

中级软件设计师下午试题模拟 36

试题一

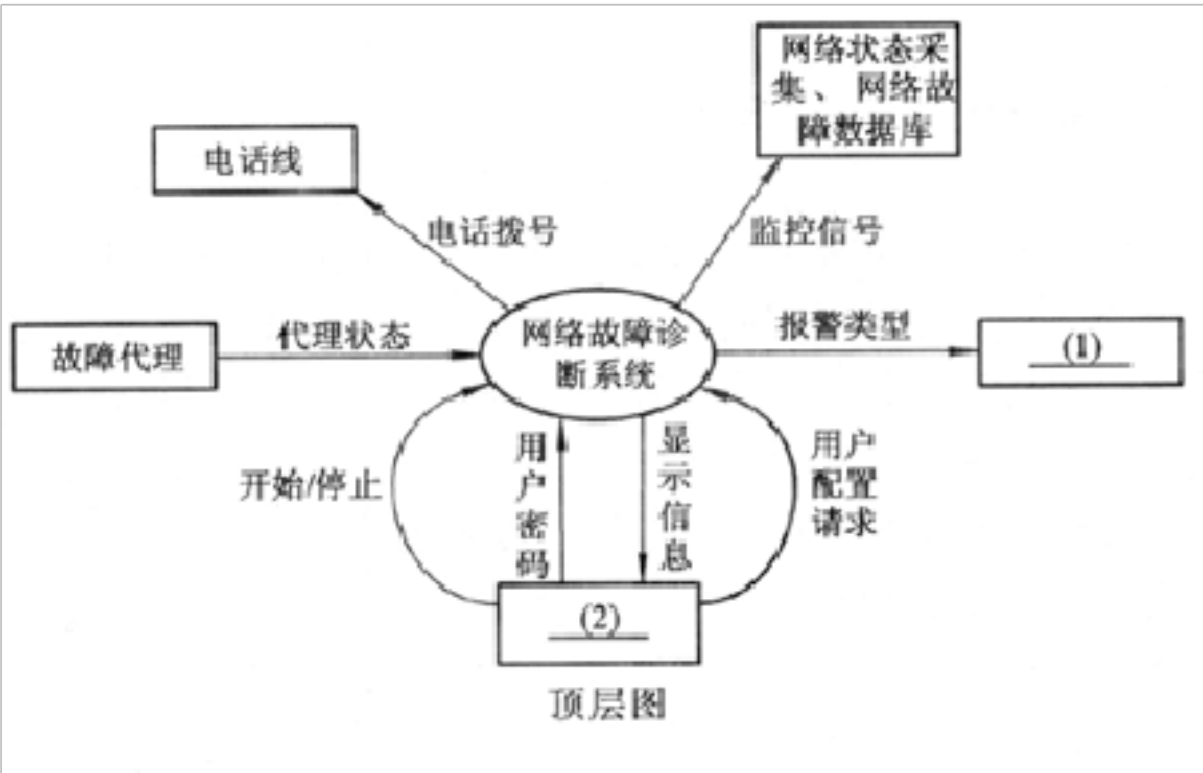
阅读下列说明和数据流图，回答问题。

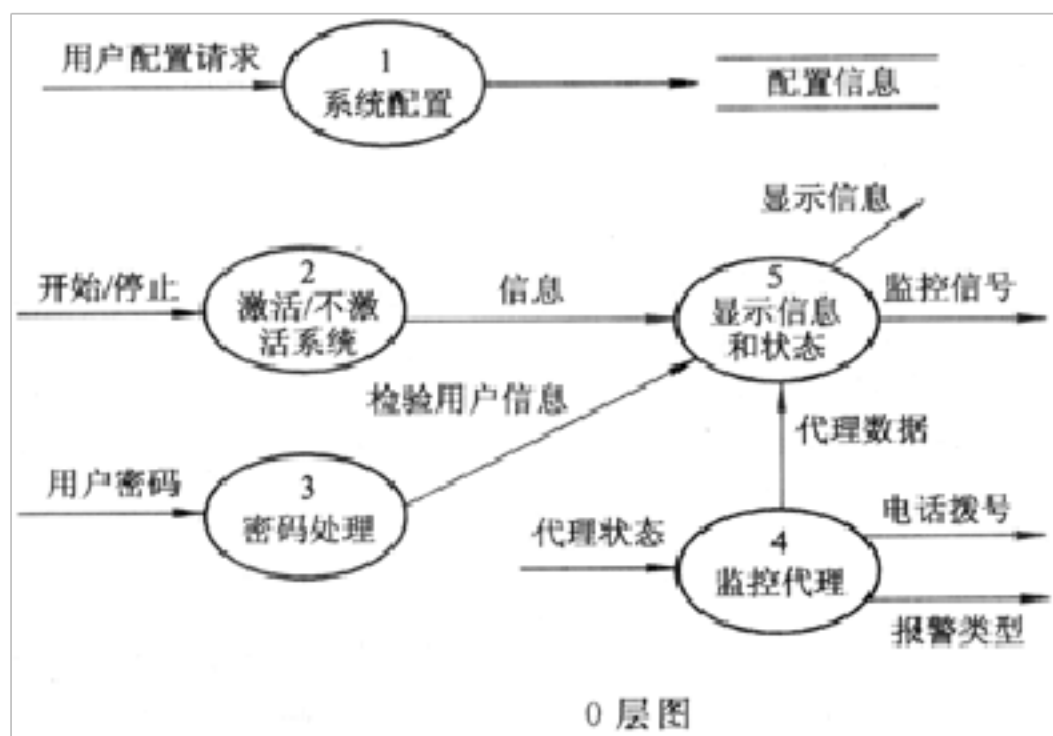
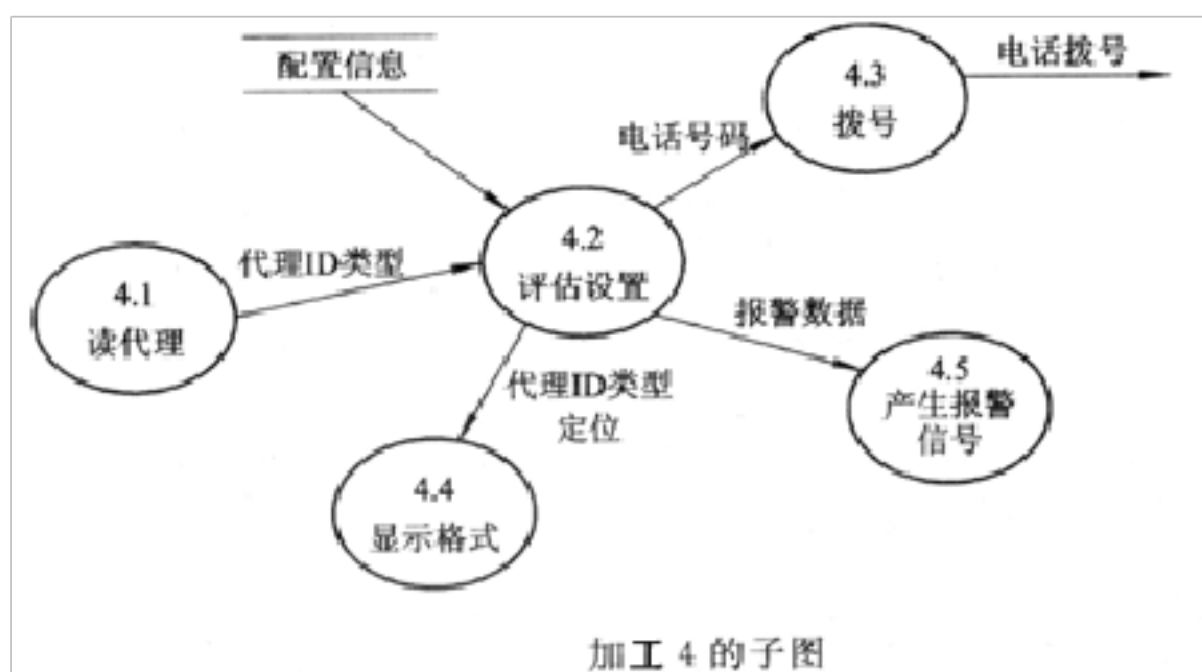
1. 【说明】

