全国计算机技术与软件专业技术资格（水平）考试

中级 软件设计师 2012 年 下半年 下午试卷 案例

（考试时间 150 分钟）

试题一 某电子商务系统采用以数据库为中心的集成方式改进购物车的功能，详细需求如下：

1:加入购物车。顾客浏览商品，点击加入购物车，根据商品标识从商品表中读取商品信息，并更新购物车表。

2:浏览购物车。顾客提交浏览购物车请求后，显示出购物车表中的商品信息。

3:提交订单。顾客点击提交订单请求，后台计算购物车表中商品的总价(包括运费)加入订单表，将购物车表中的商品状态改为待付款，显示订单详情。若商家改变价格，则刷新后可看到更改后的价格。

4:改变价格。商家查看订购自家商品的订单信息，根据特殊优惠条件修改价格，更新订单表中的商品价格。

5:付款。顾客点击付款后，系统先根据顾客表中关联的支付账户，将转账请求(验证码、价格等)提交给支付系统(如信用卡系统)进行转账；然后根据转账结果返回支付状态并更改购物车表中商品的状态。

6:物流跟踪。商家发货后，需按订单标识添加物流标识(物流公司、运单号)；然后可根据顾客或商家的标识以及订单标识，查询订单表中的物流标识，并从相应物流系统查询物流信息。

