，描述了系统的静态结构。对象模型为建立动态模型和功能模型，提供了实质性的框架。
- 为了建立对象模型，需要用适当的建模语言来表达模型，建模语言由记号和使用记号的规则组成。
- 通常，使用UML提供的类图来建立对象模型。类图描述类及类与类之间的静态关系。类图是一种静态模型，它是创建其他UML图的基础。



## 动态模型

- 一旦建立起对象模型之后，就需要考察对象的动态行为。动态模型表示瞬时的、行为化的系统的控制性质，它规定了对象模型中的对象的合法变化序列。
- 对象的生命周期由许多阶段组成。在每个特定阶段中，都有适合该对象的一组运行规律和行为规则。生命周期中的阶段也就是对象的状态。所谓状态，是对对象属性值的一种抽象。
- 通常，用UML提供的状态图来描绘对象的状态、触发状态转换的事件以及对象的行为。



## 功能模型

- 功能模型表示变化的系统的“功能”性质，它指明了系统应该“做什么”，因此更直接地反映了用户对目标系统的需求。
  - ➢在传统的结构化开发方法中，功能模型由一组数据流图组成。
  - ➢在面向对象方法中，UML用例图是进行需求分析和建立功能模型的强有力工具。用例图建立起来的系统模型称为用例模型。用例模型描述的是外部行为者所理解的系统功能，描述了开发者和用户对需求规格所达成的共识。

## 8.4 统一建模语言UML*

- UML 是Unified Modeling Language的缩写。 它是一种标准的语言，以直观的表述、定义、构造和文档化软件为主的系统的工作制品。
- UML聚集了来自下列建模的精髓:
  - ✓ 数据建模（实体关系图ERD）
  - ✓ 业务建模 （工作流）
  - ✓ 对象建模
  - ✓ 构件建模
- 它可用于软件生命周期各个过程，并适用于各种不同的实现技术。



## UML的特点

- 统一标准
  - ✓ 融合了当前一些流行的面向对象开发方法的主要概念和技术，成为一种面向对象的标准化的统一建模语言。
  - ✓ 提供了标准的面向对象的模型元素的定义和表示法，有标准的语言工具可用。
  - ✓ 已成为工业标准化组织OMG的正式标准。
- 面向对象
  - ✓ 支持面向对象的主要概念，提供了一批基本的模型元素的表示图形和方法。



- 可视化，表示能力强大
  - ✓ 一种图形化语言，系统的逻辑模型和实现模型都能用UML的模型图形清晰地表示。
  - ✓ 可以处理与软件的说明和文档有关的问题。
  - ✓ 提供了语言的扩展机制，用户可以根据需要增加定义自己的构造型、标记值和约束等。
  - ✓ 可用于各种复杂类型的软件系统的建模。
- 独立于过程
  - ✓ 系统建模语言，独立于开发过程。



- 容易掌握使用
  - ✓ 概念明确，建模表示法简洁明了，图形结构清晰，容易掌握使用。
  - ✓ 着重学习三个方面的主要内容:
    (1) UML的基本模型元素
    (2) 组织模型元素的规则
    (3) UML语言的公共机制
- 与程序设计语言的关系
  - ✓ 用Java，C++ 等编程语言可实现一个系统。
  - ✓ 一些CASE工具可以根据 UML所建立的系统模型来产生Java、C++ 等代码框架。



## UML的定义

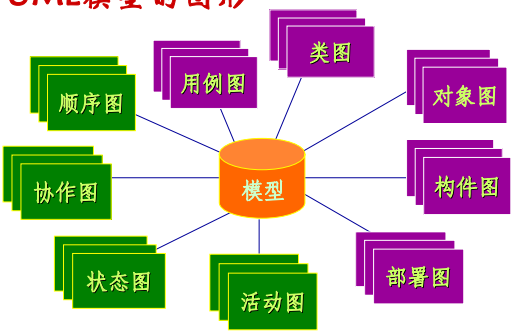
- UML是一种可视化的建模语言。UML定义有两个组成部分:
  - ✓语义用自然语言描述;
  - ✓表示法定义了可视化标准表示符号。
- 在语义上，模型是元模型的实例。UML定义给出了语法结构的精确定义。
- 使用UML时，要从不同的角度观察系统，为此定义了概念“视图”。视图是对系统的模型在某方面的投影，注重于系统的某个方面。



