第二次习题课

2024 12 12

第八章 体系结构设计

第八章复习

软件体系结构定义之一

- 描述一个计算机软件的各个功能模块内部关系以及不同模块之间的关系的构造图（可以是文字的描述，也可以是图表的描述），类似于硬件设计中的原理框图，它是软件流程图的上层抽象。

软件体系结构定义之二

- 一个程序和计算机系统软件体系结构是指系统的一个或者多个结构。结构中包括软件的构件，构件的外部可见属性以及他们之间的关系。

第八章复习

数据设计

- 一个程序和计算机系统软件体系结构是指系统的一个或者多个结构。结构中包括软件的构件，构件的外部可见属性以及他们之间的关系。

第八章复习

体系结构风格的分类

- 1. 以数据为中心的体系结构
- 数据存储驻留在这种体系结构的中心，其他构件经常访问（增删改查）数据存储
- 提高了可集成性，便于现有构件修改和新构件加入

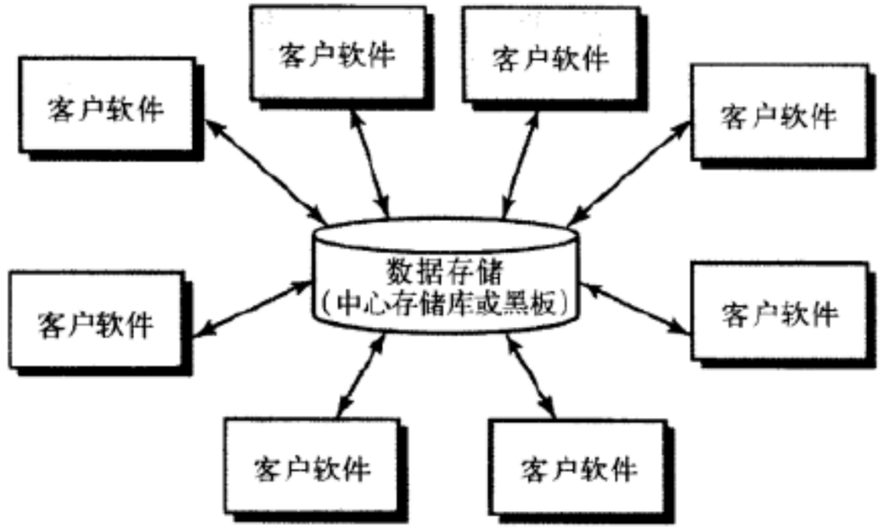
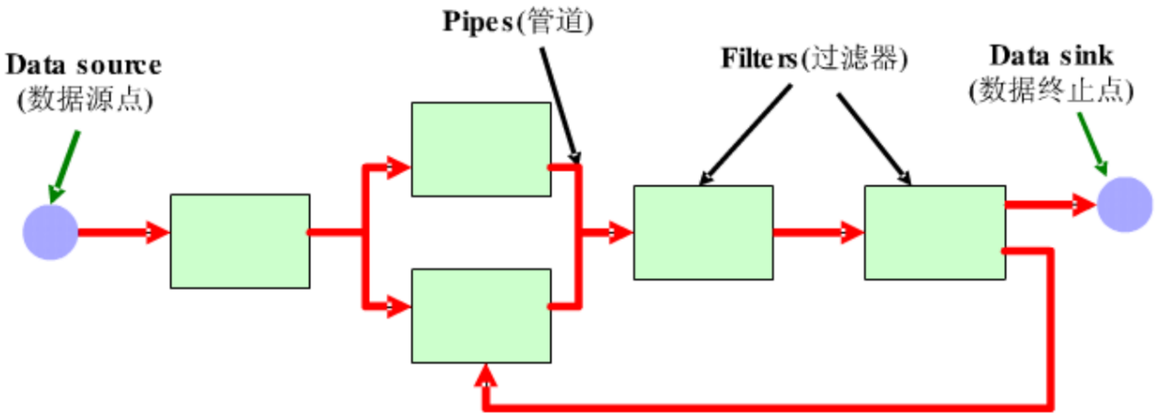


图　　以数据为中心的体系结构

第八章复习

体系结构风格的分类

- 2. 数据流体系结构
- 数据服从输入—变换---输出的简单流程
- 管道和过滤器结构拥有一组被称为过滤器的构件，每个构件独立于上游和下游而工作。
- 两种典型的数据流风格
  - 管道-过滤器（Pipe-And-Filter）
  - 批处理（Batch Sequential)



