

# 第11章 面向对象设计

## 目录

- 11.1 面向对象设计的准则
- 11.2 启发规则
- 11.4 系统分解
- 11.5 设计问题域子系统
- 11.6 设计人机交互子系统
- 11.7 设计任务管理子系统
- 11.8 设计数据管理子系统
- 11.9 设计类中的服务
- 11.10 设计关联
- 11.11 设计优化

# 面向对象设计

- 分析：提取、整理用户需求，建立问题域精确模型。
- 设计：转变需求为系统实现方案，建立求解域模型。
  - 在实际的软件开发过程中分析和设计的界限是模糊的。
  - 分析和设计活动是一个多次反复迭代的过程。
  - 面向对象方法学在概念和表示方法上的一致性，保证了在各项开发活动之间的平滑(无缝)过渡，领域专家和开发人员能够比较容易地跟踪整个系统开发过程，这是面向对象方法与传统方法比较起来所具有的一大优势。

# 11.1 面向对象设计的准则

1. 模块化
2. 抽象
3. 信息隐藏
4. 弱耦合
5. 强内聚
6. 可重用

# 模块化

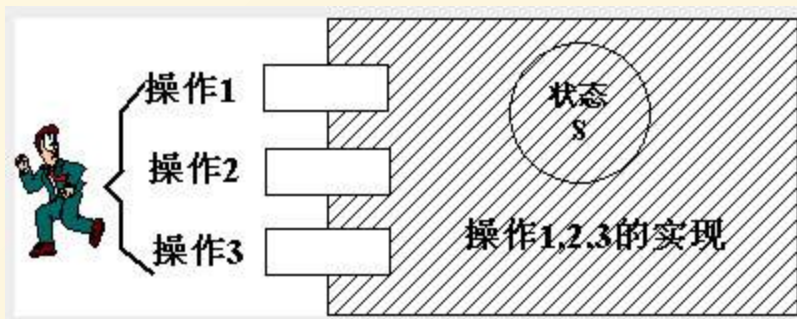
面向对象的软件开发模式，支持了系统模块化的原则：对象就是模块。它把数据结构和操作（方法）紧密地结合在一起构成模块。

# 抽象

类实际上是一种抽象数据类型，它对外开放的公共接口构成了类的规格说明（协议），这种接口规定了外界可以使用的合法操作符，利用这些操作符可以对类的实例中包含的数据进行操作。

# 信息隐藏

在面向对象方法中，信息隐藏通过对象的封装性实现：类结构分离了类的接口与类的实现，从而支持了信息隐藏。



# 弱耦合

弱的耦合可以提高软件模块的独立性，避免某一部分模块发生变化对其它模块有较大的影响。

一般来说，对象间的耦合有两大类：

- 交互耦合：对象间的耦合通过消息连接来实现。应使交互耦合尽量松散。
  - 降低消息连接复杂度（减少参数个数，降低参数复杂度）
  - 减少消息数
- 继承耦合：与交互耦合相反，应该提高继承耦合的程度。因为通过继承关系结合起来的基类和派生类，结合得越紧密越好。

# 强内聚

面向对象设计中存在三种内聚：

- 服务内聚：一个服务应该完成一个且完成一个功能。
- 类内聚：一个类应该只有一个用途，它的属性和服务应该是高内聚的。
- 一般—特殊内聚：即基类—派生类的内聚。
  - 继承既是耦合，又是内聚！



# 可重用

使用已经存在的类（包括开发环境提供的类库，及以往开发相似系统时创建的类），可以提高软件重用性，提高软件生产率。

- 代码重用
  - 源代码包含
  - 源代码继承
- 设计结果重用：重用**设计模型**
- 分析结果重用：重用**分析模型**

## 11.2 启发规则

1. 设计结果应该清晰易懂
2. 一般—特殊结构的深度应适当 ( $7 \pm 2$ )
3. 设计小而简单的类，便于开发和管理
4. 设计简单的协议（即接口）
5. 设计简单的服务
6. 把设计变动减至最小

