

# 软件工程

## ▼ I. 软件工程概述

### 软件

软件 = 程序 + 数据 + 文档

- 数据：使程序能够适当处理信息的数据结构
- 程序：能够完成预定功能和性能的可执行指令序列
- 文档：开发、使用和维护过程程序所需要的图文资料

The screenshot shows a mobile phone screen with a presentation slide. The top bar includes the time (13:59), date (6月24日周一), battery level (54%), and signal strength. The slide has a dark header with white text: '1. 软件工程概述'. Below the header, there is a section titled '1.1 软件与软件危机' with a blue double arrow icon. On the right side of the slide, there is a logo for '期末加油站' (Exam加油站) featuring a graduation cap icon. The main content area contains the definition of software: '软件 = 程序 + 数据 + 文档'. It also defines three components: '数据' (data), '程序' (program), and '文档' (documentation). A dashed-line box highlights a question: '例题 1：在软件工程中，以下哪个选项最准确地描述了软件的本质？' followed by four options: A. 软件仅指计算机程序代码; B. 软件包括程序代码和相关文档; C. 软件是运行在硬件上的指令集合; D. 软件是一个广泛的概念，不仅包括程序代码和文档，还包括相关数据和设计、测试和维护过程.

### 软件的特点

1. 软件是复杂性的
2. 软件的成本高昂

- 3. 软件开发未摆脱手工开发方式
  - 4. 软件维护与硬件有本质差别，难度较高
  - 5. 软件开发不同于传统的硬件制造过程
  - 6. 软件是一种逻辑实体，无磨损性
- 软件危机：在计算机软件开发和维护过程中所遇到的一系列严重问题。

## » 1.1 软件与软件危机



### 软件的特点：

1. 软件是复杂性的
2. 软件的成本高昂
3. 软件开发未摆脱手工开发方式
4. 软件维护与硬件有本质差，维护难度高
5. 软件开发不是传统硬件制造过程
6. 软件是一种逻辑实体，无磨损性

例题 2：软件与硬件的主要区别是什么？

- A. 软件可见而硬件不可见
- B. 软件可以触摸而硬件不可以
- C. 软件是无形的，而硬件是有形的实体
- D. 软件不需要电力即可运行

例题 3：下列哪项最准确地反映了软件的特征？

- A. 软件不会磨损，但会过时
- B. 软件的开发成本通常低于硬件
- C. 软件一旦编写，就无需更改
- D. 软件无法复制或共享

软件危机：在计算机软件开发和维护过程中所遇到的一系列严重问题。

## 软件危机

- 软件危机的两方面内容：
  1. 如何开发软件以满足对软件长久的需求
  2. 如何维护数量不断膨胀的现有软件
- 软件危机的表现：
  1. 对软件开发成本和进度估算不准确
  2. 用户对已完成软件不满意
  3. 软件质量不可靠
  4. 软件不可维护
  5. 没有适当的文档资料

- 6. 软件成本在计算机系统中占比例逐年上升
- 7. 软件开发生产率低
- 软件危机的原因：
  1. 用户需求不明确
  2. 缺乏正确的理论指导
  3. 软件开发规模越来越大
  4. 软件开发复杂度越来越高
- 软件不好，用户体验问题，等字眼均属于软件危机

## » 1.1 软件与软件危机



### 软件危机的两方面内容：

1、如何开发软件，以满足对软件日益增长的需求      2、如何维护数量不断膨胀的已有软件

### 软件危机的表现：

- 对软件开发成本和进度估算不准确
- 用户对已完成软件不满意
- 软件质量不可靠
- 软件不可维护
- 没有适当文档资料
- 软件成本在计算机系统中所占比例逐年上升
- 软件开发生产率低

### 软件危机的原因：

1. 用户需求不明确 2. 缺乏正确的理论指导 3. 软件开发规模越来越大 4. 软件开发复杂度越来越高

## 消除软件危机的途径

1. 对计算机软件有正确的认识
  2. 借鉴长期工作者的经验
  3. 使用计算机辅助工具
  4. 探索更好，更有效的管理措施
- 软件工程就是为了解决软件危机

**消除软件危机的途径：**

1. 对计算机软件应该有正确认识
2. 吸取借鉴人类长期从事各种工程项目积累的原理、概念、技术和方法
3. 积极开发和使用计算机辅助开发工具
4. 探索更好更有效的管理措施和手段对开发过程进行控制和管理

---- 软件工程就是为了解决软件危机

**例题4：“软件危机”是指( )**

- A. 计算机病毒的出现。
- B. 利用计算机进行经济犯罪活动。
- C. 软件开发和维护中出现的一系列问题。
- D. 人们过分迷恋计算机软件系统

## 软件工程

- 定义：采用工程的概念、原理、技术和方法来开发和维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，经济的开发出高质量的软件并维护它。
- 三个要素：方法、工具和过程
  - 方法：完成软件开发各项任务的技术方法，回答“怎么做”
  - 工具：为运用方法提供的自动或半自动软件工程支撑环境
  - 过程：为了获得高质量软件所需要完成的人物框架，回答“何时做”

**软件工程：**

采用工程的概念、原理、技术和方法来开发与维护软件，把经过时间考验而证明正确的管理技术和当前能够得到的最好的技术方法结合起来，经济的开发出高质量的软件并维护它。

**三个要素：**方法、工具和过程

方法：完成软件开发各项任务的技术方法，回答“怎么做”

工具：为运用方法提供的自动或半自动软件工程支撑环境

过程：是为了获得高质量软件所需要完成的一系列任务框架，回答“何时做”

**例题 1：**软件工程三要素模型（SE三要素）不包括以下哪项？

- A. 方法
- B. 工具
- C. 经济成本
- D. 过程

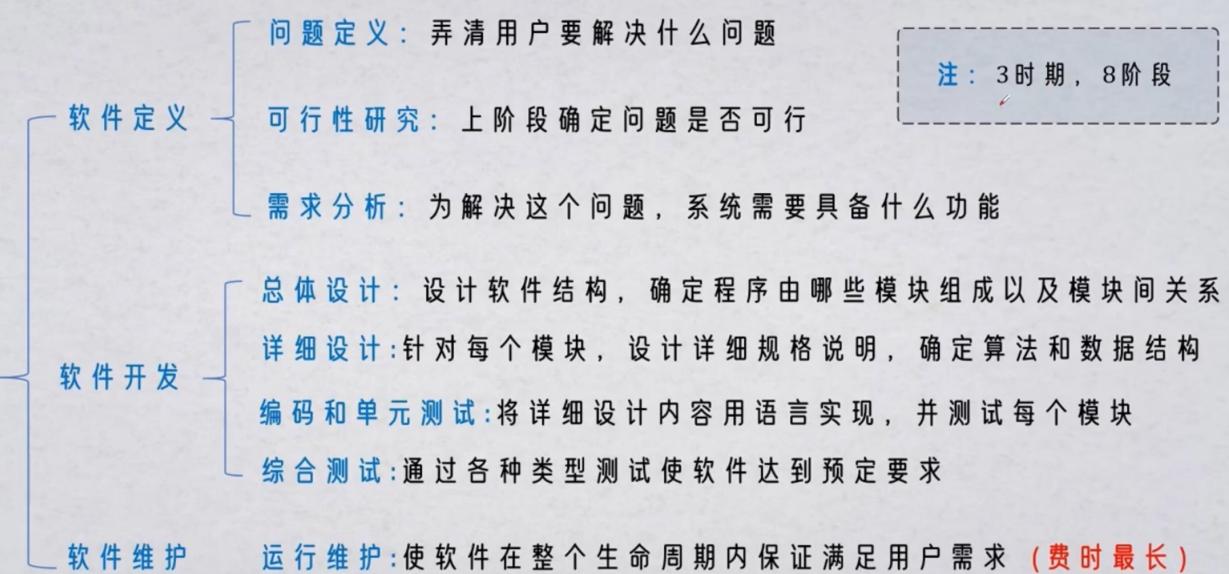
**例题 2：**下列哪个选项最好地描述了软件工程的过程模型？

- A. 编码 - 编译 - 测试 - 部署
- B. 需求分析 - 设计 - 编码 - 测试 - 维护
- C. 只有编码和测试两个阶段
- D. 问题定义 - 解决方案探索 - 解决方案实施