第八章复习

体系结构风格的分类

- 3. 调用和返回体系结构

- 主程序/子程序体系结构：主程序调用一组程序构件，这组程序构件又去调用其程序构件

- 远程过程调用体系结构：主程序/子程序的构件分布在多个计算机上

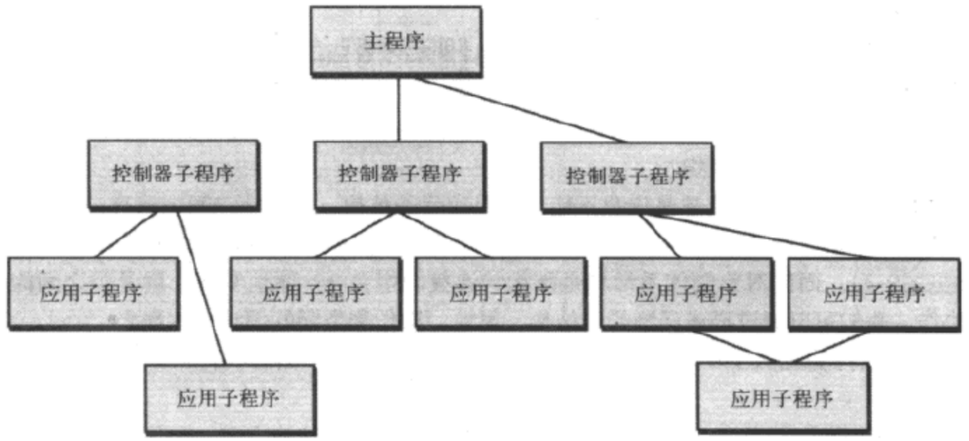


图　　主程序/子程序体系结构

第八章复习

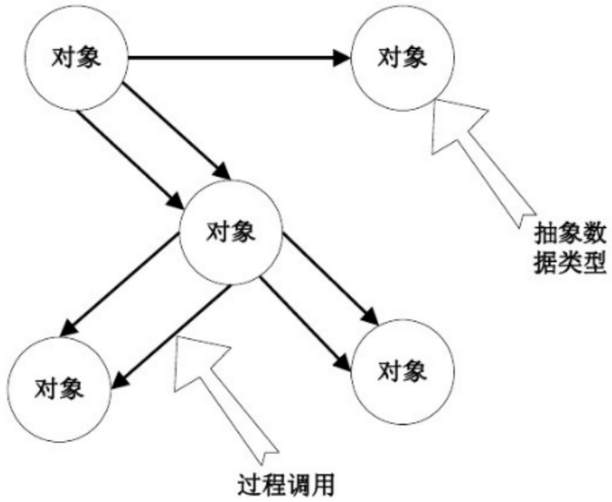
体系结构风格的分类

- 4. 面向对象体系结构

- 封装了数据和必须应用到该数据上的操作，构件间通过信息传递进行通信和合作。

- 优点：对象隐藏；可以将数据存取操作分解成一些交互的代理程序的集合。

- 缺点：对象调用必须有对象的标识；对象循环调用等问题。



第八章复习

体系结构风格的分类

- 5. 层次体系结构

- 每一层为上层服务，并作为下层服务。这样的设计允许将一个复杂问题分解为一个增量步骤序列的实现。由于每一层最多只影响两层，同时只给相邻层提供相同的接口，允许每层用不同的方法实现，同样为软件重用提供了强调的支持。

- 缺点：很难找到一个合适的、正确的层次抽象方法。

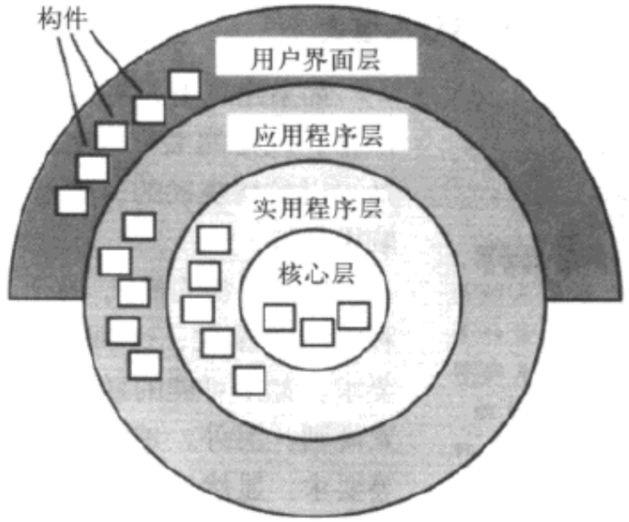
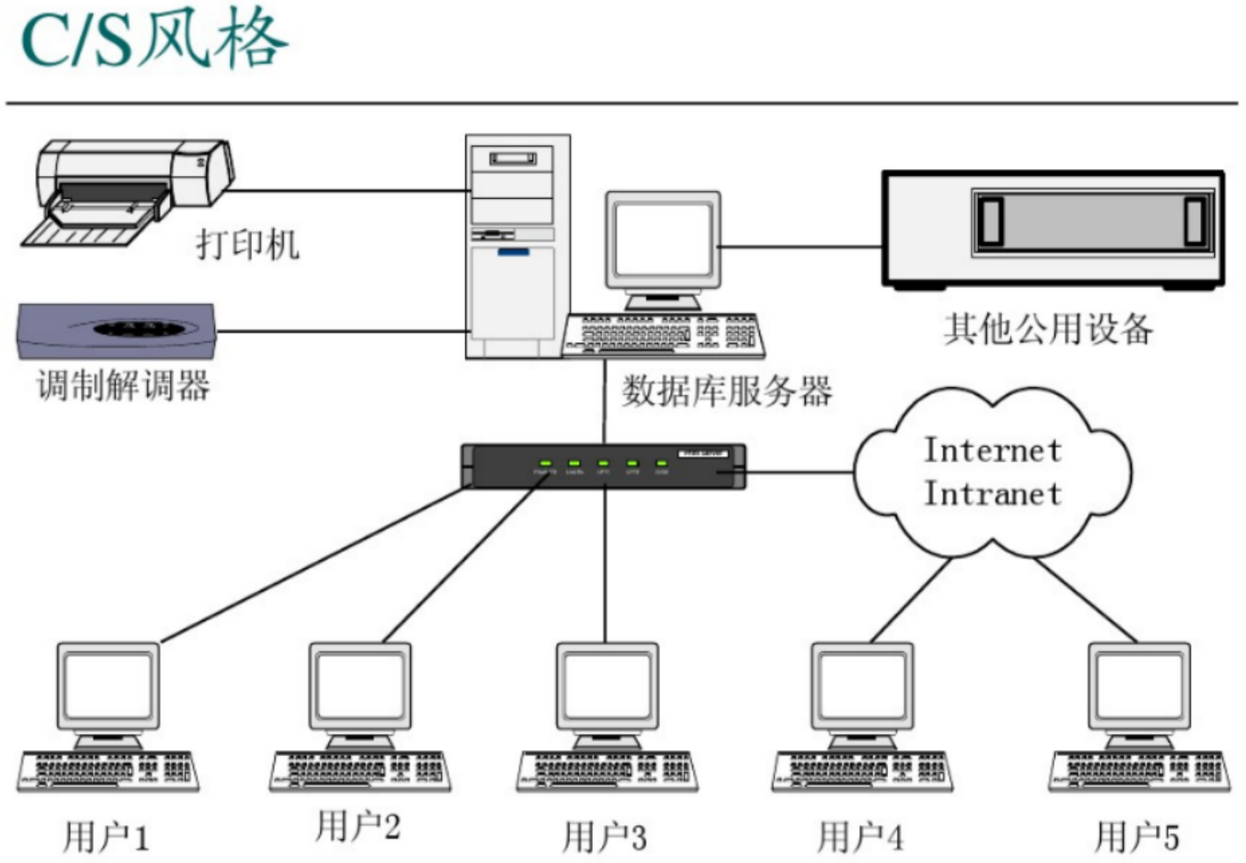