7:生成报表。根据管理员和商家设置的报表选项，从订单表、商品表以及商品分类表中读取数据，调用第三方服务 CrystalReports 生成相关报表。

8:维护信息。管理员维护(增、删、改、查)顾客表、商品分类表和商品表中的信息。

现采用结构化方法实现上述需求，在系统分析阶段得到如图 1-1 所示的顶层数据流图和图 1-2 所示的 0 层数据流图。

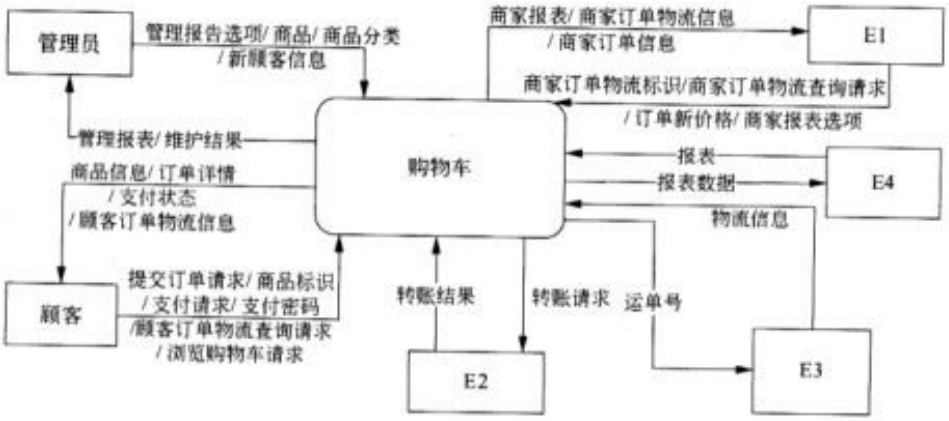


图 1-1 顶层数据流图

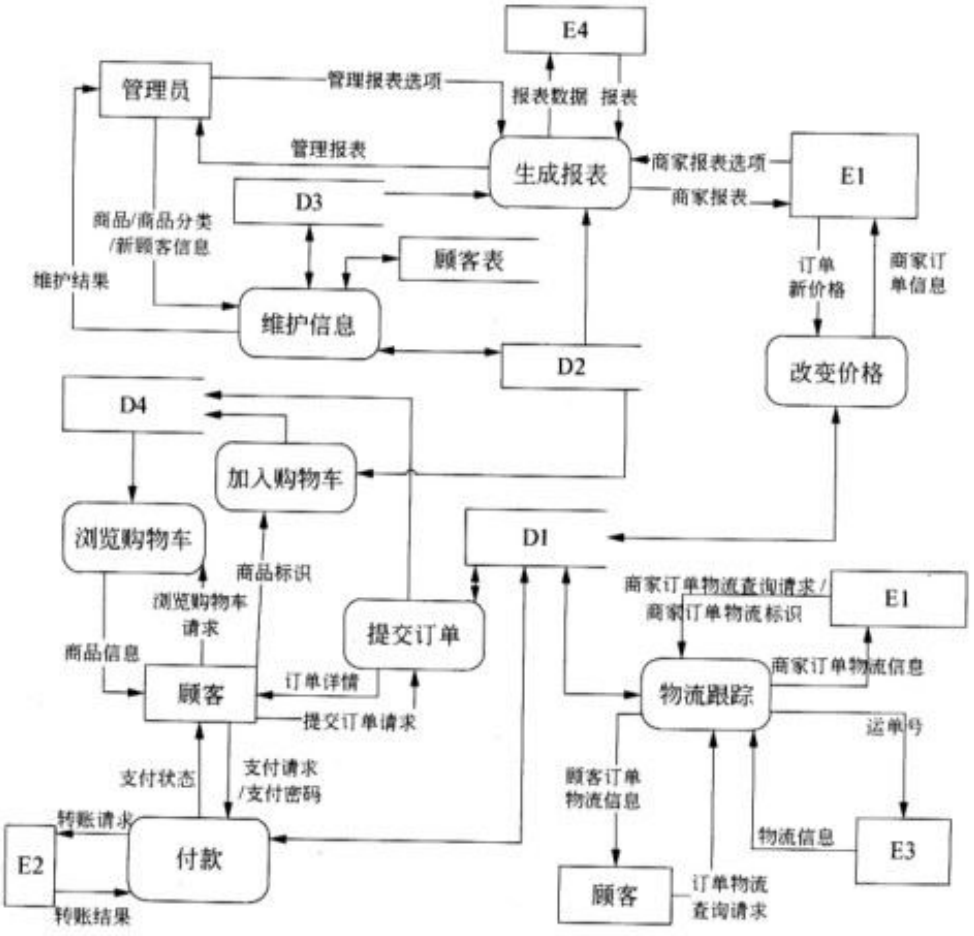


图 1-2 0 层数据流图

问题：1.1

使用说明中的词语，给出图 1-1 中的实体 E1 E4 的名称。

问题：1.2

使用说明中的词语，给出图 1-2 中的数据存储 D1 D4 的名称。

问题：1.3

图 1-2 中缺失了数据流，请用说明或图 1-2 中的词语，给出其起点和终点。

问题：1.4
根据说明，给出数据流“转账请求”、“顾客订单物流查询请求”和“商家订单物流查询请求”的各组成数据项。

试题二 某会议策划公司为了方便客户，便于开展和管理各项业务活动，需要构建一个基于网络的会议预定系统。

【需求分析】

会议策划公司设有受理部、策划部和其他部门。部门信息包括部门号、部门名称、部门主管、电话和邮箱号。每个部门有多名员工处理部门的日常事务，每名员工只能在一个部门工作。每个部门有一名主管负责管理本部门的事务和人员。

员工信息包括员工号、姓名、部门号、职位、联系方式和工资；其中，职位包括主管、业务员、策划员等。业务员负责受理会议申请。若申请符合公司规定，则置受理标志并填写业务员的员工号。策划部主管为已受理的会议申请制定策划任务，包括策划内容、参与人数、要求完成时间等。一个已受理的会议申请对应一个策划任务，一个策划任务只对应一个已受理的会议申请，但一个策划任务可由多名策划员参与执行，且一名策划员可以参与多项策划任务。

客户信息包括客户号、单位名称、通信地址、所属省份、联系人、联系电话、银行账号。其中，一个客户号唯一标识一个客户。一个客户可以提交多个会议申请，但一个会议申请对应唯一的一个客户号。

会议申请信息包括申请号、开会日期、会议地点、持续天数、会议人数、预算费用、会议类型、酒店要求、会议室要求、客房类型、客房数、联系人、联系方式、受理标志和业务员的员工号等。客房类型有豪华套房、普通套房、标准间、三人间等，且申请号和客房类型决定客房数。

【概念模型设计】

根据需求阶段收集的信息，设计的实体联系图和关系模式(不完整)如下：

【关系模式设计】

部门(部门号，部门名称，主管，电话，邮箱号)

员工(员工号，姓名，（a）,联系方式，工资)

客户(客户号，单位名称，通信地址，所属省份，联系人，联系电话，银行账号)

会议申请(（b）,开会日期，会议地点，持续天数，会议人数，预算费用，会议类型，酒店要求，会议室要求，客房数，联系人，联系方式，受理标志，员工号)

策划任务(（c）,策划内容，参与人数，要求完成时间)

执行策划(（d）,实际完成时间)

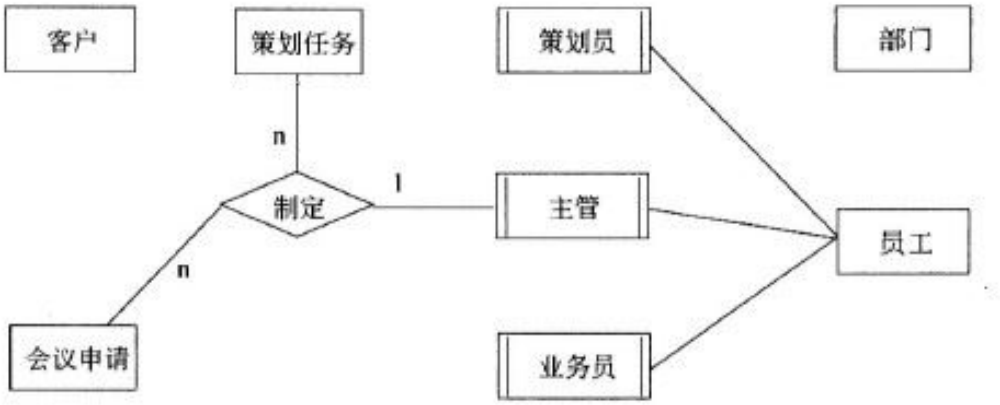


图 2-1　实体联系图

问题：2.1
根据问题描述，补充五个联系、联系的类型，完善图 2-1 的实体联系图
问题：2.2
根据实体联系图，将关系模式中的空(a) ~ (d)补充完整(1 个空缺处可能有多个数据项)。对会议申请、策划任务和执行策划关系模式，用下划线和#分别指出各关系模式的主键和外键。
问题：2.3
请说明关系模式“会议申请”存在的问题及解决方案。

试题三　某城市的各国家公园周边建造了许多供游客租用的小木屋和营地，为此，该城市设置了一个中心售票处和若干个区域售票处。游客若想租用小木屋或营地，必须前往中心

售票处进行预定并用现金支付全额费用。所有的预定操作全部由售票处的工作人员手工完成。现欲开发一信息系统，实现小木屋和营地的预定及管理功能，以取代手工操作。该系统的主要功能描述如下：

1. 管理预定申请。游客可以前往任何一个售票处提出预定申请。系统对来自各个售票处的预定申请进行统一管理。

2. 预定。预定操作包含登记游客预定信息、计算租赁费用、付费等步骤。

3. 支付管理。游客付费时可以选择现金和信用卡付款两种方式。使用信用卡支付可以享受3%的折扣，现金支付没有折扣。

4. 游客取消预定。预定成功之后，游客可以在任何时间取消预定，但需支付赔偿金，剩余部分则退还给游客。赔偿金的计算规则是，在预定入住时间之前的 48 小时内取消，支付租赁费用 10%的赔偿金；在预定入住时间之后取消，则支付租赁费用 50%的赔偿金。