某网络故障诊断系统使用故障代理(agent、SNMP Trap等)来检测各种意外情况，如大幅丢包、路由冲突、广播风暴等。网络管理员可以在安装该系统时配置安全监控程序(如故障代理程序、实时诊断程序、报警器等)，也可以在系统运行时修改配置，通过网络状态采集器和故障特征数据库，并通过控制面板上的键盘与系统进行信息交互。

在安装过程中，系统给每个故障代理赋予一个编号(即 ID)和类型，并设置管理员密码以启动和关闭系统，设置故障代理事件发生时应自动拨出的电话号码。当系统检测到一个故障代理事件时，就激活警报，拨出预置的电话号码，并报告位置和检测到的事件的性质等信息。

该网络故障诊断系统的顶层图如图 13-16 所示，0 层图如图 13-17 所示，加工 4 的子图如图 13-18 所示。





【问题 1】

将顶层图中的(1)和(2)空填充完整。

【问题 2】

0 层图中的数据文件“配置信息”是多余的吗?若是,请说明理由;若不是,请指出它会影响。层图中的哪些(哪个)加工(除加工“系统配置”之外)?

【问题 3】

指出图 13-18 所示的加工 4 的子图中遗漏的数据流。

注意:书写格式为“缺少从×到×的数据流××”或“~~缺少~~输入(出)数据流××”
若未按格式书写,将被扣分。

答案: [问题 1]

(1)报警器 (2)控制面板

[问题 2]

数据文件“配置信息”是必需的。

会影响加工“3密码处理”和加工“4监控代理”。

[问题 3]

“4 5 产生报警信号”缺少输出数据流“报警类型”。

“4 4 读代理”缺少输入数据流“代理状态”。

“4 1 显示格式”缺少输出数据流“代理数据”。

问题 1 比较简单。题目说明中提到“网络管理员可以在安装该系统时配置安全监控程序(如故障代理程序、实时诊断程序、报警器等)”，其中提到的“报警器”在顶层图中并未出现，又因为有数据流“报警类型”流向(1)所表示的实体，所以，第(1)空应填“报警器”。题目说明中提到“并通过控制面板上的键盘与系统进行信息交互”，又从(2)所代表的实体输出数据流“用户密码”、输入数据流“显示信息”易知，(2)表示实体“控制面板”。做这种题的关键就是抓住题目说明中提供的信息。

问题 2 稍有难度。题目给出的 0 层图显然是不完整的，因为在加工“4监控代理”的子图中出现了文件“配置信息”，而 0 层图中加工“4监控代理”跟文件“配置信息”没有任何联系，这显然表明 0 层图遗漏了数据流。可见，至少有“1系统配置”和“4监控代理”两个加工用到文件“配置信息”，所以它不可能是某个加工的局部文件，因此它是必需的。

在 0 层图中，加工“3密码处理”用于检验密码，要验证用户密码就得将用户输入的密码跟保存在某文件中的管理员密码进行对比验证。注意到，用户配置请求经过“1系统配置”加工之后写入到文件“配置信息”，可见，用户的相关信息也保存到了该文件中，该文件记录了配置请求及相应的请求人。所以“3密码处理”一定要用到“配置信息”文件。在验证密码时，若密码正确，则将管理员信息、验证成功的信息发送给加工“5显示信息和状态”以显示相关信息；若密码错误，则将验证失败的信息发送给加工“5显示信息和状态”提示密码错误，并要求重新输入。

问题 3 是考察数据流图的相关注意事项。关于数据流图，必须注意如下几条原则：

(1)数据流图上的所有符号只限于加工、数据流、数据存储和外部实体这四种

基本元素的符号，不得出现其它符号。

(2)顶层图必须包括前述四种基本元素，缺一不可，而且外部实体之间不能有数据流（根据上述外部实体的定义可知）。顶层图中只能有一个加工（即被开发系统），不能有其它加工。

(3)每个加工都必须至少有一个输入数据流和一个输出数据流。一个加工的输出数据流不能与其输入数据流同名，即便它们的组成相同。而且一个加工的输出数据流中的数据必须能从输入数据流中直接获得或从该加工能产生的数据中获得。