图　　层次体系结构

第八章复习

体系结构风格的分类

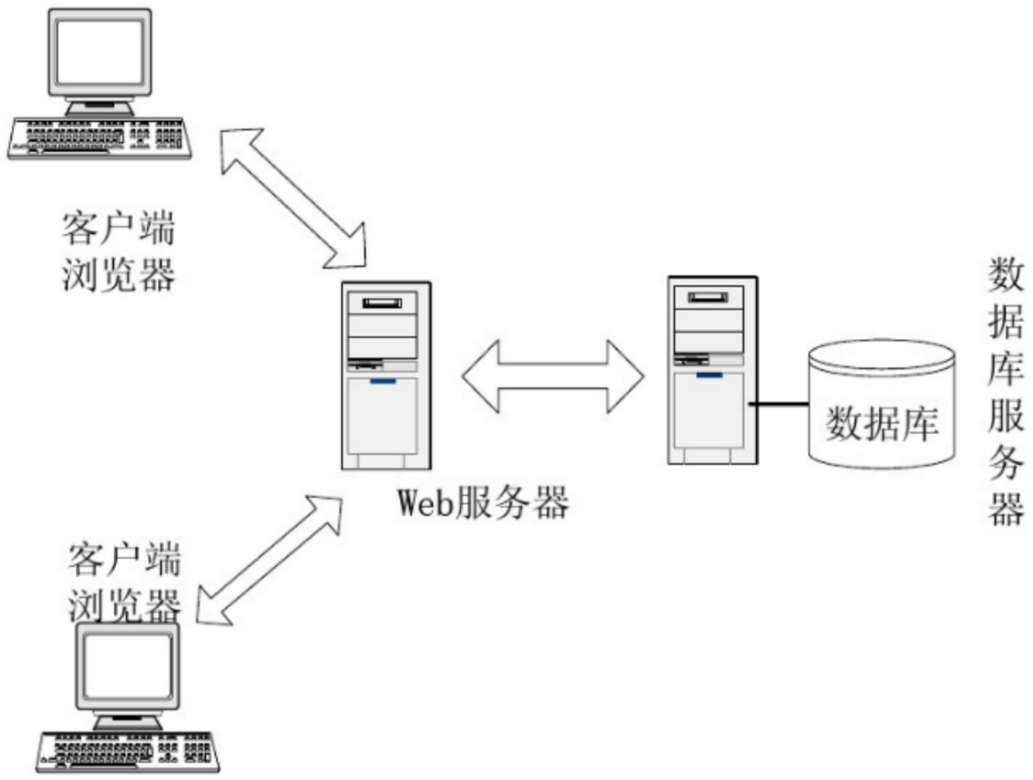
- 6. C/S风格
- 数据库服务器、客户应用程序、网络



第八章复习

体系结构风格的分类

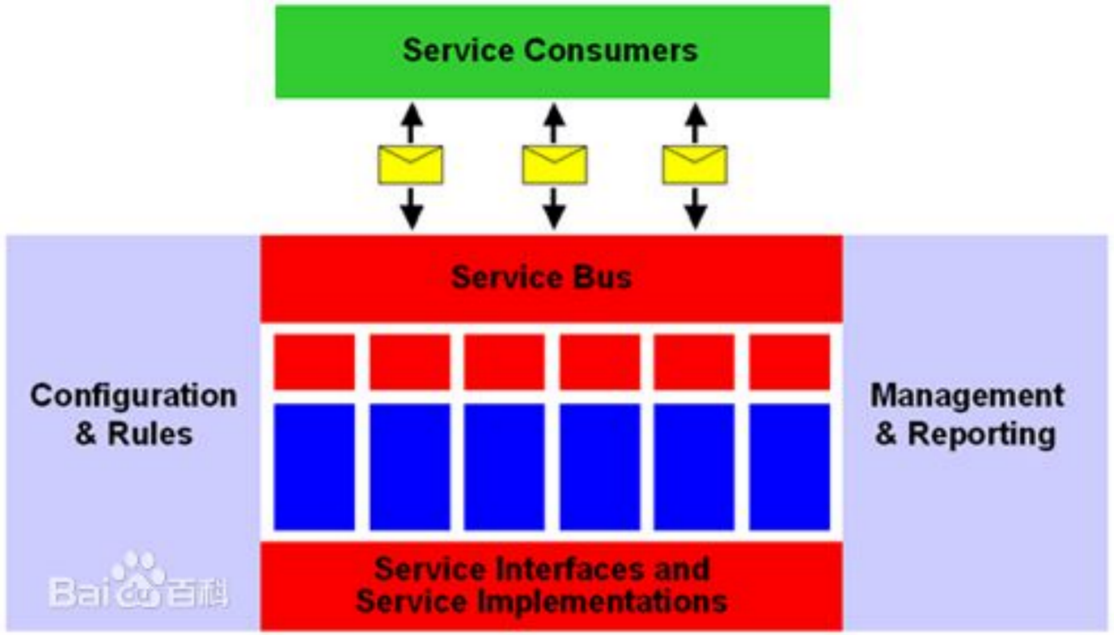
- 7. B/S风格
- 浏览器/Web服务器/数据库服务器



第八章复习

体系结构风格的分类

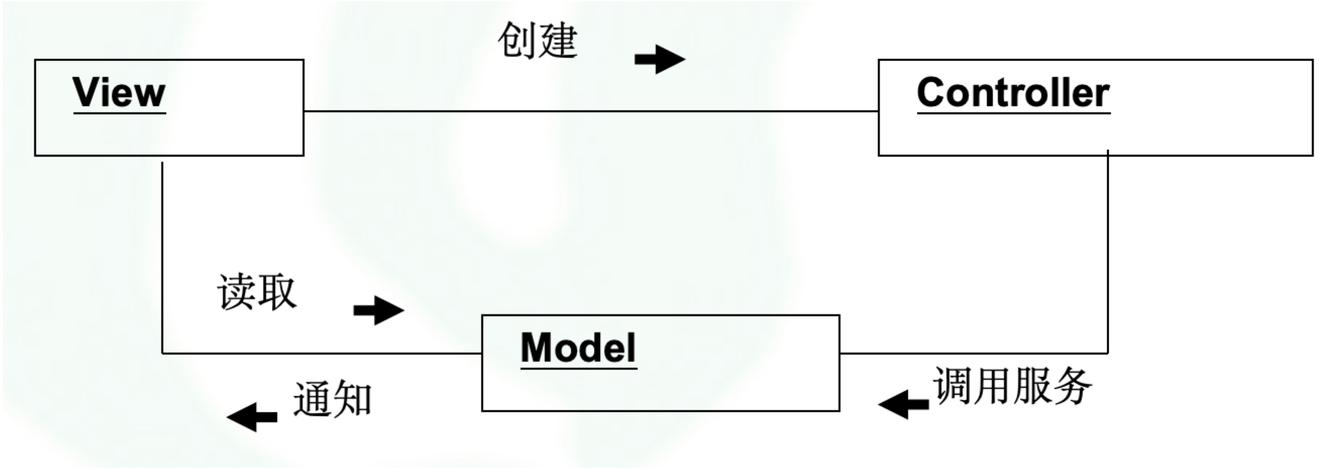
- 8. SOA
- 整个应用程序被设计和实现为一组相互交互的服务



第八章复习

体系结构风格的分类

- 9.MVC架构
- MVC架构作为”组件+交互”的例子, 包含3种组件:Model、View、Controller



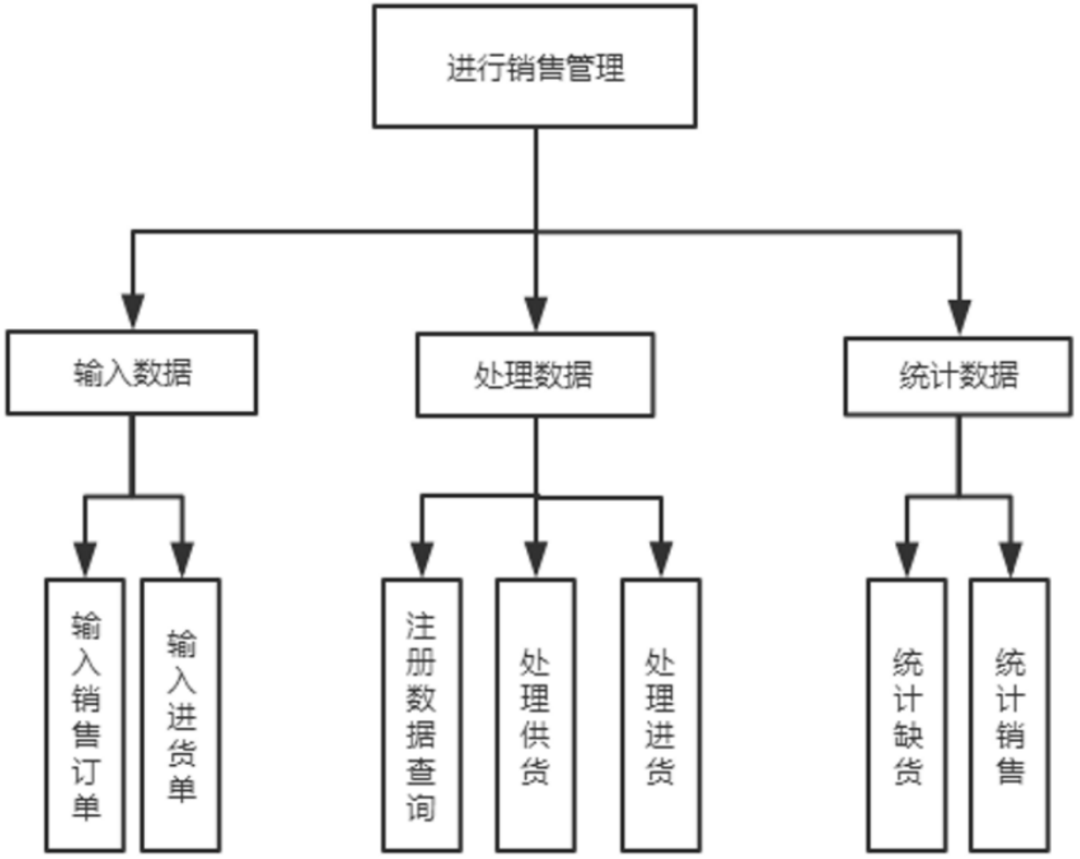