5. 自动取消预定。如果遇到恶劣天气(如暴雨、山洪等)，系统会自动取消所有的预定，发布取消预定消息，全额退款。

6. 信息查询。售票处工作人员查询小木屋和营地的预定情况和使用情况，以判断是否能够批准游客的预定申请。

现采用面向对象方法开发上述系统，得到如表 3-1 所示的用例列表和表 3-2 所示的类列表。对应的用例图和类图分别如图 3-1 和 3-2 所示。

表 3-1　用例列表

用 例 名	说　明	用 例 名	说　明
ManageInquiries	管理预定申请	ManageCashPayment	现金支付
MakeReservation	预定	ManageCrCardPayment	信用卡支付
ManagePayment	支付管理	GetDiscount	计算付款折扣
CancelReservation	游客取消预定	AutoCancelReservation	系统自动取消预定
CheckAvailability	信息查询	CalculateRefund	计算取消预定的赔偿金
PublishMessage	发布取消预定消息		

表 3-2　类列表

类　名	说　明	类　名	说　明
NationalPark	国家公园	Customer	游客
Reservation	预定申请	ReservationItem	预定申请内容
TicketingOfficer	售票处	CampSite	营地
Bungalow	小木屋	Payment	付款
Discount	付款折扣	CashPayment	现金支付
CreditCardPayment	信用卡支付	Rate	租赁费用

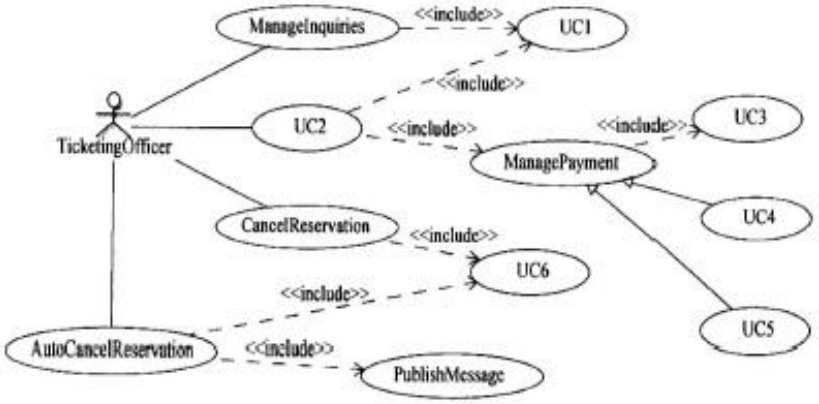


图 3-1　用例图

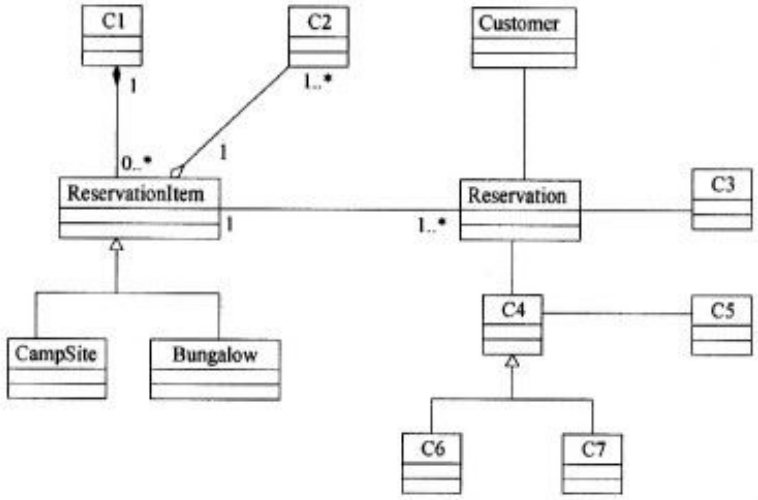


图 3-2　类图

问题：3.1
根据说明中的描述与表 3-1，给出图 3-1 中 UC1　UC6 处所对应的用例名称。
问题：3.2
根据说明中的描述与表 3-2, 给出图 3-2 中 C1　C7 处所对应的类名。
问题：3.3
对于某些需求量非常大的小木屋或营地，说明中功能 4 的赔偿金计算规则，不足以弥补取消预定所带来的损失。如果要根据预定的时段以及所预定场地的需求量，设计不同层次的赔偿金计算规则，需要对图 3-2 进行怎样的修改？(请用文字说明)

试题四　设有 n 个货物要装入若干个容量为 C 的集装箱以便运输，这 n 个货物的体积分别为 $\{s_1, s_2, \cdots, s_n\}$，且有 S_i
下面分别采用最先适宜策略和最优适宜策略来求解该问题。

最先适宜策略(*firstfit*)首先将所有的集装箱初始化为空，对于所有货物，按照所给的次序，每次将一个货物装入第一个能容纳它的集装箱中。

最优适宜策略(*bestfit*)与最先适宜策略类似，不同的是，总是把货物装到能容纳它且目前剩余容量最小的集装箱，使得该箱子装入货物后闲置空间最小。

【*C* 代码】

下面是这两个算法的 *C* 语言核心代码。

*(1)*变量说明

n：货物数

C：集装箱容量

s：数组，长度为 *n*，其中每个元素表示货物的体积，下标从 *0* 开始

b：数组，长度为 *n*，*b[i]*表示第 *i+1* 个集装箱当前已经装入货物的体积，下标从 *0* 开始

i，*j*:循环变量

k：所需的集装箱数

min：当前所用的各集装箱装入了第 i 个货物后的最小剩余容量

m：当前所需要的集装箱数

$temp$：临时变量

(2) 函数 $firstfit$

```
int firstfit() {
        int i, j;
        k = 0;
        for(i = 0; i < n; i++) {
```

```
        for(i = 0; i < n; i++){
            ____(1)____;
            while(C - b[j] < s[i]){
              j++;
            }
            ____(2)____;
            k = k > (j+1) ? k : (j+1) ;
        }
        return k;
}
```