## UML的构成

- UML的三个主要组成元素
  1) 基本构造块（basic building blocks ）
  2) 组织构造块的规则（rules）
  3) 运用于整个UML的公共机制
     （common mechanisms）
- UML包括三种基本构造块:
  1) 元素（things）
  2) 关系（relationships）
  3) 图（diagrams）

## UML模型的图形






## UML图的作用

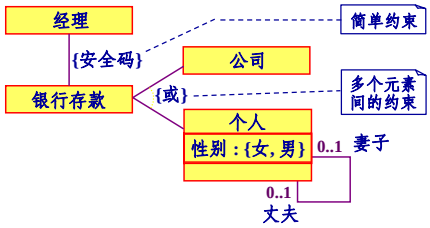
- UML 可以用于:
  - ✓ 使用用例（use cases）和参与者（actors）描述系统的边界和它的主要功能。
  - ✓ 使用交互图（顺序图、协作图）具体描述用例的实现。
  - ✓ 使用类图表示系统的静态结构。
  - ✓ 使用状态图模型化对象的行为。
  - ✓ 使用构件图和部署图展现系统的物理实现体系结构。
  - ✓ 使用构造型（stereotypes）扩展建模能力。



- 扩展机制 (extended mechanisms)

❶约束 (Constraint)

- ✓ 约束可增加新的语义或改变已有的规则。
- ✓ 约束用{…}表明，放在相关元素附近。






❷标记值（Tagged Value）

- ✓用标记值可为UML的事物增加新的特性。
- ✓标记值与类的属性不同。标记值可以看作是元数据，它的值应用到元素本身，而不是它的实例。
- ✓标记值常用来详述与代码生成或配置管理有关的特性。如指定特定类映射到的编程语言，或描述一个构件的作者或版本。
- ✓标记值用{…(标记) = …(值)}表示。

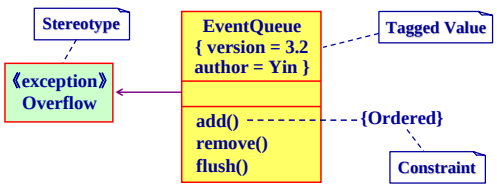


❸构造型 (Stereotype)

- ✓ 如果发现需要一个模型构造，它不在UML中但又与UML的某一成分类似，可以将它看作UML构造的一个构造型。
- ✓ 例如，UML接口是一个类，它只有一些不具有方法实现和属性的公用操作，它是一种特别的类，可以定义它为类的构造型。
- ✓ 当对节点或类建立构造型时，是通过创建类似于已有构造块的新构造块来扩展 UML。
- ✓ 在UML表示法中可以用《…》表示构造型。



- 构造型可用于分类和扩充关联、继承关系、类和构件。例如:
  - ✓ 类的构造型：边界、控制、实体、实用程序、异常;
  - ✓ 继承的构造型：uses和extends;
  - ✓ 构件的构造型：子系统。



## 小结

- 面向对象方法学认为，客观世界由对象组成。任何事物都是对象，每个对象都有自己的内部状态和运动规律，不同对象彼此间通过消息相互作用、相互联系，从而构成了我们所要分析和构造的系统。
- 面向对象方法学比较自然地模拟了人类认识客观世界的思维方式，问题空间和解空间在结构上尽可能一致。对于大型软件产品来说，面向对象范型明显优于结构化范型。使用面向对象范型能够开发出稳定性好、可重用性好和可维护性好的软件。



- 用面向对象观点建立系统的模型通常包括：对象模型（描述系统静态结构）、动态模型（描述系统控制结构）、以及功能模型（描述系统计算结构）。对象模型是最基本、最核心、最重要的。
- 统一建模语言UML是国际对象管理组织OMG批准的基于面向对象技术的标准建模语言。喷泉模型是典型的面向对象软件过程模型。