第八章复习

体系结构描述语言

- 体系结构描述语言（architectural description language, ADL）为描述软件体系结构提供一套语义和语法。

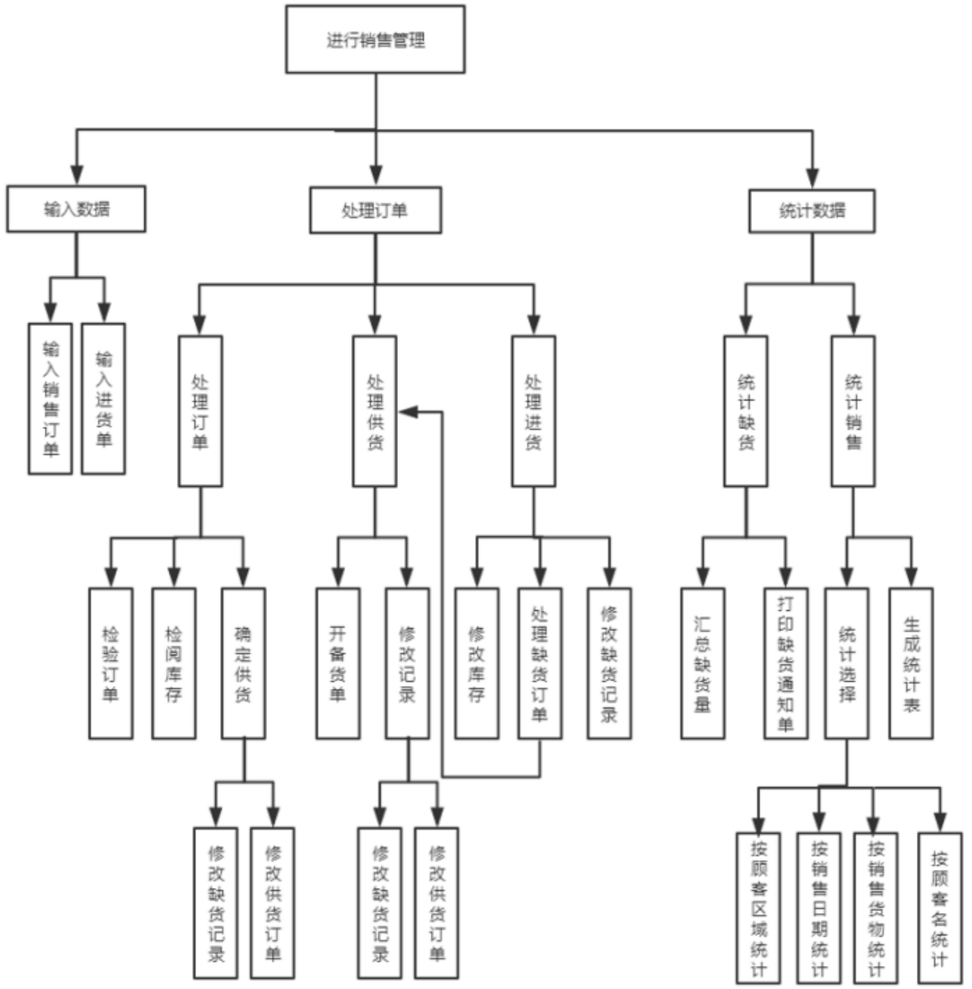
第八章作业

- 根据第5章课件“企业销售管理系统的数据流图”画出系统的软件体系结构。
- 第一级分解



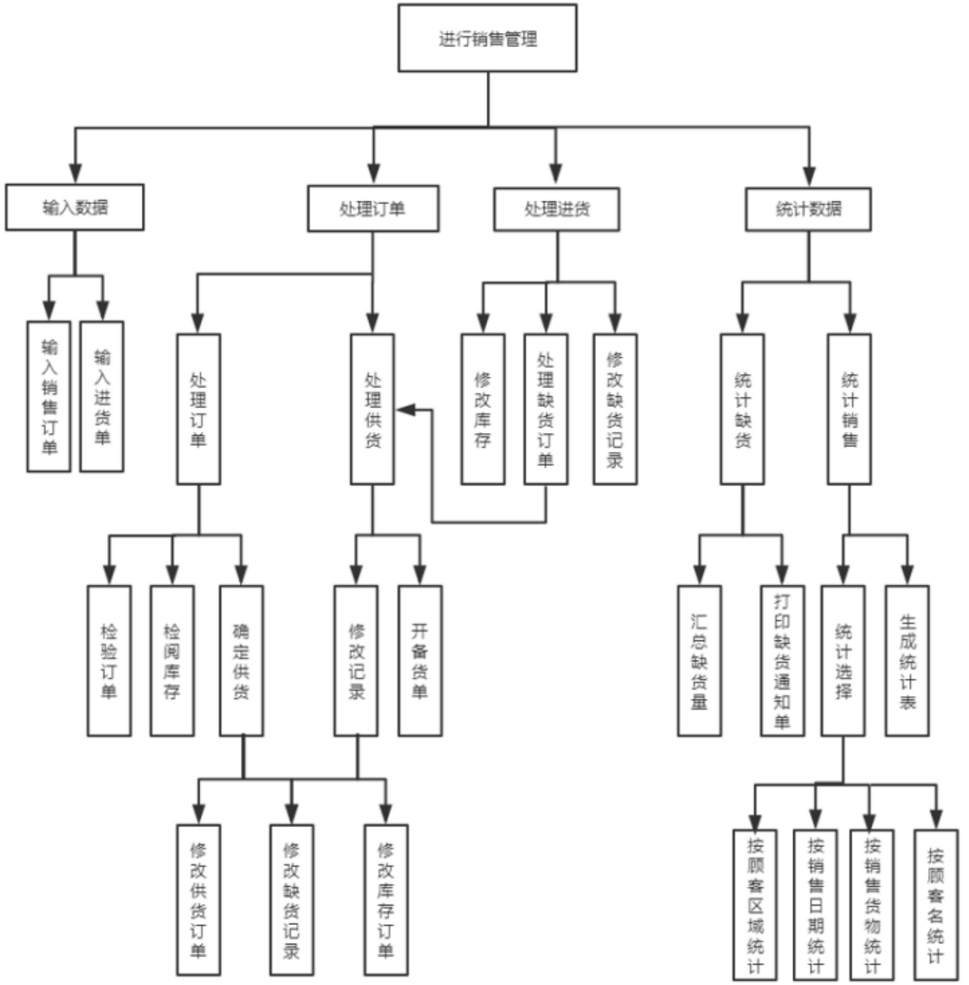
第八章作业

- 根据第5章课件“企业销售管理系统的数据流图”画出系统的软件体系结构。
- 第二级分解



第八章作业

- 根据第5章课件“企业销售管理系统的数据流图”画出系统的软件体系结构。
- 优化后的体系结构图



第九章 构件级设计建模

第九章复习

构件级设计的目的

- 避免高层次的抽象模型向低层次的程序之间转换时容易引入错误的问题。