# 设计结果应该清晰易懂

- 用词一致
- 使用已有协议
- 减少消息模式的数目
- 避免模糊的定义

# 设计简单的类

- 避免包含过多的属性
- 有明确的定义，分配给每个类的任务应简单
- 尽量简化对象之间的合作关系
- 避免提供过多服务（公共服务一般不超过7个）

# 11.4 系统分解

- 问题域：直接负责实现客户需求子系统。
- 人机交互：实现用户界面子系统。
- 任务管理：确定各类任务，把任务分配给适当的硬件或软件去执行。
- 数据管理：负责对象的存储和检索的子系统。

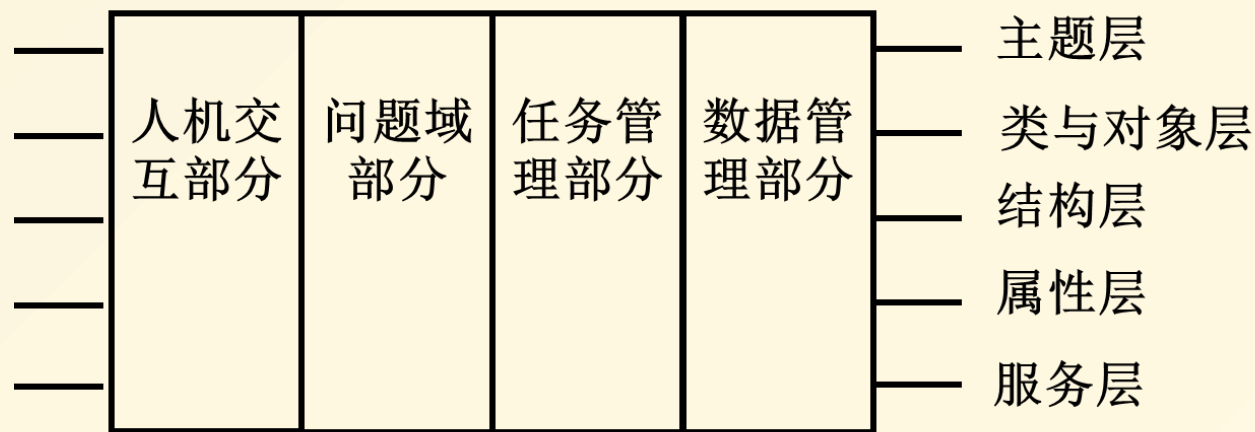


图11.2 典型的面向对象设计模型

# 11.5 设计问题域子系统

在设计问题域部分时，仅需要从实现的角度对问题域模型（对象模型、动态模型、功能模型）作一些补充、修改，主要是增添、合并或分解类与对象、属性和服务，调整继承关系等。

1. 调整需求
2. 重用已有的类