(4)父图和子图必须保持平衡。

(5)图中每个元素必须有名字，但加工和数据存储之间的数据流可有名也可无名。

(6)数据流图中不能夹带控制流，因为数据流图是实际业务流程的客观反映，只说明系统“做什么”而不表明系统应“如何做”，因此不是系统的执行序列或程序流程图。

(7)在自顶向下的分解过程中，若一个数据存储首次出现时只与一个加工有关，那么这个数据存储应作为该加工的内部文件（局部文件）而不必画出。

(8)在整套数据流图中，每个数据存储必须既有读的数据流又有写的数据流，但在某一张子图中，可以只有写数据流或者读数据流。在整套数据流图中，每条数据流必须既有出发地又有目的地。

(9)任何一条数据流必须与加工相关，不允许数据流两端连接的都不是加工，即要求其至少有一端是加工。

解答问题3的关键就是父图、子图数据平衡原则。子图是其父图中某一部分内部的细节图。它们的输入、输出数据流应该保持一致。在上一级图中有几个数据流，它的子图也一定有同样的数据流，而且它们的方向也是一致的。

在0层图中，加工“4监控代理”有1条输入数据流“代理状态”和3条输出数据流“电话拨号”、“代理数据”和“报警类型”，而在加工4的细化图中，只有输出数据流“电话拨号”，因此可以肯定缺少了“代理状态”、“报警类型”、“代理数据”这3条数据流。加工4的结构非常清晰，只需把这3条数据流对号入座即可——“报警类型”应是“4.5产生报警信号”的输出数据流；“代理状态”应是“4.1读代理”

的输入数据流；“代理数据”应该是经“4 4 显示格式”的输出数据流。

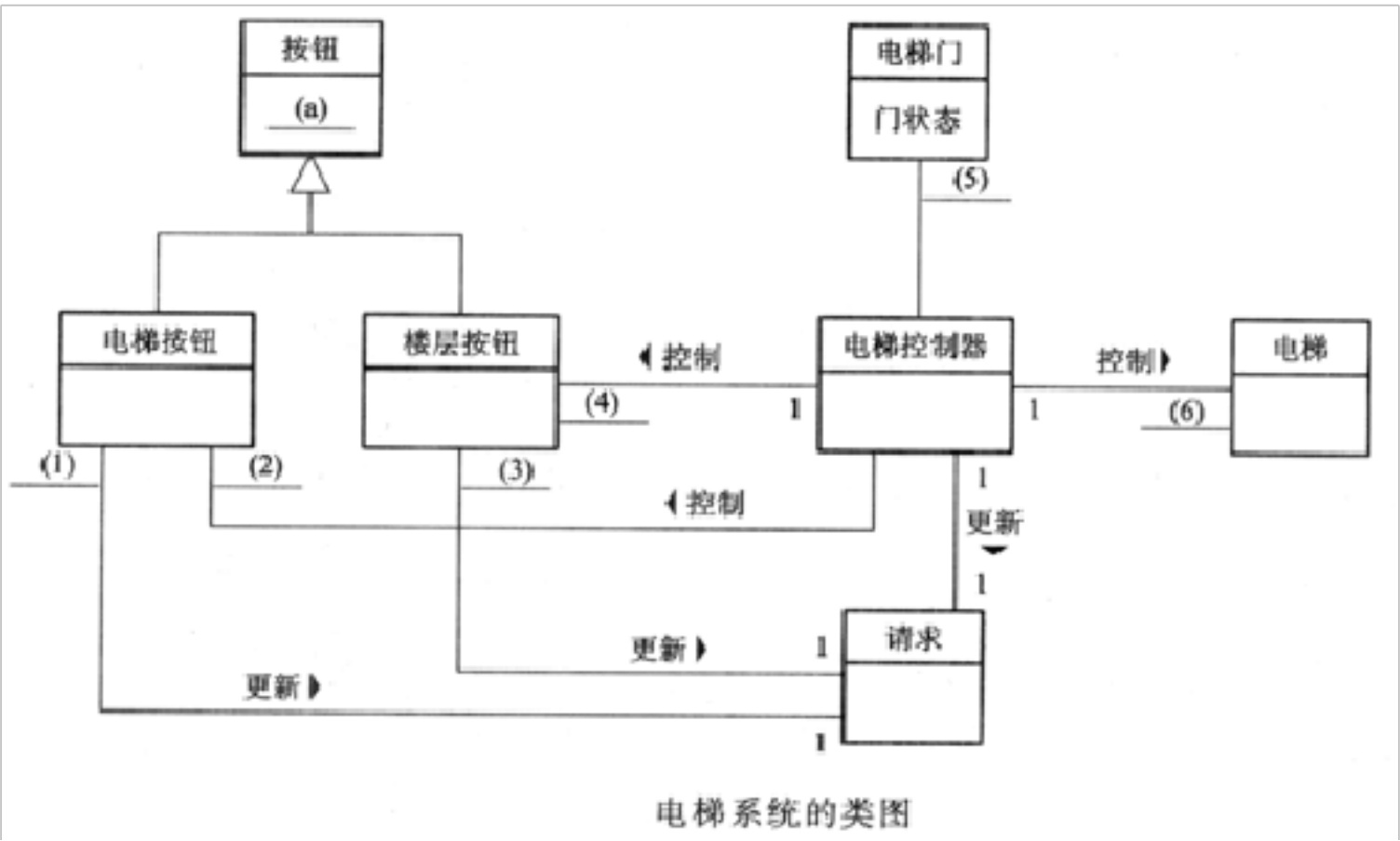
试题二

阅读下列说明和相关的类图，回答问题。

1. 【说明】

在一栋 m 层楼的大厦里,用电梯内和每个楼层的按钮来控制 n 部电梯的运作。当按下电梯内按钮请求电梯在指定楼层停下时,按钮指示灯亮,当电梯到达指定楼层时,指示灯熄灭。除了大厦的最底层和最高层之外,每层楼都有两个按钮分别指示电梯上行和下行,当这两个按钮之一被按下时相应的指示灯亮,当电梯到达此楼层时灯熄灭,电梯向要求的方向移动。当电梯无升降运动时,关门并停止在当前楼层。