## 软件生命周期

- 软件定义
  - 问题定义：弄清用户要解决什么问题
  - 可行性研究：上阶段确定问题是否可行
  - 需求分析：为解决这个问题，系统需要具备什么功能
- 软件开发
  - 总体设计：设计软件结构，确定程序由哪些模块组成以及模块间关系
  - 详细设计：针对每个模块，设计详细规格说明，确定算法和数据结构
  - 编码和单元测试：将详细设计内容用语言实现，并测试每个模块
  - 综合测试：通过各种类型测试使软件达到预定要求
- 软件维护
  - 运行维护：使软件在整个生命周期内保证满足用户需求（费时最长）

软件生命周期



## 软件过程

### 瀑布模型

- 将软件生命周期的各个活动规定为固定顺序连接的阶段工作，最终得到软件产品
- 优点：每个阶段，必须提交文档并验收，有利于大型软件开发过程中的人员的组织和管理。
- 缺点：瀑布模型各个阶段不能回溯修改，需要严格按照模型进行开发，缺乏灵活性。

### 软件过程：

为了获得高质量软件所需要完成的一系列任务框架。通常用软件生命周期模型描述软件过程。

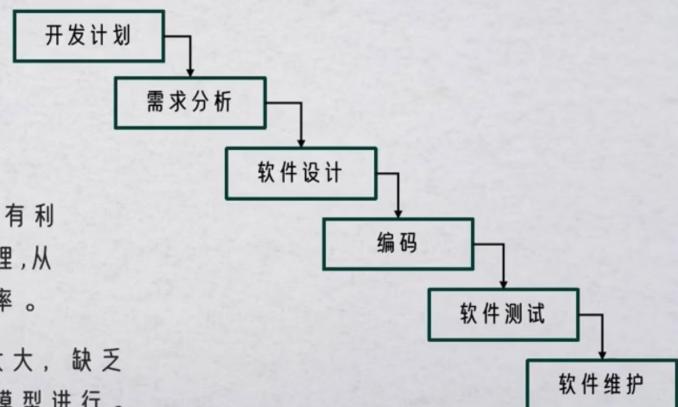
主要包含：瀑布模型、增量模型、快速原型模型、螺旋模型和喷泉模型。

#### 瀑布模型：

将软件生存周期的各项活动规定为依照固定顺序连接的若干阶段工作，最终得到软件产品。

**优点：**每个阶段，必须提交文档并验收，有利于大型软件开发过程中人员的组织、管理，从而提高了大型软件项目开发的质量和效率。

**缺点：**开发过程一般不能逆转，否则代价太大，缺乏灵活性。实际的项目开发很难严格按该模型进行。



### 增量模型

- 将系统分成各个子集，独立开发，独立交付
- 优点：短时间内可提交完成部分功能，逐渐增加产品功能使用户适应产品更快
- 缺点：增量构建、分化以及集成困难（尤其并行开发），容易退化为边做边改模型

### 软件过程：

为了获得高质量软件所需要完成的一系列任务框架。通常用软件生命周期模型描述软件过程。

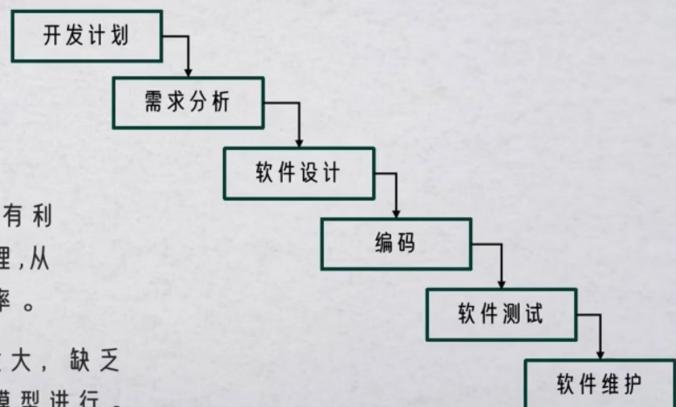
主要包含：瀑布模型、增量模型、快速原型模型、螺旋模型和喷泉模型。

#### 瀑布模型：

将软件生存周期的各项活动规定为依照固定顺序连接的若干阶段工作，最终得到软件产品。

**优点：**每个阶段，必须提交文档并验收，有利于大型软件开发过程中人员的组织、管理，从而提高了大型软件项目开发的质量和效率。

**缺点：**开发过程一般不能逆转，否则代价太大，缺乏灵活性。实际的项目开发很难严格按该模型进行。



### 快速原型模型

- 先建立一个软件的简易版模型，然后根据原型进行软件生命周期，完成之后再根据变化需求改正
- 优点：开发的软件产品通常满足用户需求，开发基本是线性过程
- 缺点：准确原型设计困难，不利于开发人员创新

### 螺旋模型

- 在建立快速原型模型之前先进行风险分析，软件生命周期结束后再次进入风险分析进行循环
- 优点：利于把软件质量作为开发目标，维护和开发部分分开
- 缺点：风险分析难度高

## >>> 1.4 软件过程



### 快速原型模型：

快速建立可运行的程序，它完成的功能往往是最终产品功能的一个子集。

**优点：**1. 开发的软件产品通常满足用户需求  
2. 软件产品开发基本是线性过程

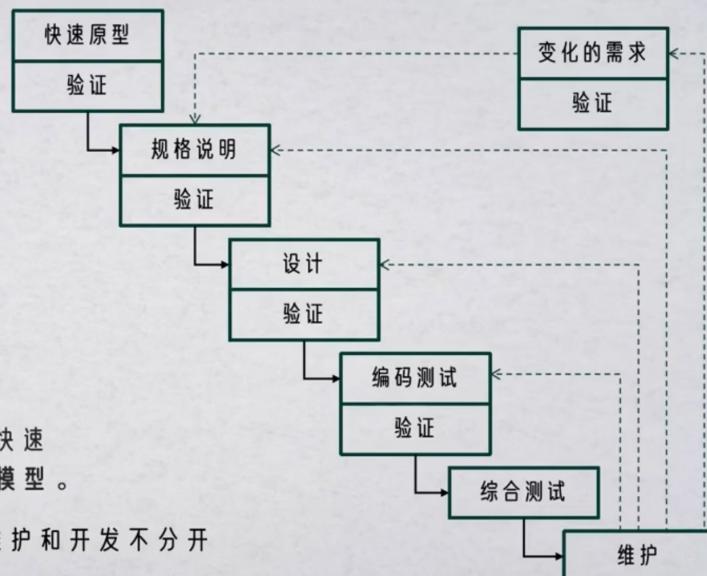
**缺点：**1. 准确原型设计困难 2. 不利于开发人员创新

### 螺旋模型：

在每个阶段之前都增加了风险分析过程的快速原型模型看作增加了风险分析的快速原型模型。

**优点：**1. 利于把软件质量作为开发目标 2. 维护和开发不分开

**缺点：**1. 风险估计困难



### 喷泉模型

- 适合面向对象开发，各个阶段没有明显的分界

## >>> 1.4 软件过程



### 喷泉模型：

喷泉模型是一种以用户需求为动力，以对象作为驱动的模型，适合于**面向对象**的开发方法，在设计活动结束后才开始编码活动，而是允许各开发活动交叉、迭代地进行。

**注：**各阶段无明显分界，可以迭代交叉



## ▼ 小结

The screenshot shows a slide titled '课程小结' (Course Summary) with a blue double arrow icon. On the right is a logo for '期末加油站' (Final Exam加油站) featuring a graduation cap and the text 'QIMIZIAYOUZHAN'. The slide content includes:  
课程小结：  
1、软件的概念和特点，软件危机及其表现的形式。  
2、软件工程的概念及三要素  
3、软件生命周期的三时期八阶段。  
4、各类的软件过程，包含：顺序、面向对象、增量模型、快速原型模型、螺旋模型和喷泉模型。

## ▼ II. 可行性研究任务

### 可行性分析任务

- 用最小的代价在最小的时间内确定问题是否能够解决
- 了解客户的要求及实现环境
- 从经济、技术和社会因素等三方面研究并论证本软件项目的可行性
  - 经济可行性：经济效益是否大于开发成本
  - 技术可行性：现有技术能否实现，系统操作方式是否可行
  - 法律、社会效益