3. 组合问题域的类
4. 增添基类以定义公共函数集合
5. 调整继承层次

# ATM系统问题域子系统结构

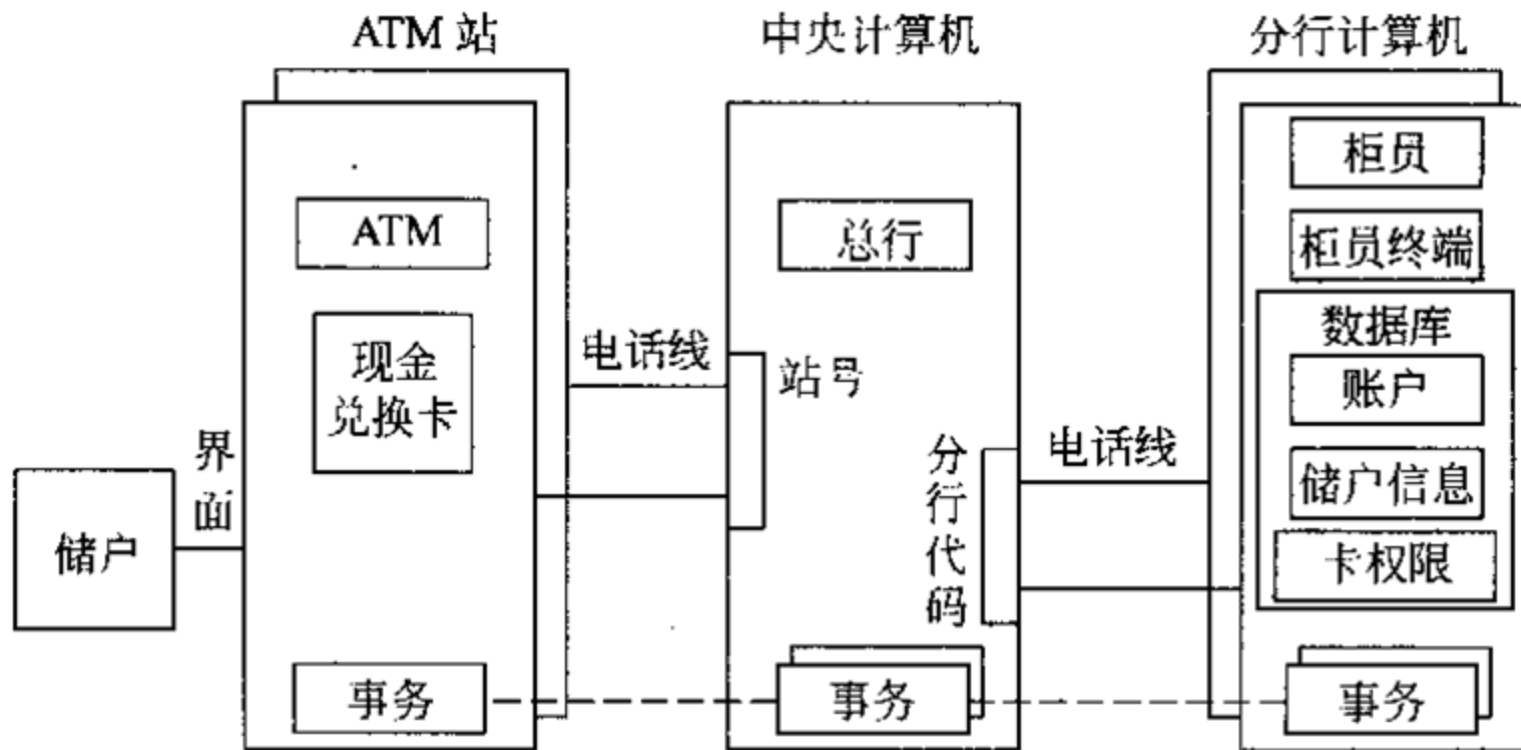


图 11.7 ATM 系统问题域子系统的结构

# 11.6 设计人机交互子系统

在面向对象分析中，已经对用户界面需求作了初步分析，在面向对象设计过程中，要对系统人机交互部分进行详细设计，其中包括指定窗口和报表的形式、设计命令层次等等。

设计人机交互子系统的策略：

- 1 ) 分类用户；
- 2 ) 描述用户；
- 3 ) 设计命令层次；
- 4 ) 设计人机交互类。



# 11.7 设计任务管理子系统

1. 分析并发性

2. 设计任务管理子系统

- 1 ) 确定事件驱动型任务
- 2 ) 确定时钟驱动型任务
- 3 ) 确定优先任务
- 4 ) 确定关键任务
- 5 ) 确定协调任务
- 6 ) 尽量减少任务数
- 7 ) 确定资源需求