（3）函数 bestfit

```
int bestfit()  {
        int i, j, min, m, temp;
        k = 0;
        for(i = 0; i < n; i++) {
          b[i] = 0;
        }
        for(i = 0; i < n; i++) {
          min = C;
          m = k + 1;
          for(j = 0; j < k + 1; j++) {
            temp = C - b[j] - s[i];
            if(temp > 0 && temp < min){
                ____(3)____;
                m = j;
            }
          }
          ____(4)____;
          k = k > (m+1) ? k : (m+1) ;
        }
        return k;
}
```

问题： 4.1

根据【说明】和【C代码】，填充C代码中的空(1)（4)。

问题： 4.2

根据【说明】和【C 代码】，该问题在最先适宜和最优适宜策略下分别采用了(5)和(6)算法设计策略，时间复杂度分别为 (7) 和(8)(用 0 符号表示)。

问题： 4.3

考虑实例 n= 10, C= 10, 各个货物的体积为{4,2,7,3,5,4,2,3,6,2}。该实例在最先适 宜和最优适宜策略下所需的集装箱数分别为(9)和(10)。考虑一般的情况，这两种求解策略能否确保得到最优解？ (11)(能或否)

试题五 现欲开发一个软件系统，要求能够同时支持多种不同的数据库，为此采用抽象工厂模式设计该系统。以 SQLServer 和 Access 两种数据库以及系统中的数据库表 Department 为例，其类图如图 5-1 所示。

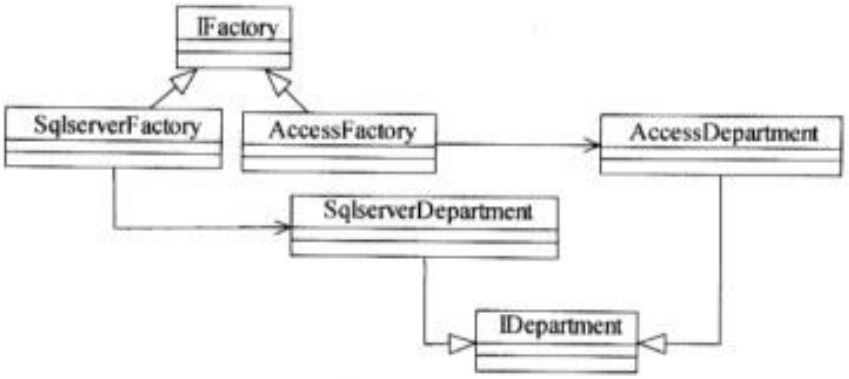


图 6-1 类图

问题： 5.1

【C++代码】

```
#include <iostream>
using namespace std;

class Department { /* 代码省略 */ };
class IDepartment {
public:

    (1)     = 0;
    (2)     = 0;
};

class SqlserverDepartment :    (3)    {
public:
   void Insert(Department* department) {
        cout << "Insert a record into Department in SQL Server!\n";
// 其余代码省略
}
   Department GetDepartment(int id) {
        /* 代码省略 */
}
};

class AccessDepartment :    (4)    {
public:
   void Insert(Department* department) {
        cout << "Insert a record into Department in ACCESS!\n";
        // 其余代码省略
}
   Department GetDepartment(int id) {
        /* 代码省略 */
}
};

    (5)    {
public:
    (6)    = 0;
};

class SqlServerFactory : public IFactory {
public:
   IDepartment* CreateDepartment() { return new SqlserverDepartment(); }
```

```
class SqlServerFactory : public IFactory {
public:
   IDepartment*CreateDepartment(){  return new SqlserverDepartment();  }
// 其余代码省略
};

class AccessFactory : public IFactory {
public:
   IDepartment* CreateDepartment() {  return new AccessDepartment(); }
// 其余代码省略
};
```

试题六　现欲开发一个软件系统，要求能够同时支持多种不同的数据库，为此采用抽象工厂模式设计该系统。以 SQLServer 和 Access 两种数据库以及系统中的数据库表 Department

为例，其类图如图 6-1 所示。

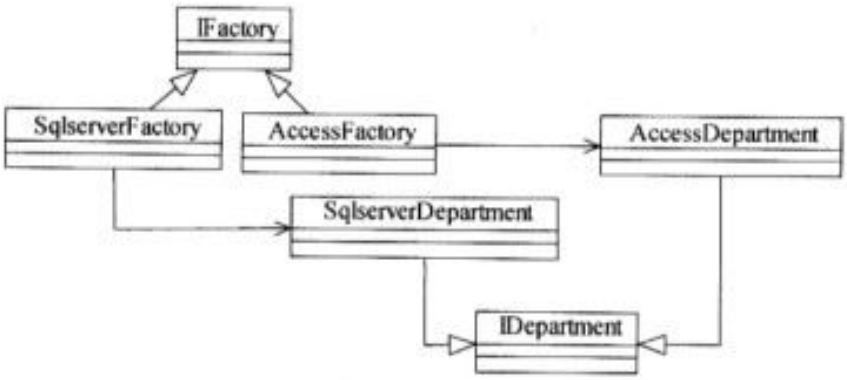


图 6-1　类图

问题：6.1

【Java 代码】

```
import java.util.*;

class Department {  /* 代码省略 */  }

interface IDepartment {
    ____(1)____;
    ____(2)____;
}

class SqlserverDepartment ____(3)____ {
   public void Insert(Department department) {
      System.out.println("Insert a record into Department in SQL Server!");
      // 其余代码省略
   }
   public Department GetDepartment(int id) {
/*  代码省略 */
   }
}

class AccessDepartment ____(4)____ {
   public void Insert(Department department) {
       System.out.println("Insert a record into Department in ACCESS!");
       // 其余代码省略
   }
```

```
    }
    public Department GetDepartment(int id) {
/*  代码省略   */
    }
}

    (5)      {
      (6)      ;
}

class SqlServerFactory implements IFactory {
    public IDepartment CreateDepartment() {
        return new SqlserverDepartment();
    }
    // 其余代码省略
}

class AccessFactory implements IFactory {
    public IDepartment CreateDepartment() {
        return new AccessDepartment();
    }
    // 其余代码省略
}
```

试题一 答案：　解析： E1:商家. E2:支付系统. E3:物流系统. E4:CrystalReport 或第三方服务.