## >>> 2.1 可行性研究任务



### 可行性分析任务：

用最小的代价在最小的时间内确定问题**是否**能够解决。

可行性研究的主要任务是“了解客户的要求及现实环境，从**经济、技术和社会因素**等三方面研究并论证本软件项目的可行性。

**经济可行性**：经济效益是否大于开发成本

**技术可行性**：现有技术能否实现，系统操作方式是否可行（操作可行性）

**其他可行性**：法律、社会效益

**例题1**：可行性研究通常包括哪些要素（）

- A. 经济可行性
- B. 技术可行性
- C. 法律可行性
- D. 所有以上答案

**例题2**：在软件工程中，以下哪项是进行可行性研究的目的（）

- A. 确定项目的预算和时间表
- B. 评估项目的技术难度
- C. 分析项目是否值得投资
- D. 所有以上答案

## 数据流图

绘制思路：

- 确定系统输入和输出，源点以及重点
- 画出系统顶层数据流图
- 画出分层数据流图

## >>> 2.2 数据流图与数据字典



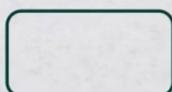
数据流图：

描述信息流和数据从输入到输出过程所经受的变换。没有任何具体物理部件只是描绘数据在软件中流动和被处理的逻辑过程。

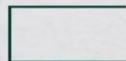
基本图形元素：



源点 / 终点



变换数据的  
处理



数据存储



数据流

## >>> 2.2 数据流图与数据字典



数据流图绘制思路：

1. 确定系统输入输出、源点以及终点
2. 画系统顶层数据流图（基本系统模型）
3. 画出分层数据流图（细化基本系统模型，描述系统的主要功能）

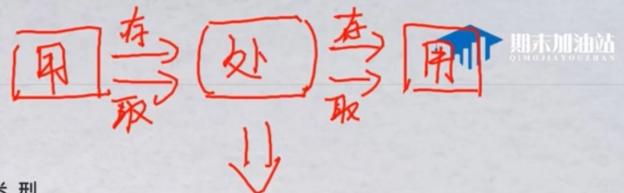
例题 1：数据流图主要用来表示什么？

- A. 数据对象的属性
- B. 数据在系统中的流动
- C. 系统的硬件配置
- D. 软件的算法逻辑

例题 2：在数据流图中，如果一个处理过程需要来自多个数据源的输入，这通常意味着什么？

- A. 数据流图有误，因为一个处理不能有多个输入
- B. 该处理过程可能需要进行数据合并或整合
- C. 每个输入都必须通过不同的路径到达最终处理过程
- D. 该处理过程一定涉及复杂的业务规则

## >>> 2.2 数据流图与数据字典



例题3：某银行的储蓄系统的工作流程如下：

该系统分为了存款单据和取款单据两种类型。

如果是存款单据则记录存款人姓名、身份证号、存款日期、到期日期、利率及密码信息，并打印出存款单据返还给储户。

如果是取款单据，则系统首先核对密码（存款时上传的密码），若密码正确，则系统计算利息并印出利息清单给储户。请用数据流图描绘本系统



注：画图题请按照步骤  
进行思考

## 数据字典

- 选一： $[a|b]$  表示从a和b中选一个
- 重复： $x\{a\}y$ , 将a重复从x到y遍,  $x\{a\}x$ , 将a重复x遍
- 抽象需要从最底层开始定义

## >> 2.2 数据流图与数据字典



### 数据字典：

数据字典是关于数据的信息集合，即对数据流图中包含的所有元素定义的集合。

符号表示： = + [ ] { } ( )

等价 连接 选一 重复 可选

**例题 4：**某校的可用电话号码有以下几类：电话分为校内和校外电话。校内电话号码由 4 位数字组成，第一位数字不是 0。校外电话又分为本市电话和外地电话两类。拨校外电话需要先拨 010，若是本市电话则接着拨 8 位数字，若是外地电话则拨 3 位区码（非零数）后再拨 8 位电话号码。请写出其数据字典表示。

## >> 2.2 数据流图与数据字典



### 数据字典：

数据字典是关于数据的信息集合，即对数据流图中包含的所有元素定义的集合。

符号表示： = + [ ] { } ( )

等价 连接 选一 重复 可选

**例题 4：**某校的可用电话号码有以下几类：电话分为校内和校外电话。校内电话号码由 4 位数字组成，第一位数字不是 0。校外电话又分为本市电话和外地电话两类。拨校外电话需要先拨 010，若是本市电话则接着拨 8 位数字，若是外地电话则拨 3 位区码（非零数）后再拨 8 位电话号码。请写出其数据字典表示。

电话 = [校内 | 校外] 校外 = [本市 | 外地] 外地 = 010 + 3 非零数 {3} + 8 数 {8}   
校内 = 非零数 {3} + 8 数 {8} 本市 = 010 + 8 数 {8} 非零数 = [1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]   
数 = [0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9]

## >>> 2.3 需求分析



### 需求分析：

需求分析是为用户所看到的系统建立一个概念模型（需共同拟定），是对需求的抽象描述。

### 模型分类：

- **功能模型**：(数据流图)：描绘数据在系统中流动时被处理的逻辑过程，指明系统具有的变换数据的功能。
- **数据模型** (实体-联系图)：描绘数据对象及数据对象之间的关系
- **行为模型** (状态转换图)：描绘系统的各种行为模式在不同状态间转换的方法

### 例题1：以下哪个是需求分析的挑战？

- A. 需求通常是明确和不变的
- B. 需求分析通常不需要用户参与
- C. 需求分析可以完全依赖书面文档完成
- D. 需求可能来自多个不同的利益相关者，有时会存在冲突

## E-R图

## >>> 2.4 E-R图



### 实体联系图(E-R图)：

实体：描述数据对象。



属性：描述数据对象的性质。



联系：描述数据对象之间的交互方式。



- 一对一联系 1:1

- 一对多联系 1:N

- 多对多联系 M:N

例题1：一个学生可选修多门课，一门课有若干学生选修；  
一个教师可讲授多门课，一门课仅有一个教师讲授；  
学生的属性有学号、姓名；教师的属性有教师编号，教师姓名；课程的属性有课程号、课程名。请画出该系统  
E-R图

## >>> 2.4 E-R 图

实体联系图(E-R图):

实体:描述数据对象。

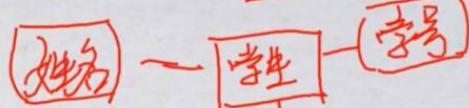
属性:描述数据对象的性质。

联系:描述数据对象之间的交互方式。

- 一对一联系 1:1
- 一对多联系 1:N
- 多对多联系 M:N

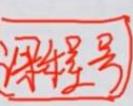


对:学,深,教



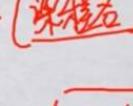
M

N



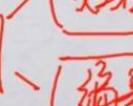
M

N



M

N



M

N



例题 1: 一个学生可选修多门课, 一门课有若干学生选修;

一个教师可讲授多门课, 一门课仅有 1 个教师讲授;

学生的属性有学号、姓名; 教师的属性有教师编号, 教

师姓名; 课程的属性有课程号、课程名。请画出该系统

E-R 图

M:N

1:N

## 状态转移图

## >>> 2.5 状态转移图

状态转移图:

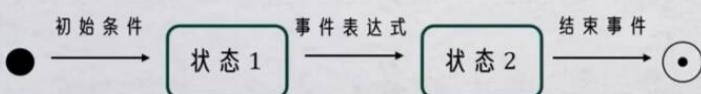
通过描绘系统的状态及引起系统状态转换的事件, 来表示系统的行为。

状态: 系统的行为模式, 包括初态 ●、终态 ○、中间状态 □

事件: 是指在某个特定时刻发生的事情, 即对系统从一个状态转换到另一个状态的事件抽象。

状态图中两个状态之间带箭头的连线称为状态转换, 箭头指明了转换方向。

箭头线上一般需标出触发转换的事件表达式。如果在箭头线上未标明事件, 则表示在源状态的内部活动执行完之后自动触发转换。



注: 状态转移图只能有一个初态, 但可有多个终态

## >>> 2.5 状态转移图

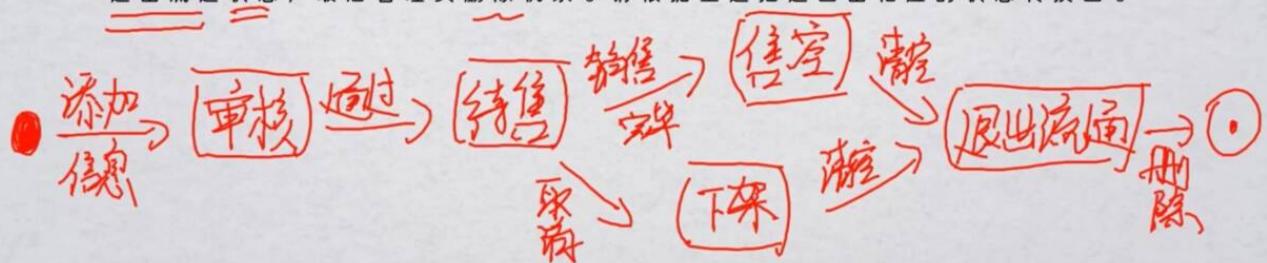


例题1：机票预定系统中涉及的状态是机票的状态。

①

当新的航空公司入驻之后，系统管理员会向数据库中添加机票相关信息该机票进入审核状态，当该机票审核通过后，则处于待售状态；

在销售过程中，如果该机票全部销售完毕，则进入售罄状态；销售过程中，如果航班取消，机票会进入下架状态，机票售罄或下架后，管理员清空机票信息，机票会进入退出流通状态，最后管理员删除机票。请根据上述描述画出相应状态转换图。



## ▼ 小结

### >>> 课程小结



#### 课程小结：

- 1、可行性研究任务的内容。
- 2、数据流图的绘制方式与数据字典的表达形式。
- 3、需求分析的定义和作用。
- 4、E-R图（**重点**）和状态转移图的内容及绘制方式。

### ▼ III. 软件设计

#### 总体设计

- 总体设计不考虑具体实现



#### » 3.1 总体设计



##### 总体设计：

总体设计又称为概要设计或初步设计。

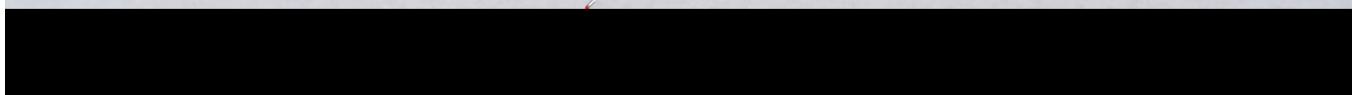
##### 任务：

- 确定系统中每个程序由哪些模块组成以及这些模块相互间的关系。
- 划分出物理元素。包括程序、文件、数据库、文档等。

设计过程包括系统设计阶段和结构设计阶段。

**例题 1：**在软件开发的总体设计阶段，哪项任务是最为关键的（）

- A. 编写详细代码
- B. 确定软件的架构和组件
- C. 进行单元测试
- D. 制定项目计划



#### 层次图和hipo图

- hipo图就是加上序号的层次图

## >>> 3.2 总体设计工具



层次图：

用方框和连线表示上下层的调用关系



HIPO图：

HIPO图的画法就是在层次图里除了最顶层的方框之外，每个方框都加了编号



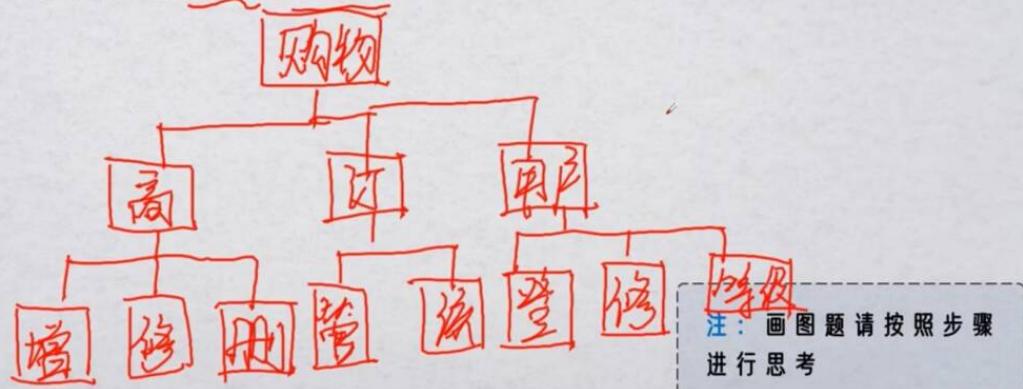
## >>> 3.2 总体设计工具



例题1：请实现一个简单购物商城的总体设计，其主要包含三块内容：

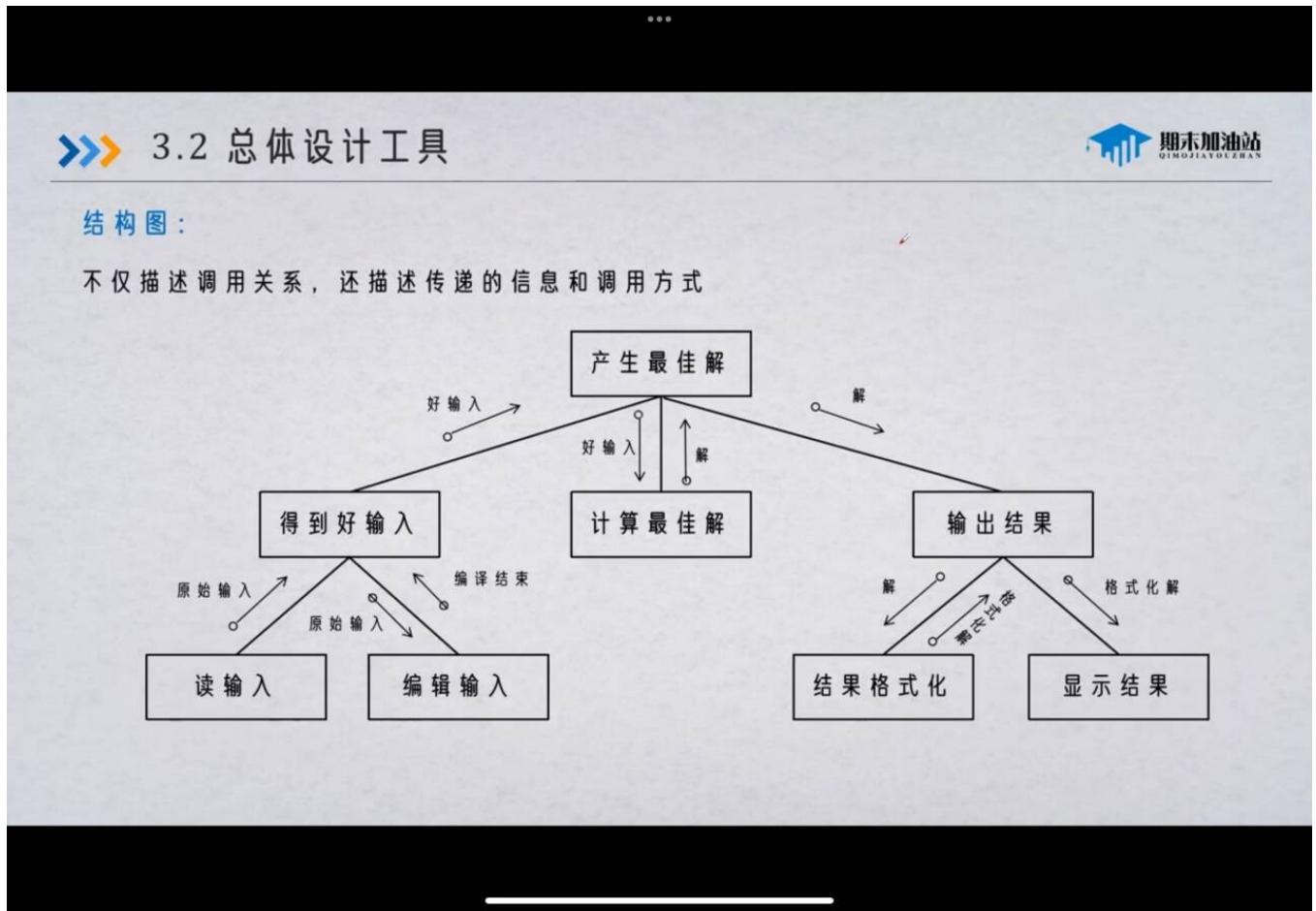
- ✓ 1.商品管理模块：可以增加、修改、删除商品信息
- ✓ 2.订单管理模块：管理员可以对订单进行管理和统计
- 3.用户管理模块：登录注册，修改用户，等级管理等。