软件设计师火云龙采用面向对象分析的方法,经过三次逐步求精设计之后,得到如图 13-19 所示的类图。



【问题 1】

按钮类有一个重要的属性, 请将该属性填入图 13-19 中的(a)处。

【问题 2】

识别关联的多重性是面向对象建模重要的一步，请根据题目说明填空(1)~(6)。

【问题 3】

软件设计师火云龙在初步设计类时，将门(的状态)作为电梯的属性，后经思考，将电梯门设计成一个独立的类，这么做的好处是什么？

软件设计师火云龙在初步设计类时，并没有设计“请求”类，后来又加入了该类，这么做的原因是什么？

答案：[问题 1]

(a)指示灯

[问题 2]

(1)mn (2)mn (3) $2m-2$

(4) $2m-2$ (5)n (6)n

[问题 3]

一旦将电梯门作为一个独立的类，则打开或关闭电梯门的唯一办法就是向“电梯门”类对象发送消息。如果电梯门类封装得好，就能保证不会在错误的时间关闭或打开电梯门，从而有效杜绝严重意外事故的发生。

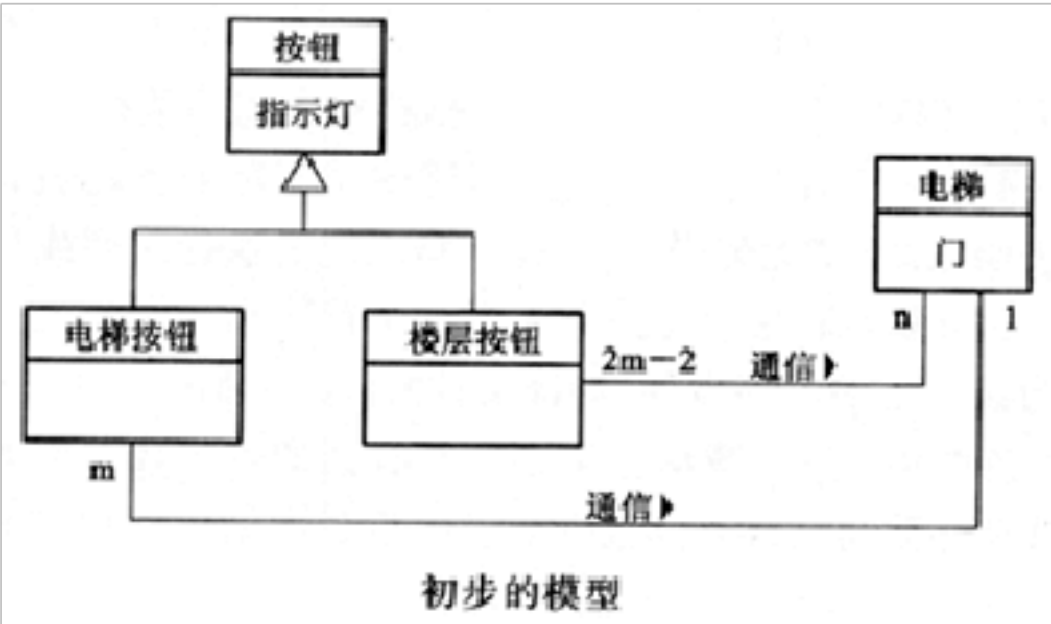
出于安全考虑，在未经授权时不能随意修改“请求”，因此将“请求”设计成独立的类，如果“请求”类封装得好就能有效保证电梯安全运行。

从图 13-19 可知，电梯按钮类和楼层按钮类都是按钮类的子类，电梯按钮和楼层按钮的共同点就是用指示灯跟乘客进行交互，所以，按钮类必定有个重要的属性“指示灯”。识别关联的多重性是面向对象建模重要的一步。题目的说明中提到“在一栋 m 层楼的大厦里，用电梯内和每个楼层的按钮来控制 n 部电梯的运作”、“除了大厦的最底层和最高层之外，每层楼都有两个按钮分别指示电梯上行和下行”，所以这栋大厦中一共有 mn 个电梯按钮，共有 $2m-2$ 个楼层按钮，电梯控制器类控制着所有的电梯按钮、楼层按钮和电梯，所以第(1)~(6)空应分别填： mn 、 mn 、 $2m-2$ 、 $2m-2$ 、 n 、 n 。注意，不可以用*号代替，*号表示多个，具体多少是未知，而题目已经给定具体数目，所以应该用 m 、 n 来表示。

识别类是面向对象分析的第一步，常用的方法是名词分析法。题目的说明其实就是电梯系统的需求说明，出现了如下主要名词：大厦、电梯、楼层、按钮、运

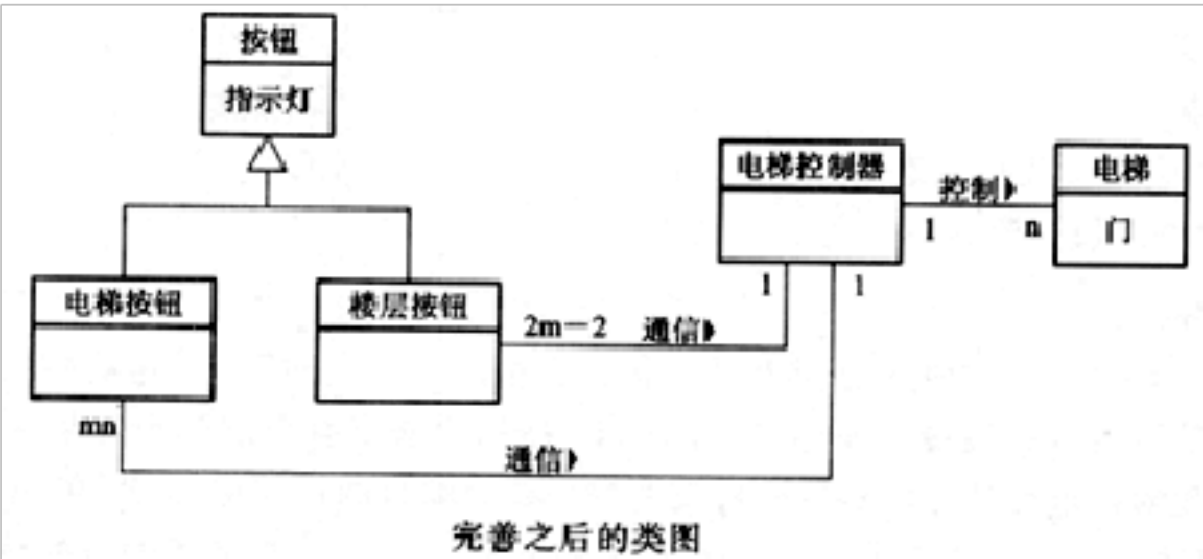
动、指示灯、请求、最底层、最高层、方向和门。这些名词所代表的事物可作为类的初步候选者。其中，大厦、最底层、最高层和楼层处于问题范畴之外，因此不必考虑。运动、方向、指示灯和门可作为其它类的属性，例如，指示灯(的状态)可作为按钮类的属性，方向和门(的状态)可作为电梯类的属性。