解析：本问题考查顶层 DFD 。顶层 DFD 一般用来确定系统边界，将待开发系统看作一个加工，图中只有唯一的一个处理和一些外部实体，以及这两者之间的输入输出数据流。题目要求根据描述确定图中的外部实体。外部实体可以是和系统交互的人，以及和系统交互的外部系统或服务。分析题目中的描述，并结合已经在顶层数据流图中给出的数据流进行分析。分析题目中的说明，管理员维护系统中信息，顾客和商家是系统的主要使用者；商家查看订购自家商品的订单信息，根据特殊优惠条件修改价格，更新订单表中的商品价格，还可以添加物流标识并进行物流跟踪；使用支付系统进行支付，通过物流系统进行物流跟

踪，以及第三方服务 CrystalReport 生成报表。可以看出，和系统的交互者钮括管理员、顾客、商家三类人，支付系统、物流系统和 CrystalReport 三种外部系统。

对应图 1-1 中数据流和实体的对应关系，管理员和顾客已经给出，可知 E1 为商家， E2 为支付系统， E3 为物流系统， E4 为第三方服务 CrystalReport 。

D1 ：订单表.D2:商品表.D3:商品分类表.D4:购物车表.

本问题考查 0 层 DFD 中数据存储的确定。根据说明中所描述的处理和相关数据存储之间的连接关系，判定每个数据存储。加入购物车和浏览购物车分别读取和更新购物车表中的数据；改变价格和提交订单要读取和更新订单表中的数据；维护信息时需要维护商品表和商品分类表，生成报街要读取商品表和商品分类表，加入购物车时，需要读取商品表中的商品信息。

根据描述和图 1-2 中的数据存储的输入输出数据流提示，可知： D1 为订单表， D2 为商品表， D3 为商品分类表， D4 为购物车表。

本问题考查绘制 0 层 DFD 时是否将本层该绘制的数据流全部绘制出。对照顶层数 据流图和 0 层数据流图，检查是否和外部实体之间的数据流一致；仔细对照说明中的描述和图 1-2 中给出的数据流，检查是否遗漏掉信息。说明中：提交订单处理时，后台计算购物车表中的商品的总价，即需要读出购物车表中的相关价格进行计算，读取出其中数据；付款需要读取顾客表中关联的支付账户，并向支付系统提交转账请求，然后根据转账结果更改购物车表中商品的状态；生成报告时根据管理员和商家设置的报告选项，从订单表、商品表以及商品分类表中读取数据，再调用第三方服务 CrystalReports 生成相关报告。将这些说明和图 1-2 进行对照，发现缺少了从付款到购物车表(D4)、从购物车表到提交订单、从顾客表到付款，以及从订单表(D1)到生成报表等 4 条数据流。

转账请求=验证码+价格+账号信息 顾客订单物流查询请求=顾客标识+订单标识

商家订单物流查询请求=商家标识+{订单标识}

解析：本问题考查在绘制数据流图时数据流的数据项组成。数据流图描述了系统的分解，但它并没有给出图中各成分的说明。通常采用数据字典为数据流图中的每个数据流、文件、处理，以及组成数据流或文件的数据项做出说明。对于数据流，通常列出该数据流的各组成数据项，并采用数据字典定义式中出现的符号进行表示，如“=”表示“被定义为”，“+”表示“与” “{……}广表示其中数据可以有多个等等。本试题说明中：付款时，需根据顾客表中关联的支付账户将转账请求(验证码、价格等)提交给支付系统；物流跟踪时，根据顾客和商家的标识以及订单标识进行查询，而且在改变价格时商家查 看订购自家商品的订单信息，可知商家可以查询一批订单。可以看出，提交给支付系统的请求中包含支付账户、验证码与价格；顾客订单查询请求中有顾客标识、订单标识；商家订单查询请求中有商家标识、订单标识(一批订购自家商品的订单标识)。因此“转账请求=支付账

户+验证码+价格”；“商家订单物流查询请求=物流标识+{订单标识} ”；“顾客订单物流标识=物流标识+订单标识”。

图 1-2 中缺少的数据流：

起点	终点
付款	D4 或购物车表
D4 或购物车表	提交订单
顾客表	付款
D1 或订单表	生成报表

试题二 答案： 解析：

根据题意，一个客户可以提交多个会议申请，但一个会议申请对应唯一的一个客户号， 故应在客户和会议申请之间增加一个 1 :n 的“提交”联系：由于业务员负责受理会议申请，若申请符合公司规定则置受理标志并填写业务员的员工号，因此业务员和会议申请之间有一个 1 :n 的“受理”联系；由于一个已受理的会议申请对应一个策划任务，一个策划任务只对应一个已受理的会议申请，但一个策划任务可由多名策划员参与执行，且一名策划员可以参与多项策划任务，因此策划任务和策划员之间有一个 n: m 的“执行”联系；由于每个部门有多名员工处理部门的日常事务，每名员工只能在一个部门工作，因此部门和员工之间有一个 1 :n 的“所属”联系；又由于每个部门有一名主管负责管理本部门的事务和人员，而该主管也是一名员工，因此主管和部门之间有一个 1 : 1 的“管理”联系。

a)部门号，职位

b)申请号，客房类型，客户号

c)申请号，员工号

d)申请号，员工号

关系模式为：

会议申请(申请号，客房类型，客户号#，开会日期，会议地点，持续天数，会议人数，预算费用，会议类型，酒店要求，会议室要求，客房数，联系人，联系方式，受理标志，员工号#)

策划任务(申请号#，员工号#，策划内容，参与人数，要求完成时间)

执行策划(申请号#，员工号#，实际完成时间)