请画出对应系统设计的层次图和HIPO图



结构图

- 从左到右进行数据流动



## 设计原理与理念

### >>> 3.3 设计原理与概念



耦合:

是对一个软件结构内 不同模块互连 程序的度量。

注：软件设计的目标就是高内聚，低耦合

无直接耦合 =》 数据耦合 =》 标记耦合 =》 控制耦合 =》 外部耦合 =》 公共耦合

内聚:

用来度量一个 模块内部各个元素 彼此结合的紧密程度。

偶然内聚 =》 逻辑内聚 =》 时间内聚 =》 过程内聚 =》 通信内聚 =》 顺序内聚 =》 功能内聚

软件深度: 软件结构中控制的层数;

软件宽度: 软件结构内同一个层次上的模块总数的最大值

模块的扇入: 一个模块直接调用其它模块的数目

模块的扇出: 一个模块被其它模块调用的数目

### >>> 3.3 设计原理与概念

模块  $\Rightarrow$  模块



耦合:

是对一个软件结构内 不同模块互连 程序的度量。

模块间

注：软件设计的目标就是高内聚，低耦合

无直接耦合 =》 数据耦合 =》 标记耦合 =》 控制耦合 =》 外部耦合 =》 公共耦合

内聚:

用来度量一个 模块内部各个元素 彼此结合的紧密程度。

模块内

偶然内聚 =》 逻辑内聚 =》 时间内聚 =》 过程内聚 =》 通信内聚 =》 顺序内聚 =》 功能内聚

软件深度: 软件结构中控制的层数;

3

软件宽度: 软件结构内同一个层次上的 模块总数 的 最大值

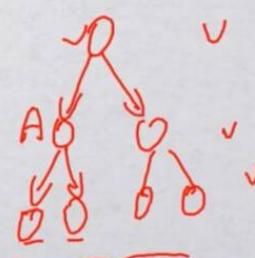
4

模块的扇入: 一个模块直接 调用 其它模块的数目

2

模块的扇出: 一个模块被其它模块 调用 的数目

1



**详细设计：**

详细设计确定每个模块的内部特征，即每个模块内部的执行过程。

模块的独立性可以用两个定性标准度量：耦合和内聚

**详细设计的任务：**

过程设计：即设计软件体系结构中所包含的每个模块的实现算法

数据设计：设计软件数据结构

接口设计：设计软件内部各模块之间的接口

**例题 1：**详细设计的主要任务是什么（ ）

- A. 确定软件的高层结构
- B. 编写伪代码和详细的算法描述
- C. 分析系统的可行性和需求
- D. 确定系统的安全需求

## 程序流程图

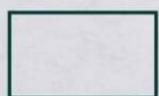
- 选择时要在连线处表明t还是f

## >>> 3.5 详细设计工具

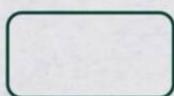


### 程序流程图：

又称为程序框图，它是历史最悠久使用最广泛的描述软件设计的方法



处理



源点 / 终点



选择

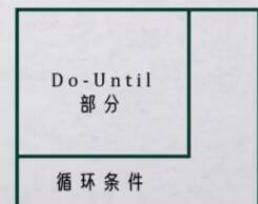
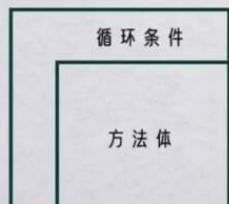
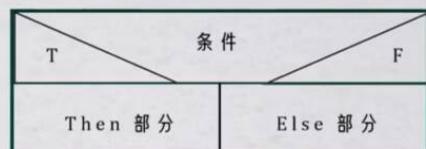
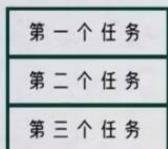
## 盒图

## >>> 3.5 详细设计工具



### 盒图 (N-S)：

仅由顺序、选择、循环三种基本结构组成。



## 判定表

- 条件打勾表示条件满足，底下的结果打勾表示满足打勾的条件则会有什么结果

» 3.5 详细设计工具

 期末加油站  
QIMOJIAYOUZHAN

**判定表：**  
当算法中包含多重嵌套的条件选择时判定表却能够清晰地表示复杂的条件组合与应做的动作之间的对应关系。

**组成：**  
左上部列出所有条件，左下部是所有可能的动作。右上部是表示各种条件组合，右下部是和每种条件组合相对应的动作。

条件 1	√	-	-
条件 2	√	-	√
条件 3	-	√	√
结果 1	√	-	√
结果 2	-	√	√

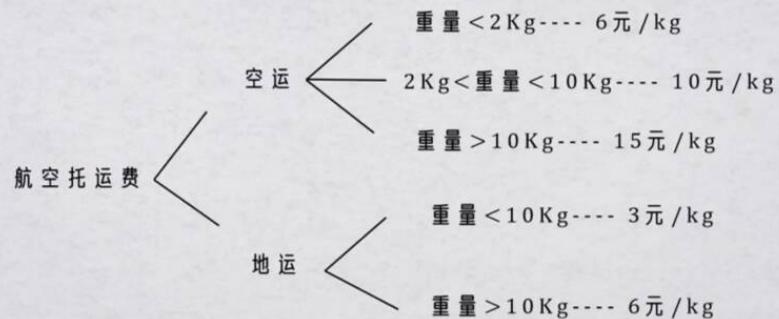
## 判定树

## >> 3.5 详细设计工具



判定树：

判定树是用一种树图形方式来表示多个条件、多个取值所应采取的动作



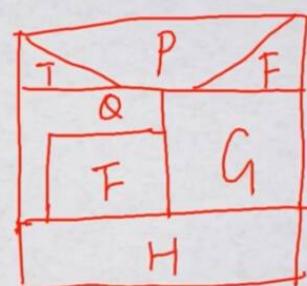
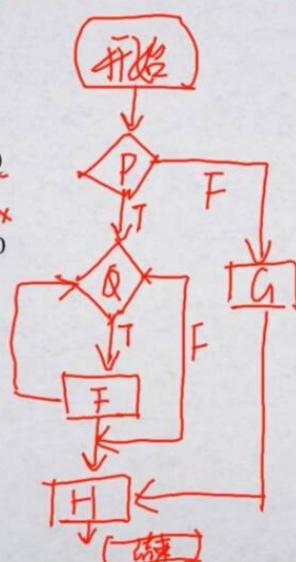
## 流程图和盒图例题

## >> 3.5 详细设计工具



例题 1：请根据下列伪代码，画出对应的程序流程图和盒图。

```
START  
IF P  
THEN  
  WHILE Q  
    Do F  
    END DO  
ELSE  
  END IF  
  H  
STOP
```