构件级设计建模时机

- 在体系结构设计第一次迭代后完成

构件级设计形式

- 用编程语言表示
- 用能够容易转化为代码的中间表示(如图形的、表格的或基于文本的)

第九章复习

什么是构件

- 通俗地讲，构件是一段程序，该程序能完成一个相对独立的功能，并有一定的通用性。
- 面向对象观点：在面向对象的设计中，构件指一个协作类的集合。
- 传统观点：程序的一个功能要素，程序由处理逻辑及实现处理逻辑所需的内部数据结构以及能够保证构件被调用和实现数据传递的接口构成。
  - 控制构件----协调不同模块之间的调用
  - 问题域构件-----完成部分或全部用户的需求
  - 基础设施构件----负责完成问题域中的相关处理的功能

第九章复习

基本设计原则

- 开关原则（The Open-Closed Principle ,OCP）
  - 模块应该对外延有开放性，对修改具有封闭性。
  - 即设计者应该采用一种无需对构件自身内部（代码或者内部逻辑）做修改就可以进行扩展（在构件所确定的功能域内）的方式来说明构件。

第九章复习

基本设计原则

- Liskov替换原则（Liskov Subsitution Principle, LSP）
  - 子类可以替换它们的基类
  - 源自基类的任何子类必须遵守基类与使用该基类的构件之间的隐含约定（前置条件、后置条件）。

第九章复习

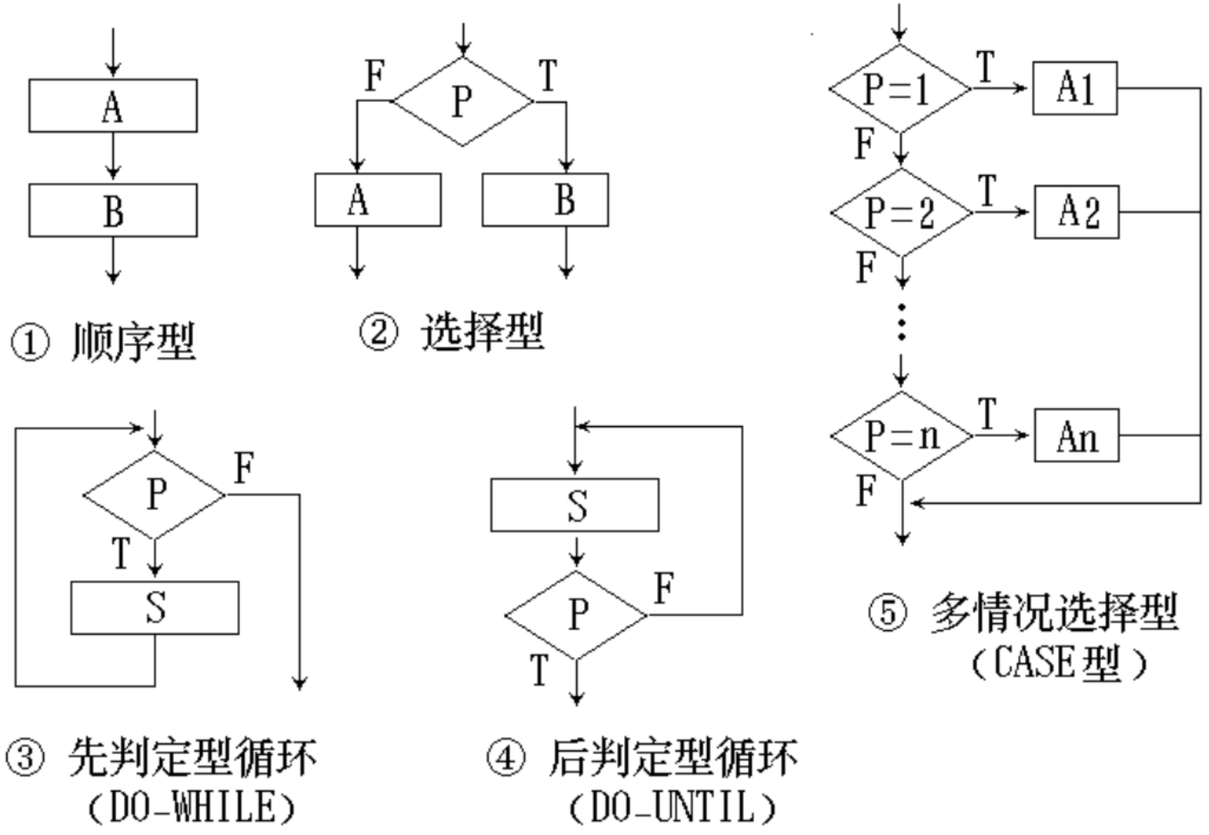
基本设计原则

- 接口分离原则(Interface Segregation principle, ISP)
  - 多个用户专用接口比一个通用接口要好
  - 设计者应该为每一个主要的客户类型都设计一个特定的接口。
  - 如SafeHome中FloorPlan类用于安全和监督功能，两处操作有些不同，监督功能多关于摄像头的操作，定义两个接口。