解析：根据题意，在员工关系模式中，部门与员工之间是一个 1 :n 的联系，需要将 1 端(即部门)的码“部门号”并入员工关系；又因为每个员工担任相应职位，故员工关系模式

欢迎添加“职位”属性；可见，空(a)应填写“部门号，职位”。

在会议申请关系模式中，由于申请号、客房类型、客户号为主键，故空(b)应填写“申请号，客房类型，客户号”；在策划任务关系模式中，申请号、员工号为主键，故空(c)应填写“申请号，客户号”；由于一个策划任务可由多名策划员参与执行，且一名策划员可以参与多项策划任务，故在执行策划关系模式中，执行策划又由于一个业务员可以安排多个托运申请，申请号、员工号为主键，故空(d)应填写“申请号，客户号”。

会议申请关系模式的主键为“申请号，客房类型”，因为，申请号、客房类型能唯一标识该关系模式的每一个元组。会议申请关系模式的外键为客户号及员工号，因为， 客户号及员工号分别为客户及员工关系模式的主键，故为该关系模式的外键。

策划任务关系模式的主键为申请号，因为，申请号能唯一标识该关系模式的每一个元组，故申请号为该关系模式的主键。策划任务关系模式的外键为员工号，因为，员工 号为员工关系模式的主键，故为该关系模式的外键。

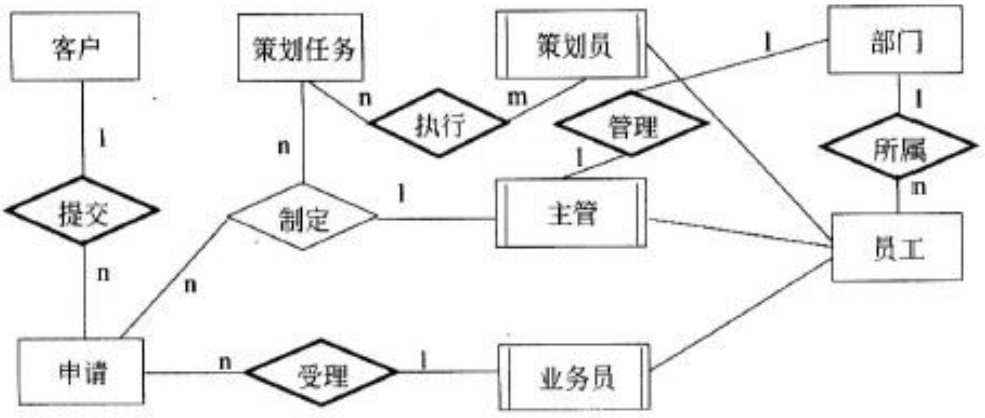
执行策划关系模式的主键为“申请号，员工号”，因为，申请号 k 员工号能唯一标识该关系模式的每一个元组，故“申请号，员工号”为该关系模式的主键。执行策划关 系模式的外键为申请号及员工号，因为，申请号和员工号分别为会议申请和员工关系模式的主键，故为该关系模式的外键。

会议申请存在数据冗余及数据修改的不一致性问题，应该将关系模式分解为如下两个模式：

会议申请 1（申请号，客户号，开会日期，会议地点，持续天数，会议人数，预算费用，会议类型，酒店要求，会议室要求，联系人，联系方式，受理标志，员工号）

会议申请 2（申请号，客房类型，客房数)。

关系模式“会议申请”存在数据冗余及数据修改的不一致性问题，应该将关系模式分解。



试题三 答案：　解析： UC1 ： CheckAvailablity

UC3 ： GetDiscount

UC5 ： ManageCrCardPayment

UC2： MakeReservation

UC4 ： ManageCashPayment

UC6 ： CalculateRefund

UC4 和 UC5 可以互换。

本题要求将图 3-1 所给出的用例图补充完整。题目说明中已经给出了所有可能的用例的列表(如表 3-1 所示)。这就省去了寻找用例的步骤，只需要依据用例列表中给出的 用例，在说明中确定用例与 Actor 之间的关系即可将图补充完整。

用例图的构成要素有：参与者(Actor)、用例(Usecase)以及用例之间的关系。题目中的信息系统的主要用户是售票处的工作人员(TicketingOfficer),所以在图 3-1 中只给出了 1 个参与者。由说明可知，售票处工作人员利用该系统可以实现 6 个主要的功能： 管理预定申请(Managelnquiries)、预定(MakeReservation)、支付管理(ManagePayment)、 游客取消预定(CancelReservation)、自动取消预定(AutoCancelReservation)和信息查询(CheckAvailability)。其中“管理预定申请”、“支付管理”、“游客取消预定”、“自动取消预定”和“支付管理”均已经出现在图 3-1 中。支付租赁费用是预定过程中的一个必要步骤，而 UC2 和“支付管理”之间又是“include”关系，可以推断出 UC2 应该对应用例“预定(MakeReservation)”。那么用例“管理预定申请”和“预定”具有的相同步骤就是 UC1 所对应的用例，由此推断出 UC1 对应用例“信息查询(CheckAvailability)”。

由功能“支付管理”的说明可知，它具备两个能力：管理支付方式(信用卡或现金)以及计算折扣。 UC4 和 UC5 与用例“支付管理”之间是概括关系，说明 UC4 和 UC5 是 支付方式的两个特化，所以 UC4 为“现金支付(MangeCashPayment)”， UC5 为“信用卡支付(ManageCrCardPayment)”。 UC3 对应“计算折扣(GetDiscount)”。

这时用例列表中只剩下用例 CalcuateRefimd （计算取消预定的赔偿金)没有出现在图中了，那么它就是 UC6 对应的用例。从图 3-1 来看， UC6 应该表示用例“游客取消预定(CancelReservation)”和“自动取消预定(AutoCancelReservation)”中包含的公共事件流。不管是哪种类型的取消预定，都需要计算赔偿金，以决定退还给用户的费用，所以 UC6 对应用例 CalcuateRefimd 。