# 程序复杂度

## 3.6 程序复杂度定量度量



### 程序复杂度定量度量：

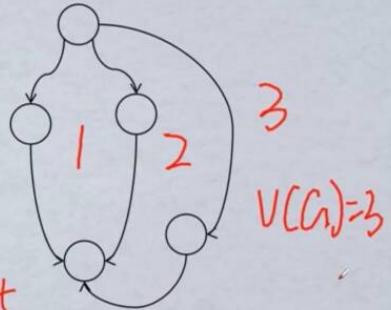
定量的度量详细设计模块的质量。

### 计算方法：

- 1、流图中的区域数等于环形复杂度 ✓
- 2、流图 G 的环形复杂度  $V(G) = E - N + 2$ , E是流图中边的条数, N是结点数。 ✓

注：将图转化为程序流图计算即可

例题 1：请计算以下图形对应程序的复杂度



$$E=6, N=5$$

$$V(G)=6-5+2=3$$

## ▼ 小结



## ">>>> 课程小结



### 课程小结：

- 1、总体设计以及总体设计的工具。
- 2、软件设计的原理概念和目标。
- 3、详细设计及详细设计工具。
- 4、程序复杂度的定量度量。

< >

## ▼ IV. 软件测试

### 测试与实现

## >>> 4.1 实现与测试



### 实现：

软件生命周期的**编码**和**测试**统称为实现；

**测试的目标**：为了发现错误而执行程序的过程。

- 编码阶段（单元测试）
- 测试阶段（各种综合测试）

### 软件测试的方法：

#### 黑盒测试法：

不考虑其内部结构和处理过程，测试软件是否能够正确接收输入数据，产生正确的输出数据。即测试程序是否正确的实现了其功能。又称为功能测试。

#### 白盒测试法：

完全知道程序的内部结构和处理算法，根据程序内部的逻辑结构测试程序内部的主要执行通路是否能够按照预定的要求正确工作。又称结构测试

**例题1：**软件测试方法中，黑盒测试方法和白盒测试方法是常用的方法，其中黑盒测试方法主要用于测试（ ）

- A. 结构合理性
- B. 软件外部功能
- C. 程序正确性
- D. 程序内部逻辑

**例题2：**软件测试按照功能划分可以分为：

- A. 黑盒测试和单元测试
- B. 白盒测试和黑盒测试
- C. 集成测试和单元测试
- D. 白盒测试和静态测试

## 软件测试步骤

- 单元测试在编码阶段
- 子系统测试：将每一个模块集成成子系统。系统测试：将每一个子系统集成成系统
- 依照文档：单元测试依照详细设计文档。确认测试依照需求规格文档。

## >>> 4.2 软件测试的步骤



软件  
测  
试  
步  
骤

### 1. 单元测试（模块测试）：

将每个模块作为一个单独的实体进行测试。  
发现的错误编码和详细设计阶段的错误。

测试依据：详细设计文档

测试方法：白盒测试

模块接口

局部数据结构

重要的执行通路

出错处理通路

边界条件

着重点

### 2. 集成测试：

是子系统测试和系统测试的总称。  
以发现与接口有关的问题为目标。

非渐增测式

渐增测式

自顶向下

自底向上

3. 验收测试（确认测试）：  
验证：为了保证软件正确实现了特定要求而进行的活动。  
目标是验证软件的有效性。

确认：为了保证软件确实满足了用户需求而进行的活动。

4. 平行运行：同时运行新开发出来的系统和将被它取代的旧系统，以便比较新旧两个系统的处理结果

## 白盒测试

## >>> 4.3 白盒测试



测试用例：测试输入数据和预期的输出结果。

测试集合：测试目的、测试用例的集合。

逻辑覆盖测试：

- 语句覆盖：每一条语句都至少执行一次（最弱的标准）
- 判定覆盖：每个判定表达式至少获得一次 true 和 false
- 条件覆盖：每个条件的各种可能的值至少出现一次
- 条件组合覆盖：使得每个判定表达式中条件的各种可能的值的组合都至少出现一次。
- 路径覆盖：覆盖被测程序中所有可能的路径

控制结构测试：

- 基本路径测试
- 条件测试
- 循环测试

注：白盒测试是知晓内部具体算法实现的。

# 黑盒测试

## 2.4 黑盒测试



黑盒测试又称功能测试，着重关注软件的外部行为。

### 等价划分法：

- 把程序的输入集合按输入条件划分为若干个等价类，每一个等价类表示为有效或无效的输入。
- 为每一等价类设计一个测试用例

### 边界值分析法：

- 输入等价类和输出等价类的边界应该着重测试的程序边界情况
- 选取的测试数据应该刚好等于、刚好小于、刚好大于边界值

**例题1：**黑盒测试在设计测试用例时，主要研究（ ）

- A. 需求规格说明与概要设计说明      B. 详细设计说明  
C. 项目开发计划      D. 概要设计说明与详细设计说明

**例题2：**边界值分析通常与哪种测试技术结合使用？

- A. 单元测试 B. 压力测试 C. 等价类划分 D. 回归测试

# 调试

- 测试为了发现错误，调试为了解决错误

## >>> 2.5 调试与可靠性



### 调试：

调试（也称为纠错），是在测试发现错误之后排除错误的过程。

### 结果：

1. 找到了原因，然后改正和排除；
2. 没找到原因，猜测一个原因，进行测试来验证假设。

**软件可靠性：**在给定时间间隔内，按照规格说明书的规定成功运行的概率。

**软件可用性：**在给定时间点，按照规格说明书的规定成功运行的概率。

**例题1：**软件可靠性指的是（）

- A. 软件在特定条件下，给定的时间内无故障运行的概率      B. 软件的性能优化程度  
C. 软件的用户体验满意度      D. 软件的功能完整性

**例题2：**说明软件测试和调试的目的有何区别？

## ▼ 小结

## >>> 课程小结



### 课程小结：

- 1、软件实现与测试的基本概念。
- 2、软件测试的重要步骤及内容。
- 3、白盒测试的特点与方法。
- 4、黑盒测试的特点与方法。
- 5、调试的内容及软件的可靠性、可用性。

## ▼ V. 面向对象设计

### 面向对象的概念

#### >>> 5.1 面向对象的概念



**面向对象：**

按人类习惯的思维方法，以现实世界中客观存在的对象为中心来思考和认识问题。

**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，包括具体的物理实体的抽象、人为的概念、任何有明确边界和意义的东西。

**对象的两个基本要素：**

**属性**：用来描述对象静态特征的一个数据项；

**服务**：用来描述对象动态特征（行为）的一个操作序列。

面相对象的特征（特点）：封装性，继承性，多态性。

**例题 1：**面向对象的主要特征除对象封装、继承外，还有（ ）

- A、多态性
- B、完整性
- C、可移植性
- D、兼容

### 类

## >>> 5.1 面向对象的概念



类：

具有相同属性和服务的一组对象的集合，它为属于该类的全部对象提供了统一的抽象描述。

父类与子类

父类：一个类的上层是父类。

子类：一个类的下层是子类。

单继承与多继承

单继承：一个类只允许有一个父类，即类等级为树形结构。

多继承：一个类允许有多个父类

例题 2：在面向对象的编程中，类和对象之间的关系是什么（ ）

- A. 类是对象的具体实例
- B. 对象是类的抽象模板
- C. 类和对象之间没有直接关系
- D. 类是对象的抽象模板，对象是类的实例

## 模型

## >>> 5.2 对象模型-类图



模型：

是一组图示符号和组织这些符号的规则，利用它们来定义和描述问题域中的术语和概念。

- 对象模型：定义实体，描述系统数据，定义“对谁做”
- 动态模型：描述系统控制结构，规定“何时做”
- 功能模型：描述系统功能，指明系统应“做什么”