经过上述初步筛选只剩下两个基本的候选类，即电梯类和按钮类。在题目的说明中实际指定了两种按钮，因此，应该为按钮类定义两个子类，即电梯按钮和楼层按钮。综合上述分析结果，可得出系统的类图，如图 13-50 所示。



这个模型是非常初步的模型，需要进一步充实、精化和完善。

分析上述模型会发现它存在比较明显的不足：在实际的电梯系统中，按钮并不直接与电梯通信；为了决定分派哪一部电梯去响应一个特定的请求，必须有某种类型的电梯控制器。然而在题目说明中并未提到控制器，因此它未被列入候选类中。由此可见，名词分析法只为寻找候选类提供了初步线索，不能指望依靠这种方法找出全部候选类。系统分析员必须根据领域知识、常识和经验做进一步分析，才能找出问题域中所有的类。补充了电梯控制器类之后，其类图如图 13-51 所示。



这个模型也并非十分完善。电梯类的主要功能应是执行电梯控制器发出的移动或等待命令，如果将门(的状态)作为电梯的一个属性，则电梯还要执行关门或开门的命令，这样电梯类的功能就不单一了。于是将电梯门划分出来成为一个独立的类，一旦将其作为一个独立的类，则打开或关闭电梯门的唯一办法就是向“电梯门”类对象发送消息。如果电梯门类封装得好，就能保证不会在错误的时间关闭或打开电梯门，从而有效杜绝严重意外事故的发生。同样，出于安全考虑，在未经授权时不能修改“请求”，因此将“请求”设计成独立的类。经过再次完善之后的类图如图 13-19 所示。

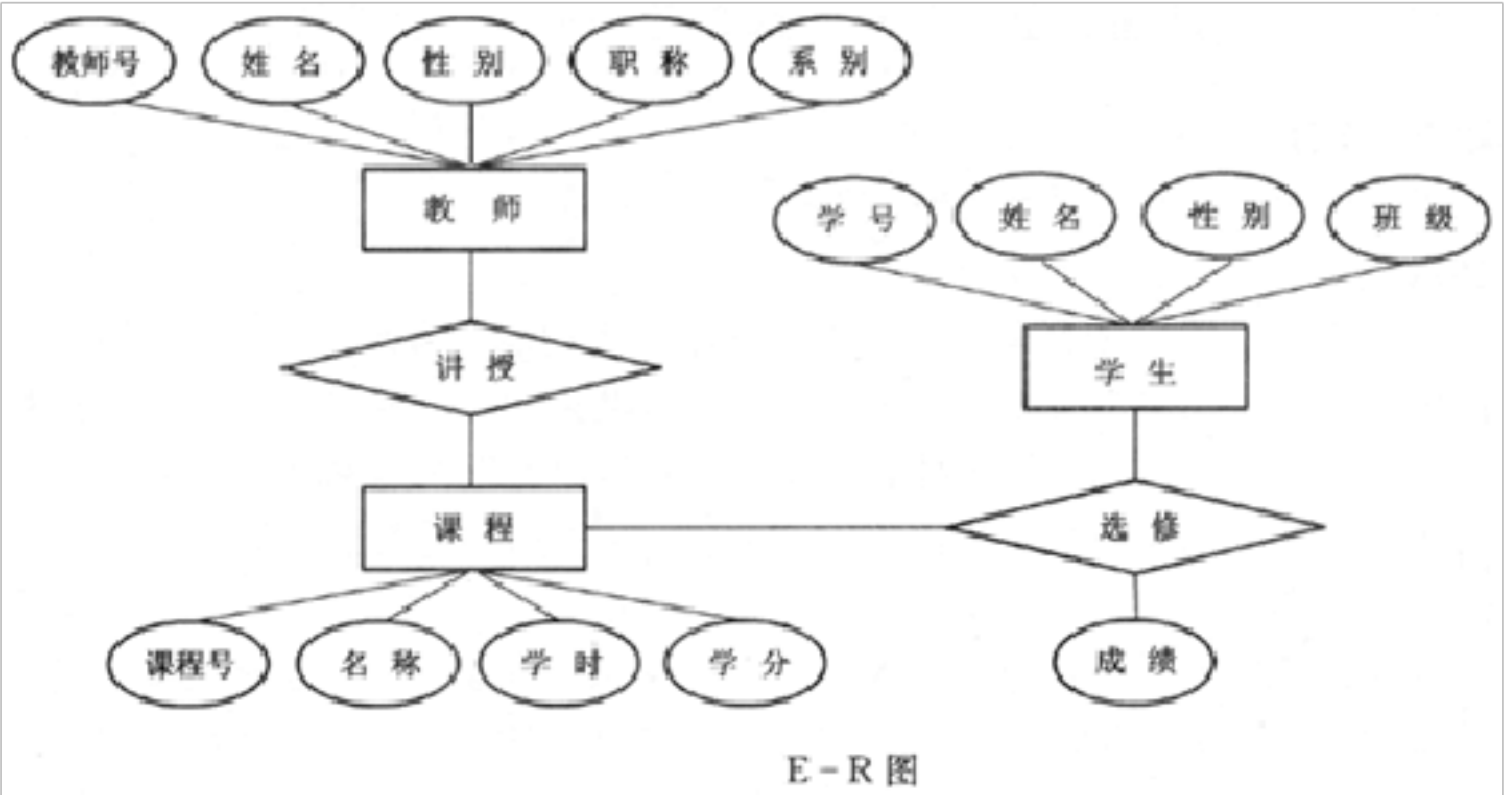
试题三

阅读以下说明和 E-R 图，回答问题。

1. 【说明】

某高校要设计一个教学管理数据库系统。通过调查，设计者了解到学生每学期按照事先安排的课程计划开始学习。每门课程由一名教师讲授；一个教师可以讲授多门课程；每名学生可以选修多门课程；学期结束后通过考试，教师登记每门课程、每名学生的成绩，并得到确认后存档。