Cl:NationalPark. C2:Rate. C3:TicketingOfficer.C4:Payment. C5:Discount. C6; CashPaymen. C7 :CreditCardPaymentC6 和 C7 可以互换。

本题考察的是类图建模。题目中已经给出了类的列表，要求考生根据说明指出每个类在类图中的位置。在解题时，可以同时参考用例图中给出的信息。

先整体地看一下类图，寻找其中是否包含继承、聚集或组装等这些层次结构，这是快速确定部分类的关键。在图 3-2 中有一个继承结构： C4 、 C6 和 C7 。在图 3-1 中，用 例之间

也有一个概括的关系，这就提示我们， C4 、 C6 和 C7 这 3 个类一定与支付功能相关。在表 3-2 中寻找与支付功能相关的类： Payment 、 CashPayment 和 CreditCardPayment 。下来就是确定这 3 个类中，哪个是父类。很明显， Payment 应该作为父类。因此 C4 对应 Payment，C6 对应 CashPayment ， C7 对应 CreditCardPayment (C6 和 C7 可以互换支付管理中还有一项计算折扣的能力，类列表中的类 Discount 表示付款折扣，而与 C5 与 C4 之间具有关联关系，所以 C5 应该对应类 Discout 。

Cl 、 C2 分别与类“Reservationltem”之间具有组装和聚集的关系，而从说明中可知，具有这种整体部分关系的只有公园、预定及租赁费用之间，所以 C1 对应 NationalPark，C2 对应 Rate 。最后的一个类 C3 对应 TicketingOfficer ，即用例图中的 Actor 。

解答 1:增加一个新的类，该类与类 Reservationltem 之间有关联关系。或解答 2:修改 Rate 类，使其具有计算赔偿金的功能。

在面向对象方法中，好的类模型对需求的变化应该具有一定的适应性。本题考察的就是这一点。根据题目，现在对原有的赔偿金计算规则要进行修正。除了考虑取消预定的时间之外，同时要考虑所预定的小木屋或营地的地段以及需求量。修正类模型时通常两种基本方式，一种是修改已有的类，使其适应新的需求；第二种是增加一个新的类来完成新的需求，但是需要同时考虑新增加的类与已有类之间的关系。这道题目两种修改方法都可以采用。

若要修改已有的类，需要首先了解哪个类与现在的新需求是有相关性的。新需求针对的是赔偿金，赔偿金又与租赁费用相关，所以要找原先与租赁费用相关的那个类，即 Rate 。解决方案之一就是修改 Rate ，使其能够按照新的规则计算赔偿金。

第二种修改方式，增加一个专门计算赔偿金的类。按照新的计算规则，这个类就与游客的每次预定内容相关，因此这个新增加的类应该与类 Reservationltem 之间有关联关系。

试题四 答案： 解析： (1) j = 0

(2) b[j] = b[j] + s[i］及其等价形式

(3) min = temp

(4) b[m] = b[m] + s[i]及其等价形式

根据最先适宜算法思想，每取出一个货物，从第一个集装箱开始判断该货物是否能放入集装箱，若能则放入，因此空(1) 填 j = 0 。 while 循环判断，若货物不能放入集装 箱，则考虑下一个集装箱。不满足 while 循环中的条件，说明货物能放入集装箱，因此空(2) 填 b[j] = b[j] + S[i]。根据最优适宜算法思想，每取出一个货物，从第一个集装箱开始，确定能放入该货物且剩余容量最小的集装箱，并把该货物放入该集装箱中。 if 条件判断，

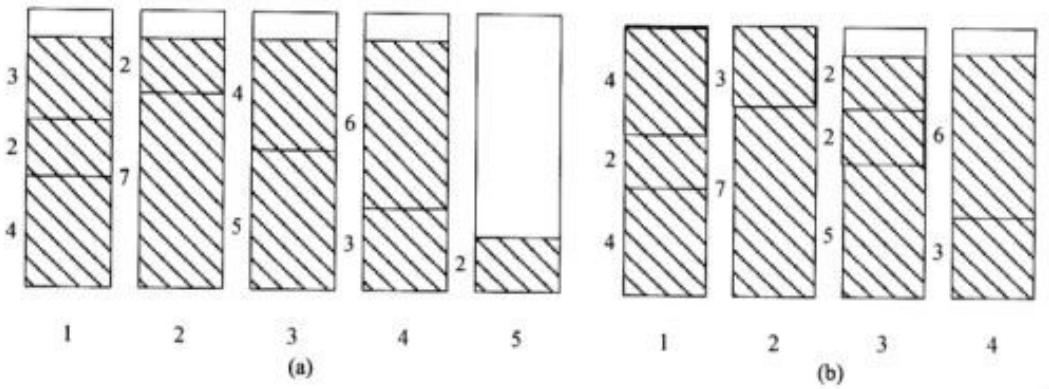
若找到了比能放入货物且剩余容量更小的集装箱，则剩余容量最小值改为当前的集装箱的剩余容量，因此空(3) 填 min = temp 。确定了集装箱后，把货物装入集装箱中，空(4) b[m] = b[m] + s[i]。

(5) 贪心(6) 贪心(7) $0(n^2)$ (8) $0(n^2)$

最先适宜算法总是把货物放入第一个能放入的集装箱，最优适宜算法总是把货物放入能容纳该货物且剩余容量最小的集装箱，因此都是基于贪心策略进行的，空(5) 和(6) 填贪心。函数 firstfit 中的 for 循环考虑 n 个货物，其中嵌套了 while 循环，最多的集装箱数为 n，因此时间复杂度为 $0(n^2)$。函数 bestfit 中的 for 循环考虑 n 个货物，其中嵌套了 for 循环检查每个集装箱的剩余容量，最多的集装箱数为 n，因此时间复杂度为 $0(n^2)$。

(9) 5 (10) 4(11) 否

对实例 n = 10, C = 10, S = {4,2,7,3,5,4,2,3,6,2}, 根据最先适宜和最优适宜算法，其具体的装箱方案分别如下图(a)和(b)所示。