类图，是常见的对象模型

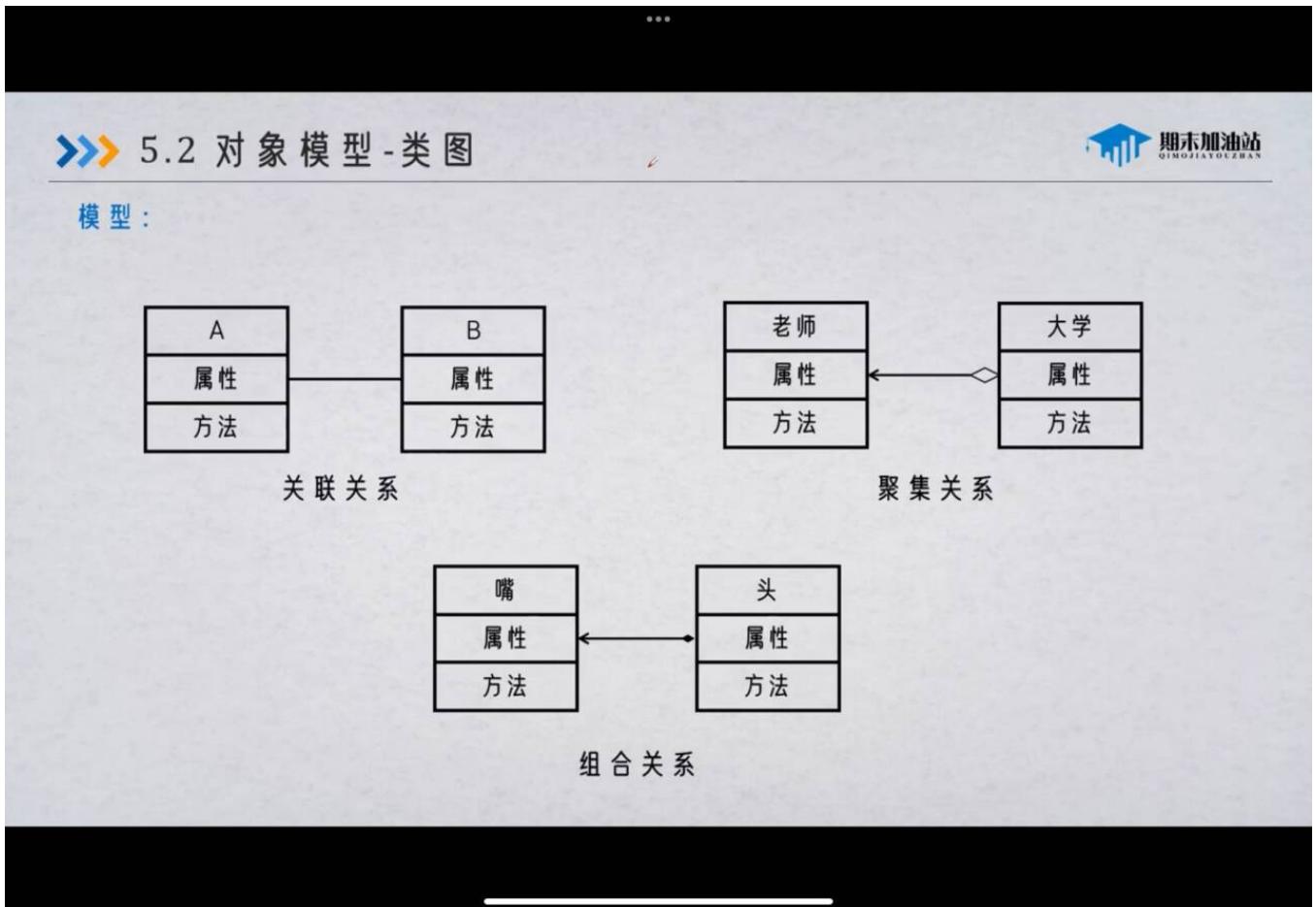
类和类之间的联系称为“关系”。

通常分为：**关联、聚合、组合、依赖和泛化**。

注：聚合强调的是“has a”的关系，  
而组合强调的是“contains a”的关系

## 类图的画法

- 关联关系中如果有1对多或者多对多的情况需要在连线起点处标注此类是1还是n
- 聚集关系中被聚集的类用空心菱形表示，聚集的类用箭头表示，并且聚集的类和被聚集类并非强绑定，没有被聚集的类也可以有聚集的类
- 组合关系中被组合的类用实心菱形表示，组合的类用箭头表示。并且被组合类和组合类强绑定，没有被组合类就不会有组合类

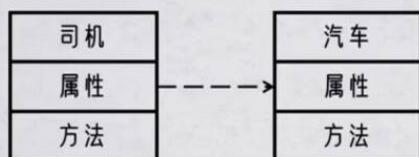


- 依赖关系表示一个类需要调用另一个类的方法。被依赖的对象用箭头指向
- 继承（泛化）中，被继承的类用空心三角形指向

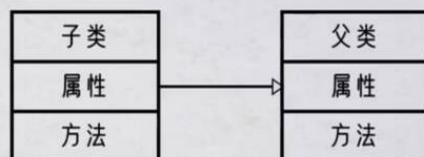
## >>> 5.2 对象模型 - 类图



模型：



依赖关系

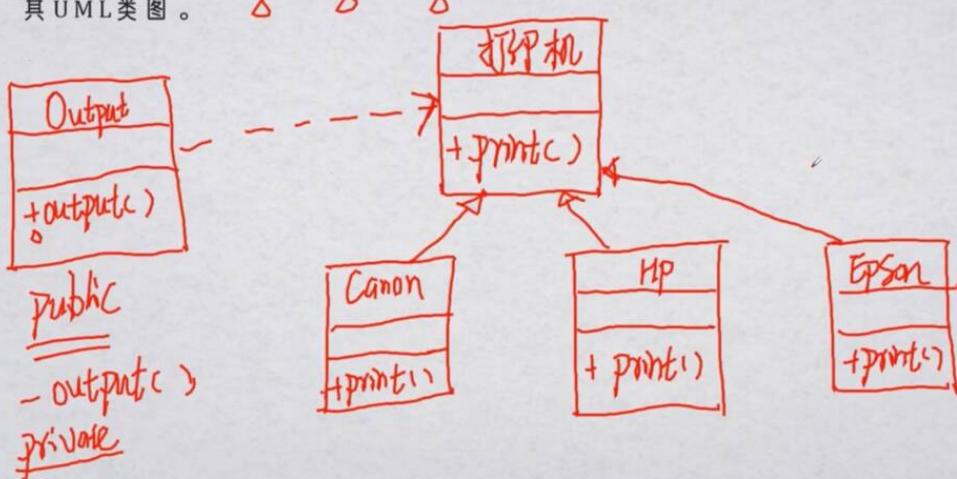


泛化（继承）关系

## >>> 5.2 对象模型



例题 1：某输出类 Output 中包含 output 方法，该方法可以使用于多种不同品牌、不同型号的打印机，如：Canon、HP、EPSON，每种打印机都有不同的 print 方法。请设计出其 UML 类图。



用例图

## >>> 5.3 功能模型 - 用例图



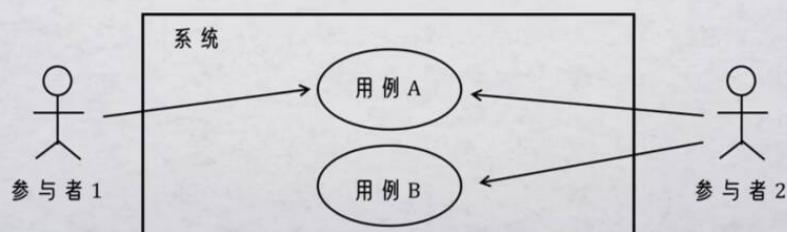
### 动态模型：

用来描述系统与时间相关的动态行为，从对象的“事件”和“状态”的角度出发，表现对象经过相互作用后，随时间改变的不同运算顺序。

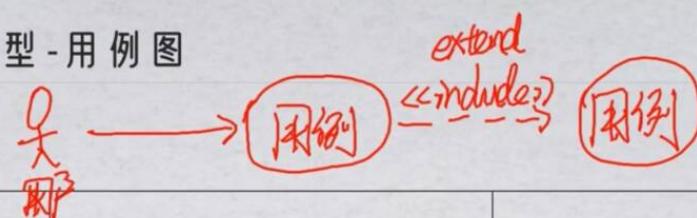
动态模型包括“状态图”和“事件追踪图”。

### 功能模型则一般使用数据流图和用例图

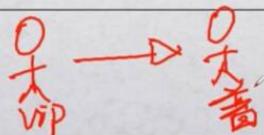
用例图是用户与系统交互的最简表示形式，展现了用户和与其相关的用例之间的关联关系。



## >>> 5.3 功能模型 - 用例图



关系类型	描述	描述符号
关联	参与者与用例之间的关系	→
包含	用例之间的关系	→<<include>>
扩展	用例之间的关系	→<<extend>>
泛化	参与者或用例间的关系	→