第九章复习

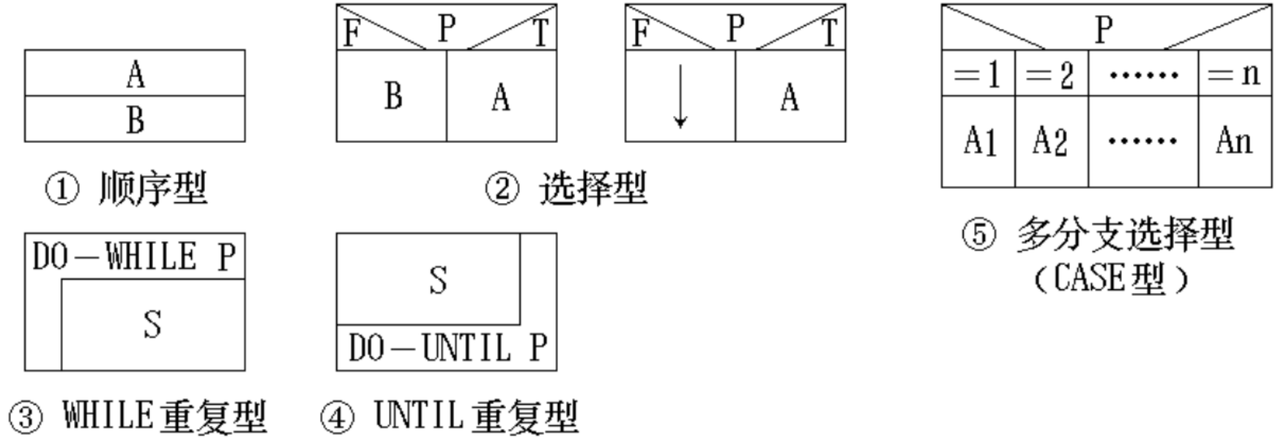
图形化设计表示

- 流程图



第九章复习

图形化设计表示

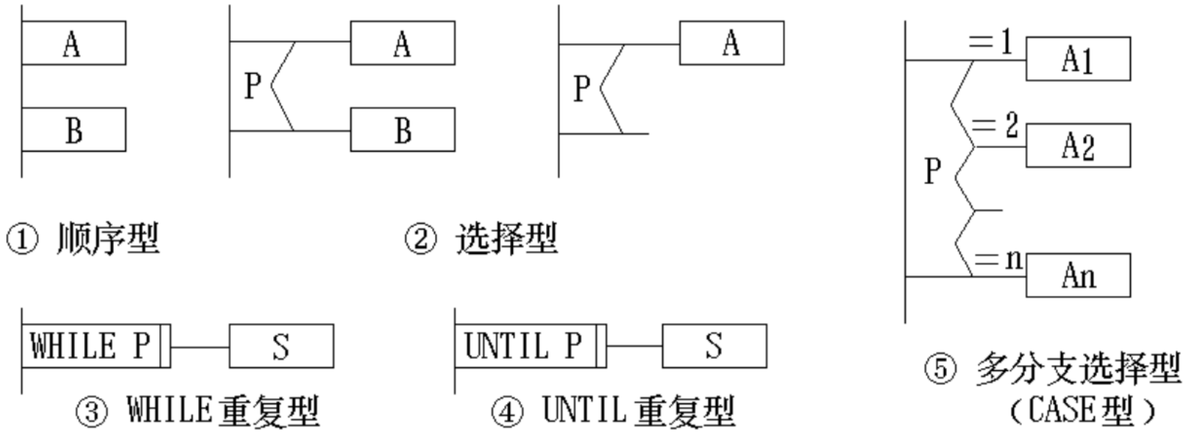


- 盒图，也叫N-S图
  - 没有箭头，不允许随意转移控制；
  - 每个矩形框(Case中条件取值例外)都是一个功能域(即一个特定结构的作用域)，结构表示明确；
  - 局部及全程数据的作用域易见；
  - 易表现嵌套关系(embedded structure)以及模块的层次结构。

第九章复习

图形化设计表示

- 问题分析图PAD
  - 结构清晰，层次分明，易读；
  - 支持逐步求精的设计思想；
  - 容易将PAD自动转换为高级语言源程序。



第九章作业

- 逐步求精和重构是一回事吗？如果不是，他们有什么区别?

- 两者不是一回事。逐步求精是为了避免高层次的抽象模型向低层次的程序之间转换时容易引入的错误，完成构建级设计的过程。而重构是通过调整程序改善软件的质量、性能。一般，在逐步求精过程中也会包括对于某个构件进行重构的过程。而对某个构件进行重构，也有一个逐步求精分析实现的过程。