数据库工程师风清扬设计了如图 13-20 所示的 E-R 图。



【问题 1】

把该 E-R 图中的实体和联系转化为相应的关系模式，并指出其主码、候选码以及外码(若无外码也须指出)。

【问题 2】

现要查询出所有的没有学生选修的课程的全部信息，请写出相应的 SQL 语句。注意，要查询出符合条件的课程的全部信息。

【问题 3】

现要查询出所有选修了教师号为 '001' 的老师所授课程的学生的学号与姓名，请写出相应的 SQL 语句。

答案： [问题 1]

方案 1:

教师(教师号, 姓名, 性别, 职称, 系别)主码: 教师号; 候选码: 教师号; 无外码

学生(学号, 姓名, 性别, 班级)主码: 学号; 候选码: 学号; 无外码

课程(课程号, 名称, 学时, 学分, 教师号)主码: 课程号; 候选码: 课程号; 外码: 教师号

选修(学号, 课程号, 成绩)主码: (学号, 课程号); 候选码: (学号, 课程号); 外码: 学号, 课程号

方案 2:

教师(教师号, 姓名, 性别, 职称, 系别)主码: 教师号; 候选码: 教师号; 无外码

学生(学号, 姓名, 性别, 班级)主码: 学号; 候选码: 学号; 无外码

课程(课程号, 名称, 学时, 学分)主码: 课程号; 候选码: 课程号; 无外码

讲授(课程号, 教师号)主码: 课程号; 候选码: 课程号; 外码: 教师号

选修(学号, 课程号, 成绩)主码: (学号, 课程号); 候选码: (学号, 课程号); 外码: 学号, 课程号

[问题 2]

SELECT * FROM C WHERE NOT EXISTS(SELECT Cno FROM SC WHERE SC.Cno=C.Cno)

```
SELECT * FROM C WHERE Cno NOT IN(SELECT Cno FROM SC WHERE SC.Cno=C.Cno)
```

注：C、S、SC、Cno、Sno、Tno 分别表示课程表、学生表、选修表、课程号、学号、教师号。亦可直接用汉字表示。其中的 * 可以用 C 的全部属性来表示。

[问题 3]

```
SELECT Sno Sname FROM S, C, SC
```

```
WHERE S.Sno=SC.Sno AND C.Cno=SC.Cno AND C.Tno='001'
```

```
SELECT Sno, Sname FROM S
```

```
WHERE sno IN(SELECT sno FROM SC WHERE cno IN(SELECT cno FROM C WHERE  
tno='001'))
```

注：C、S、SC、Cno、Sno、Sname、Tno 分别表示课程表、学生表、选修表、课程号、学号、学生姓名、教师号。亦可直接用汉字表示。

能推导出全部属性而且没有多余属性的属性组称为候选码。若候选码多于一个，则选定其中的一个为主码。在数据库设计中，对于一般的关系模式用“编号”作为主码，如在学生关系模式中用学号作为其主码。

关系模型的逻辑结构是一组关系模式的集合。ER 图是由实体、实体的属性和实体之间的联系三个要素组成的。将 ER 图中的实体、实体之间的联系转换为关系模式，一般遵循如下原则：

一个实体型转换为一个关系模式。实体的属性就是关系的属性，实体的码就是关系的码。

对于实体的联系，有以下几种不同的情况：

(1) 一个 1:1 联系可以转换为一个独立的关系模式，也可以与任意一端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，每个实体的码均是该关系的候选码。如果与某一端实体对应的关系模式合并，则需要在该关系模式的属性中加入另一个关系模式的码和联系本身的属性。

(2) 一个 1:n 联系可以转换为一个独立的关系模式，也可以与 n 端对应的关系模式合并。如果转换为一个独立的关系模式，则与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为 n 端实体的码。

(3) $m:n$ 联系转换为一个关系模式。与该联系相连的各实体的码以及联系本身的属性均转换为关系的属性，而关系的码为各实体码的组合。

(4)三个或三个以上实体间的以上多元联系可以转换为一个关系模式。与该多元联系相连的各实体的码以及联系本身的属性均转换为关系的属性。而关系的码为各实体码的组合。

由于教师和课程之间是1对多的关系，因此可以把讲授联系合并到课程中去。而学生与课程是多对多的关系，所以要把选修联系独立作为一个关系表。

关系模式 R 中的属性或属性组 X 并非 R 的码，但 X 是另一个关系模式的码，则称 X 是 R 的外部码，也称外码。例如，有关系模式： S (学号，姓名，年龄，系号)、 D (系号，系名，宿舍号)，那么“学号”是 S 的主键，“系号”是 S 的外码。

该题中，选修关系模式的主码是学号和课程号的组合，而其中的学号是学生表的主码，却不是选修表中的主码，所以学号是外码。课程号是课程表的主码，却不是选修表中的主码，所以课程表也是外码。

问题2比较简单，在 SC 中可以找到选修的全部课程，要想查询没有学生选修的课程，只要用谓词 $NOT EXISTS$ 或 $NOT IN$ 进行判断即可。因此，实现该查询的SQL语句可写成：

```
SELECT * FROM C WHERE NOT EXISTS(SELECT Cno FROM SC WHERE SC.Cno=C.Cno)
```

或

```
SELECT* FROM C WHERE Cno NOT IN(SELECTCno FROM SC WHERE SC.Cno=C.Cno)
```

其中 C 、 S 、 SC 、 Cno 、 Sno 、 Tno 分别表示课程表、学生表、选修表、课程号、学号、教师号。前一条SQL语句中的两个SELECT语句的关系类似于双层循环，第1个SELECT是外层循环，此SELECT每扫描一条记录，内层的SELECT Cno FROM SC WHERE $SC.Cno=C.Cno$ 执行一次，例如，现在的外层SELECT扫描到的记录的 $Cno='001'$ ，则内层的语句为： $SELECT Cno$ FROM SC WHERE $SC.Cno='001'$ ，这样实际是在 SC 中查找有没有关于'001号课程的选课记录，如果没有，则 $NOT EXISTS$ (成立)，输出记录。