## 三个模型之间的关系

## >>> 5.4 模型间的关系



### 三种模型间的关系：

三个模型从三个侧面描述了开发的系统，它们相互补充、互相配合。

功能模型指明系统必须做什么；动态模型规定什么时候做；对象模型定义了做事情的实体。

**例题 1：**在功能模型中，用例图的主要目的是什么？

- A. 描述系统的状态变化
- B. 表示系统与外部实体的交互
- C. 定义系统的内部结构
- D. 展示系统的数据处理过程

**例题 2：**面向对象的分析方法主要是建立三类模型，即（ ）

- A. 系统模型、ER模型、应用模型
- B. 对象模型、动态模型、应用模型
- C. E-R模型、对象模型、功能模型
- D. 对象模型、动态模型、功能模型

## ▼ 小结

## >>> 课程小结

### 课程小结：

- 1、面相对象的特点与概念。
- 2、类图的特点及绘制方式。
- 3、用例图的特点及绘制方式。
- 4、功能模型，动态模型，对象模型三者的概念及关系。

## ▼ VI. 软件维护

### 软件维护的概念

- 预防性维护在出问题之前进行
- 非结构化维护：只能从代码维护
- 结构化维护：有完整的软件配置



#### >>> 6.1 软件维护的概念



##### 定义：

是在软件已经交付使用之后，为了改正错误或满足新的需要而修改软件的过程

##### 分类：

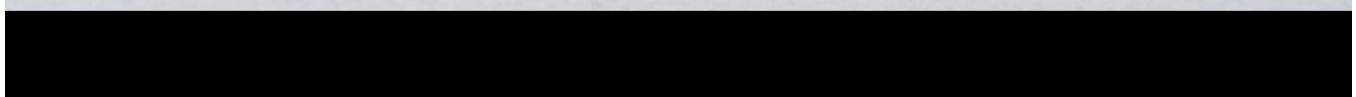
- 改正性维护：诊断和改正错误
- 适应性维护：适应软硬件环境
- 完善性维护：增加新功能、性能优化
- 预防性维护：提高可维护性与可靠性

##### 特点：

- 结构化维护与非结构化维护差别巨大
  - 非结构化维护：唯一成分是程序代码，那么维护活动从艰苦地评价程序代码开始。
  - 结构化维护：有完整的软件配置存在，那么维护工作从评价设计文档开始。
- 维护的代价高昂
- 维护的问题很多

##### 例题 1：软件维护的主要任务是什么（ ）

- A. 提高软件性能
- B. 修复软件中的缺陷
- C. 适应环境变化
- D. 满足新的需求



### 可维护性

## >>> 6.2 软件的可维护性



可维护性：

软件被理解、改正、优化的难易程度

决定因素：

- 可理解性：理解软件的难易程度。
- 可测试性：论证程序正确性的容易程度。
- 可修改性：程序容易修改的程度。
- 可移植性：转移到另一计算环境的难易程度。
- 可重用性：同一个软件不做修改或稍加改动，就可以在不同环境中多次重复使用。

注：预防性维护也称为软件再工程，是为了提高未来的可维护性或可靠性，而主动地修改软件

例题1：在软件可维护性包括（ ）。

- A. 软件正确性、灵活性、可移植性
- B. 软件可测试性、可理解性、可修改性
- C. 软件可靠性、可复用性、可使用性
- D. 软件灵活性、可靠性、高效性

## 规模估算

## >>> 6.3 规模估算与甘特图



软件项目管理：

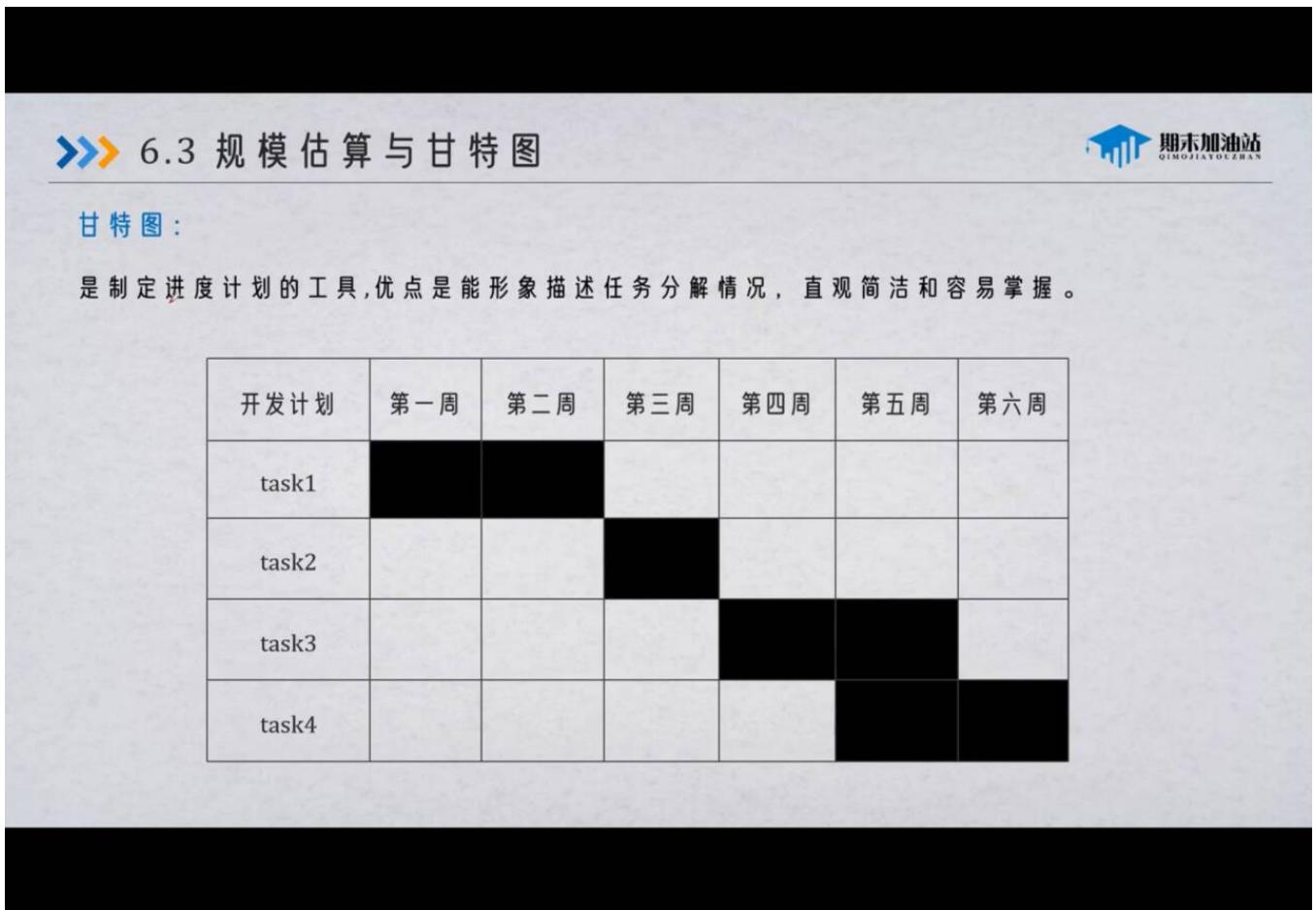
通过计划、组织和控制等一系列活动，合理地配置和使用各种资源，以达到既定目标的过程。

估算软件规模方法：

1. 代码行技术：根据以往开发经验和历史数据，估算实现一个功能所需源代码行数。
  - 优点：代码是所有项目都有的“产品”，容易计算代码行数。
  - 缺点：源代码为软件配置的一个部分，用来衡量整个软件规模不太合理。不同语言实现同一软件所需代码行数不相同
2. 功能点技术：以**功能点(FP)**为单位度量软件规模

## 甘特图

- 横轴代表时间，纵轴代表任务
- 被涂黑表示任务代表在哪一周完成



## 保障与配置管理

## >>> 6.4 保障与配置管理



### 软件质量：

软件质量就是“软件与明确地和隐含地定义的需求相一致的程度”

### 软件质量保障措施主要有：

- 基于非执行测试（复审或评审）；
- 基于执行测试（软件测试）；
- 程序正确性证明（数学方法）

软件配置管理：是在软件生命期内管理变化的一组活动，用来标识、控制、报告变化，确保适当的实现了变化。

软件配置管理5项任务：变更标识、版本控制、变更控制、配置审计和状态报告。

能力成熟度模型(CMM)：是用于评价软件机构的软件过程能力成熟度模型，用于帮助软件开发机构建立一个有规模的，成熟的软件过程。

五个等级从低到高：初始级、可重复级、已定义级、已管理级、优化级。

## ▼ 小结

## >>> 课程小结



### 课程小结：

- 1、软件维护的相关概念。
- 2、软件可维护性的定义及特点。
- 3、规模估算及甘特图的特点。
- 4、质量保障，软件配置管理及能力成熟度模型。