# 11.8 设计数据管理子系统

## 1. 选择数据存储管理模式

- 文件管理系统
- 关系数据库管理系统
- 面向对象数据库管理系统

## 2. 设计数据管理子系统

- 设计数据格式
- 设计相应的服务

采用成熟的商品化关系数据库。根据数据库范式设计，保持数据一致性、完整性。

# 11.9 设计类中的服务

1. 确定类中应有的服务

2. 设计实现服务的方法

- 设计实现服务的算法

- 算法复杂度

- 容易理解、容易实现

- 容易修改

- 选择数据结构

- 定义内部类和内部操作

# 11.10 设计关联

## 实现单向关联

关联 → 一个类的属性



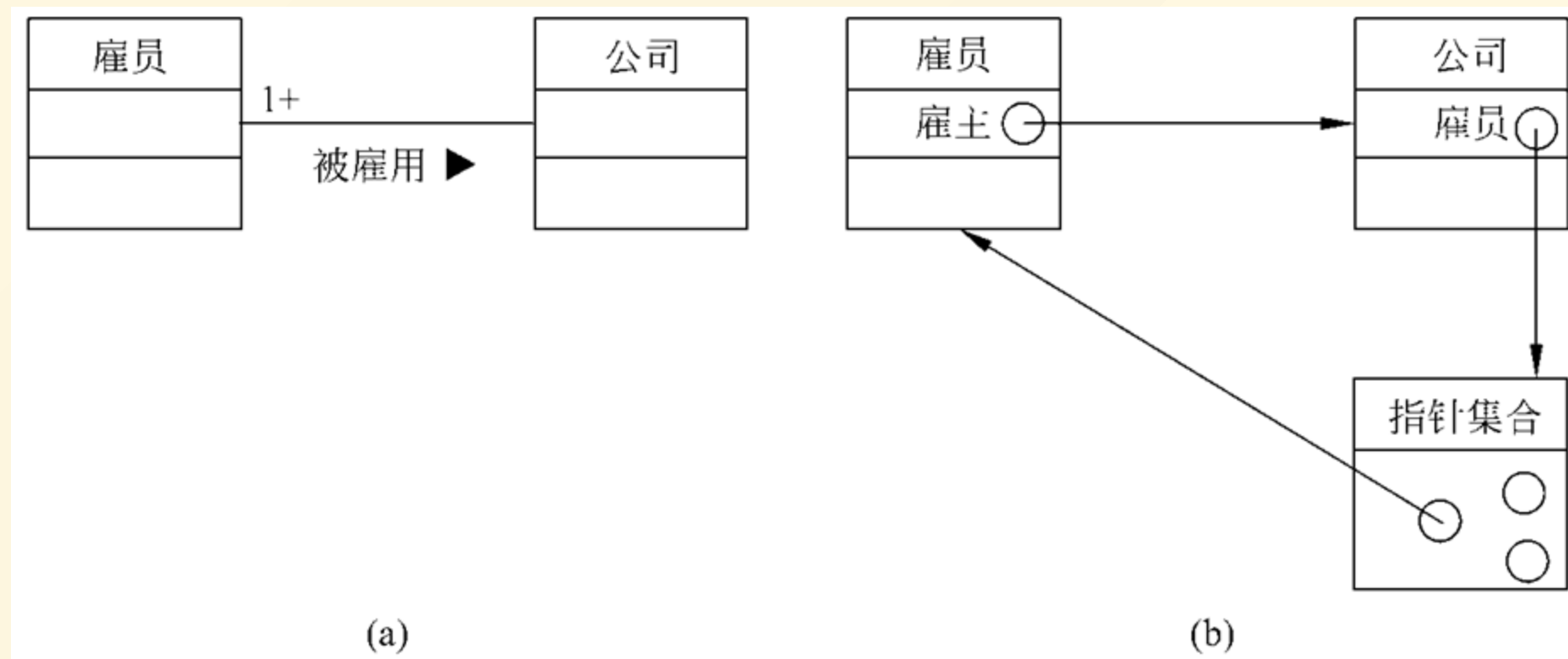
(a)



(b)