问题3中的查询涉及到学生、教师、选修三个表中的属性，显然要进行连接查询：

```
SELECT Sno, Sname FROM S, C, SC
```

WHERE S.Sno=SC.Sno AND C.Cno=SC.Cno AND C.Tno='001'

WHERE 关联起来进行查询。或写成三层嵌套的形式：

SELECT Sno, Shame FROM S

WHERE sno IN(SELECT sno FROM SC WHERE cno IN(SELECT cno FROM C WHERE
tno =, 001,))

首先最内层的(SELECT cno FROM C WHERE tno='001')找出'001'号教师所授课程的课程号，再在 SC 中根据课程号找出选了这些课程的学生学号，最后根据学号在 S 中查找出学生的学号和姓名。

试题四

阅读以下说明，回答问题 1、问题 2 和问题 3。

1. 【说明】

某单位正在使用一套 C/S 模式的应用软件系统，现在需要升级为 B/S 应用模式，但需要保持业务的连续性。开发人员提出用 Web Service 作为中间层的接口进行开发。

【问题 1】

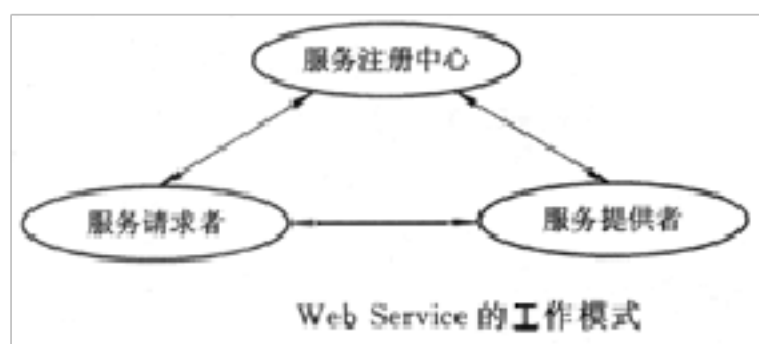
请用 120 字以内文字，从业务的继承性、升级成本(时间、工作量)和扩展性三个方面简要说明开发人员所提方案的优点。

【问题 2】

Web Service 的三个基本技术是 WSDL SOAP UDDI，它们都是以 XML 为基础定义的。请用 120 字以内文字，简要说明 WSDL SOAP 和 UDDI 的作用。

【问题 3】

服务注册中心、服务提供者和服务请求者之间的交互和操作构成了 Web Service 的体系结构，如图 13-21 所示。请用 180 字以内文字，说明这三者的主要功能及其交互过程。



[问题 1]

由于 **Web Service** 具有支持软件和数据重用、集成方便、支持对别人的服务拿来就用等特性，升级开发人员不必重新设计业务流程，只需将 **Web Service** 作为中间层的接口进行开发即可，从而较好地保持了业务的连续性，也大大减少了工作量并减低了升级成本，也不必修改原来的 **Web** 服务及中间层，易于扩展且扩展空间也较大。

[问题 2]

SOAP提供了标准的 **RPC**方法来调用 **Web Service** 协议,定义了服务请求者和服务提供者之间的消息传输规范。

WSDL为服务提供者提供了用 **XML**格式描述 **Web Service** 的标准格式。

UDDI提供了一种机制让 **Web** 服务提供商发布他们的产品，并最终让他们的客户能定位他们所提供的 **Web** 服务。

[问题 3]

服务提供者使用 **WSDL**来定义抽象的服务描述，然后把具体的服务发布到服务注册中心；服务请求者使用查找(**Find**)操作从服务注册中心检索服务描述，然后使用服务描述与服务提供者进行绑定(**Bind**)，并调用 **Web Service** 实现访问。服务注册中心是连接服务提供者和服务请求者的桥梁与纽带。

Web Service 是一套标准，它定义了应用程序如何在 **Web** 上实现互操作性，支持用不同的语言(如 **VB**、**Java** 等)在不同的平台上(如 **Windows**、**UNIX**、**Linux**等)编写 **Web Service**，而后通过 **Web Service** 的标准对外发布服务，其他用户或应用也通过 **Web Service** 的标准来对这些服务进行查询和后续的访问调用。

Web Service 由 **SOAP**(简单对象访问协议)、**WSDL**(服务描述语言)、**UDDI**(服务注册检索访问标准)三个协议有力地支持和实现：

(1)SOAP (Simple Object Access Protocol, SOAP)提供了标准的 RPC方法来调用 Web Service 协议,定义了服务请求者和服务提供者之间的消息传输规范。SOAP用 XML来格式化消息,用 HTTP来承载消息,它有很大的可扩展性和平台语言无关性,在各种平台上很容易实现。

(2)WSDL 服务描述语言(Web Service Description Language, WSDL)为服务提供者提供了用 XML 格式描述 Web Services 的标准格式,以表达一个 Web Service 能提供什么功能,它的位置在哪里,如何调用它等。

(3)UDDI 服务注册检索访问标准(Universal Discovery, Description, Integration, UDDI)提供了一种机制让 Web 服务提供商发布他们的产品,并最终让他们的客户能定位他们所提供的 Web 服务。其核心组件是 UDDI商业注册,它利用 WSDL语言来描述企业及其提供的 Web 服务。

Web Service 通过服务的建立、描述、发布、查找、调用等步骤来实现不同平台间服务的分布调用,具体描述如下:

(1)Web 服务的建立(Build): 可用不同的语言在不同的平台上开发 Web 服务。

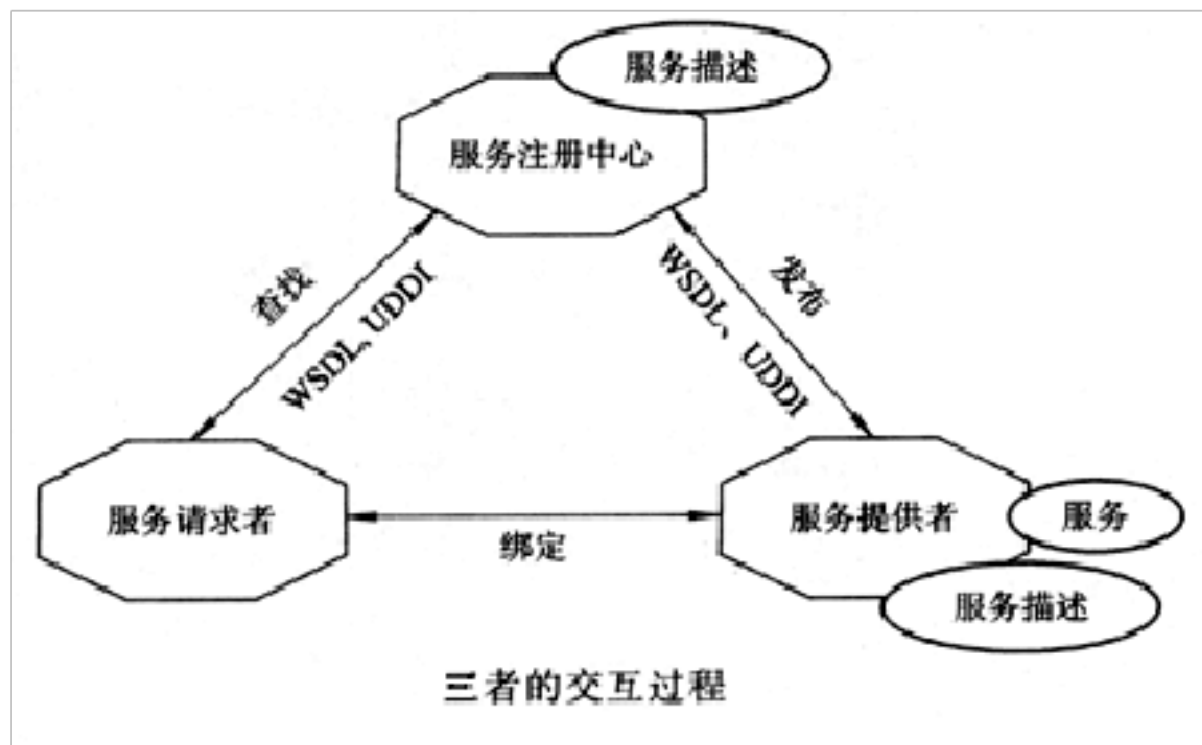
(2)Web 服务的描述(Description): Web 服务开发出来后,用 WSDL的标准来服务请求和响应的参数格式及其他协议相关的描述。

(3)Web 服务的发布(Publish): 为了使服务可访问,服务提供者需要首先将服务进行一定的描述并发布到注册服务器上。

(4)Web 服务的查找(Find): 服务请求方根据注册服务器提供的规范接口发出查询请求,以获取绑定服务所需的相关信息。

(5)Web 服务的调用(Bind): 服务请求方通过分析从注册服务器中得到的服务绑定信息,包括服务的访问路径、服务调用的参数、返回结果、传输协议、安全要求等,对自己的系统进行相应配置,进而远程调用服务提供者所提供的服务。

图 13-52 描述了服务请求者、服务提供者、服务注册中心这三者的交互过程。



Web Service

到这一目标，Web Service 完全基于 XML、XSD 等独立于平台、独立于软件供应商的标准，主要适用于企业内部不同应用的集成和数据交互、B2B 集成、代码和数据重用，以及通过 Web 进行客户端和服务器的通信的场合。

Web Service 允许用户在重用代码的同时，重用代码后面的数据。使用 Web Service 时，用户不用再像以前那样，要先从第三方购买、安装软件组件，再从应用程序中调用这些组件，而只需要直接调用远端的 Web Service 就可以了。举个例子，如果想在应用程序中确认用户输入的邮件地址，那么只需把这个地址直接发送给相应的 Web Service，这个 Web Service 就会查阅街道地址、城市、省区和邮政编码等信息，确认这个地址的确在相应的邮政编码区域。Web Service 的提供商可以按时间或使用次数对这项服务进行收费。这样的服务要通过组件重用来实现是不现实的，因为那样的话，必须下载并安装好包含街道地址、城市、省区和邮政编码等信息的数据库，而且这个数据库还是不能实时更新的。另一种软件重用的情况是，把好几个应用程序的功能集成起来。例如，想要建立一个局域网上的门户网站应用，让用户既可以查看股市行情，又可以管理他们的日程安排，还可以在线购买电影票。现在 Web 上有很多应用程序供应商，都在其应用中实现了上面的这些功能。一旦他们把这些功能都通过 Web Service 发布出来，就可以非常轻易地把所有这些功能都集成到门户网站中，为用户提供一个统一的、友好的界面。

可见，因为 Web Service 具有上述特性，所以使用它会较好地保持业务的连续

性，也正因为 **Web Service** 具有软件重用、支持对别人的服务拿来就用等特性，使得升级的工作量比单独开发要少得多、成本也低得多，不但易于扩展而且扩展空间也比较大。

试题五

阅读以下说明和 C++ 程序代码，将程序填充完整。

1. 【说明】

有时我们希望给某个对象而不是整个类添加一些功能。例如，一个图形用户界面工具箱允许你对任意一个用户界面组件添加一些特性，例如边框，或是一些行为，例如窗口滚动。使用继承机制是添加功能的一种有效途径，从其他类继承过来的边框特性可以被多个子类的实例所使用。但这种方法不够灵活，因为边框的选择是静态的，用户不能控制对组件加边框的方式和时机。

一种较为灵活的方式是将组件嵌入另一个对象中，由这个对象添加边框。我们称这个嵌入的对象为装饰。这个装饰与它所装饰的组件接口一致，因此它对使用该组件的客户透明。它将客户请求转发给该组件，并且可能在转发前后执行一些额外的动作(例如画二个边框)。透明性使得你可以递归地嵌套多个装饰，从而可以添加任意多的功能。

装饰对象结构模式的意图就是动态地给一个对象添加一些额外的职责。就增加功能来说，该模式相比生成子类更为灵活。其示意类图如图 13-22 所示。

以上内容仅为本文档的试下载部分，为可阅读页数的一半内容。如要下载或阅读全文，请访问：<https://d.book118.com/777014201153010006>