第九章作业

简述实施构件级设计的步骤

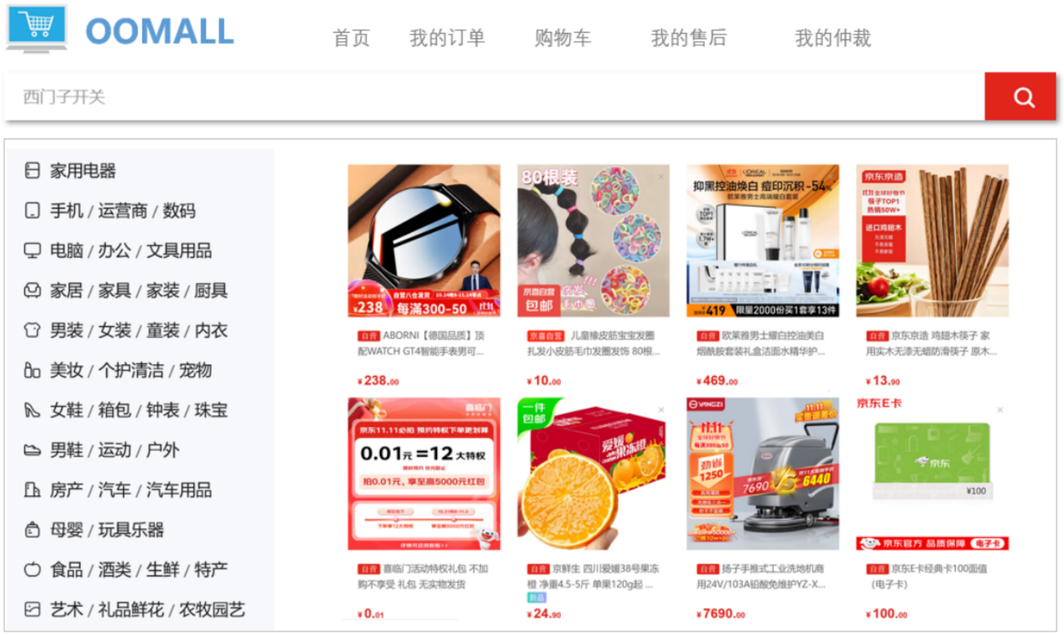
- 标识出所有与问题域相对应的类
- 确定所有与基础设施域相对应的类
- 细化所有不能作为复用构件的类
    - 说明消息的细节流
    - 为每个构件确定适当的接口
    - 细化属性并定义数据类型和结构
    - 描述每个操作中的处理
- 说明持久数据源（数据库或文件）等相关类
- 开发并细化类的行为表示
- 细化部署图
- 反省和检查现有的设计

第十章 完成用户界面设计

第十章作业

完善项目界面原型，下次上课前提交

- 界面设计的黄金规则：
  - 置于用户的控制之下
  - 减少用户的记忆负担
  - 保持界面一致



第十三章 质量保证

第十三章复习

实现软件质量的四大管理和实践活动

- 软件工程方法
- 项目管理技术
  - 交付日期可达、进度依赖关系清楚、进行了风险规划……
- 质量控制活动
  - 包括一套软件工程活动，帮助确保每个工作产品符合其质量目标，如检查代码，应用一系列的测试步骤
- 软件质量保证
  - 建立基础设施，以支持坚实的软件工程方法，合理的项目管理和质量控制活动。

第十三章复习

目标、属性和度量

- 需求质量：软件质量保证必须确保软件团队严格评审需求模型，以达到高水平的质量。
- 设计质量：软件团队应该评估设计模型的每个元素，以确保设计模型显示出高质量，并且设计本身符合需求。SQA寻找能反映设计质量的属性。
- 代码质量：源代码和相关的工作产品必须符合本地的编码标准，并显示出易于维护的特点。SQA应该找出那些合理分析代码质量的属性。
- 质量控制有效性：软件团队应使用有限的资源，在某种程度上最有可能得到高品质的结果。SQA分析在评审和测试上的资源分配，评估是否以最有效的方式进行分配的。

第十三章复习

统计软件质量保证

- 统计质量保证反映了一种在产业界不断增长的趋势：质量的量化。对于软件而言，统计质量保证包含以下步骤：
  - 收集软件的错误和缺陷信息，并进行分类。
  - 追溯每个错误和缺陷形成的根本原因。
  - 使用Pareto原则(80%的缺陷可以追溯到所有可能原因的20%)，将这20%原因分离出来
  - 一旦找出这些重要的少数原因，就可以开始纠正引起错误和缺陷的问题。

第十三章作业

为什么软件工程小组和独立的软件质量保证小组之间的关系经常是紧张的？这种紧张关系是否是正常的?

- 软件工程小组和独立的软件质量保证（SQA）小组之间的紧张关系主要源于角色的对立性：
  - 软件工程小组的目标是按时交付功能完善的产品
  - SQA小组则专注于发现问题、确保质量，这可能会暴露开发中的缺陷或导致交付延迟。
- 这种紧张关系在一定程度上是正常的，因为双方职责不同且相互制约，但它也是必要的，有助于在时间、成本和质量之间找到平衡。有效的沟通与协作可以缓解这种紧张，并推动共同的目标——高质量的软件交付。

第十三章作业

除了可以统计错误和缺陷之外，还有哪些可以统计的软件特征是具有质量意义的？他们是什么？是否可以直接测量？

- 除了统计错误和缺陷外，还有许多具有质量意义的软件特征可以统计
  - 代码复杂度（如圈复杂度，衡量逻辑路径复杂性）
  - 可维护性（包括代码的模块化程度和注释覆盖率）
  - 性能（如响应时间和吞吐量）
  - 可用性（如用户任务完成率和错误率）
  - 安全性（如漏洞数量和安全事件）
  - 可靠性（如平均故障间隔时间）
- 有些特征可以直接测量，如代码复杂度和性能参数；而有些特征需要通过间接方法或指标推算，例如可维护性和可用性通常需要结合多方面数据和专家评估来测量。

第十六章 安全性工程

第十六章作业

考虑你自己的手机APP。首先描述一个APP，然后列出至少3～5种安全风险。

- 以中国银行APP为例，这是一款用于查看账户余额、进行转账支付和管理财务的手机应用。以下是它可能面临的安全风险：
  - 数据泄露：用户登录凭据或交易数据可能被黑客窃取，导致隐私和资金损失。
  - 恶意软件攻击：攻击者可能通过恶意软件窃取敏感信息或劫持交易。
  - 弱加密：如果APP未对传输和存储的数据进行强加密，可能被中间人攻击或暴力破解。
  - 未授权访问：APP可能存在未处理的漏洞，允许未经授权的用户访问账户信息。
  - 钓鱼攻击：用户可能被诱骗登录伪装的APP或网站，从而泄露敏感信息。
  - 这些风险可以通过加强身份验证、采用安全加密、定期更新软件和教育用户防范意识来缓解。