# 实现双向关联

关联 → 两个类的属性



# 关联类的实现

- 一对一：与参与关联的任一类合并
- 一对多：与“多”端的类合并
- 多对多：独立的关联类

# 11.11 设计优化

## 提高效率

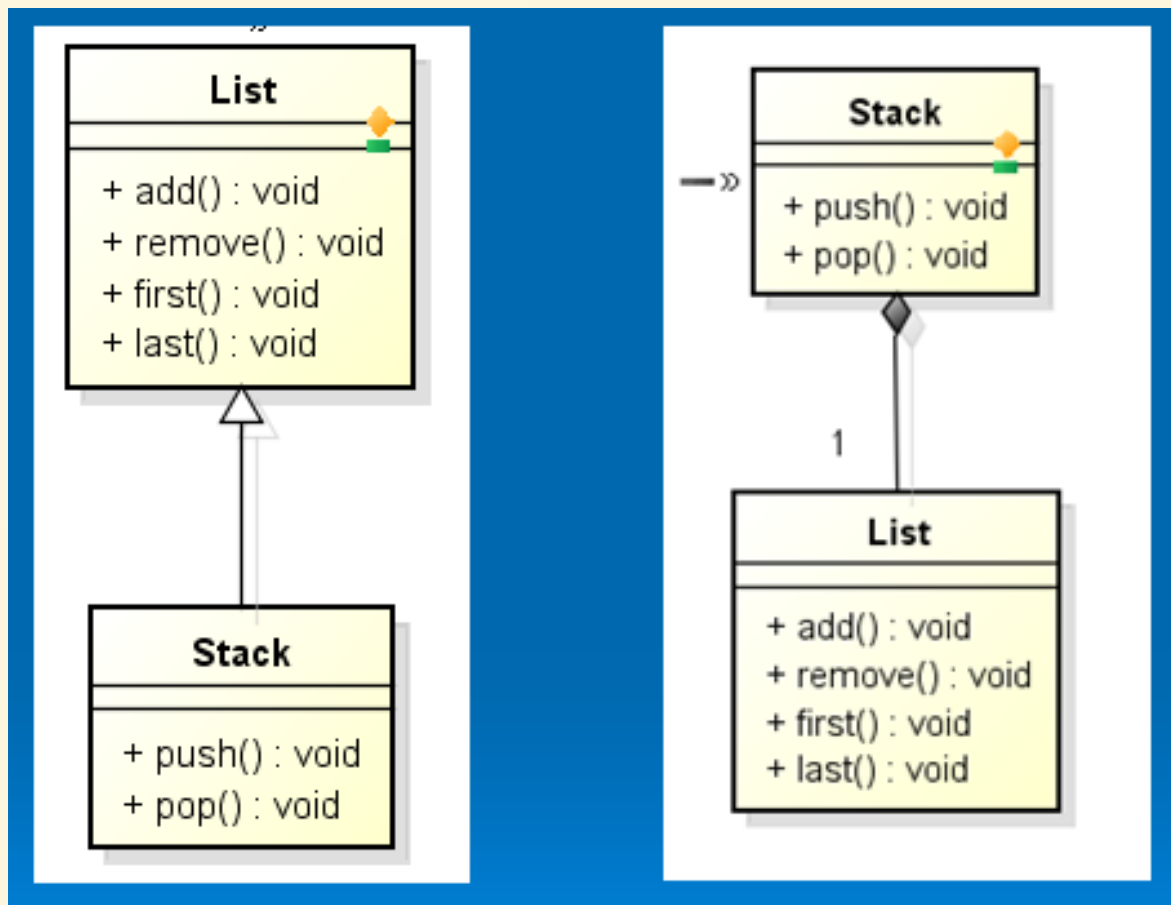
- 增加冗余关联以提高访问效率
  - 采用哈希表
  - 为需要经常查询的对象建立索引
- 调整查询次序
  - 缩小查找范围
- 保留派生属性

# 调整继承关系

- 抽象与具体
  - 具体到抽象：泛化（generalization）
  - 抽象到具体：继承（inheritance）
- 为提高继承程度而修改类定义
  - 不能违背领域知识和常识
  - 确保现有类的协议（即接口）不变
- 利用委托（delegation）实现行为共享  
比如：基于List类实现Stack类



# 利用委托（delegation）实现行为共享



*Favor composition over inheritance*

# 本章重点

- 面向对象设计的准则和启发规则
- 系统分解及各子系统的设计