试题五 答案：　解析： 1、virtualvoidInsert(Department* department)

2、virtualDepartmentGetDepartnient(intid)

3、publicIDepartment

4、publicIDepartment

5、classIFactory

6、virtualIDepartment* CreateDepartment()

本题考查抽象工厂(AbstractFactory)模式的概念及应用。

AbstractFactory 模式的意图是，提供一个创建一系列相关或相互依赖对象的接口，而无

需指定它们具体的类。 AbstractFactor 模式的结构如下图所示。

其中，类 AbstractFactory 声明一个创建抽象产品对象的操作接口；类 ConcreteFactory 实现创建具体产品对象的操作；类 productA 为一类产品对象声明一个接口；类 ConcreteProduct 具有 2 个功能：定义一个将被相应的具体工厂创建的产品对象；实现 ProductA 接口；类 Client 仅使用由 AbstractFactory 和 AbstractProduct 类声明的接口。

在以下情况可以使用 AbstractFactory 模式：

(1) 一个系统要独立于它的产品的创建、组合和表示时；

(2) 一个系统要由多个产品系统中的一个来配置时；

(3) 当要强调一系列相关的产品对象的设计一边进行联合使用时；

(4) 提供一个产品类库，而只想显示它们的接口而不是实现时。

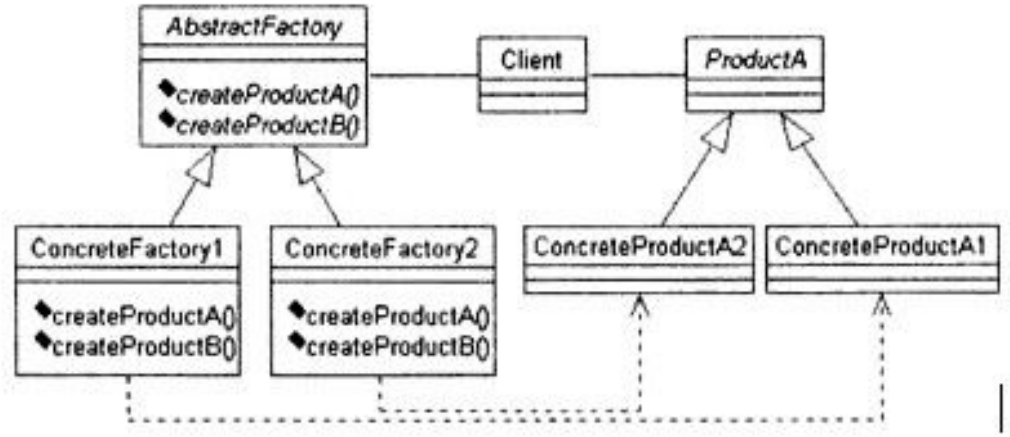
题目利用抽象工厂模式来解决在同一个软件系统中支持多种不同数据库的问题，这也是软件开发中比较常见的情形。其中的类 IFactory 相当于上图中的类 AbstractFactory; 类 IDepartment 相当于上图中的类 ProdcutA 。本题中只给出了一个产品类。

下面来分析程序。

第(1) 、(2) 空出现类 IDepartment 的定义中。类 IDepartment 的作用是为一类产品对象声明一个接口，在 C++中通常都采用抽象类来定义这种抽象操作接口。 C++中的抽 象类是包含了至少一个纯虚拟函数的类。纯虚拟函数的语法是： virtual()=0;

在程序中已经出现了纯虚拟函数的标志“=0”，因此(1) 、(2) 空应该都是纯虚拟 函数。下面来确定纯虚拟函数的原型。这需要去考察类 IDepartment 的子类，因此纯虚拟函数是在父类中定义，在子类中实现。由类图 5-1 可知，类 IDepartment 的子类分别是 SqlserverDepartment 和 AccessDepartment 。至此可以提前确定(3) 、(4) 空的内容了，即其所对应的父类。因此(3) 、(4) 空都应该填写 publicIDepartment 。我们看类 SqlserverDepartment 中的方法, 分别为:Insert 和 GetDepartment ，而在类 AccessDepartment 中也出现了这两个方法，且接口完全一致。所以这两个方法就应该是类 IDepartmeiit 中所定义的抽象接口。由此可知，(1) 空应该填写 virtualvoidInsert(Department* department)，(2) 空应该填写 virtualDepartmentGetDepartment(intid)。

空(5) 和(6) 分别缺失在类的名称以及该类中的方法。由图 5-1 和代码可知，缺少类 IFactory 的定义，所以(5) 空处应该填写 classIFactory 。那么类〖Factory 应包含的方法是什么？类 IFactory 的作用是声明一个创建抽象产品对象的操作接口，这个接口一定会同时出现在 IFactory 的子类 SqlServerFactory 和 AccessFactory 中，即 CreateDepartment 。(6) 处同样应该是一个纯虚拟函数，所以(6) 空处应该填写 virtual IDepartment* CreateDepartment() 。



试题六 答案：　解析： 1.void Insert(Department department)

2.Department GetDepartment(int id)

3.implements IDepartment

4.implements IDepartment

5.interface IFactory

6.IDepartmentCreateDepartment()

本题考查抽象工厂(AbstractFactory)模式的概念及应用。

AbstractFactory 模式的意图是，提供一个创建一系列相关或相互依赖对象的接口，而无需指定它们具体的类。 AbstractFactor 模式的结构如下图所示。

其中，类 AbstractFactory 声明一个创建抽象产品对象的操作接口；类 ConcreteFactory 实现创建具体产品对象的操作；类 ProductA 为一类产品对象声明一个接口；类 ConcreteProduct 具有 2 个功能：定义一个将被相应的具体工厂创建的产品对象；实现 ProductA 接口；类 Client 仅使用由 AbstractFactory 和 AbstractProduct 类声明的接口。

在以下情况可以使用 AbstractFactory 模式：

(1) 一个系统要独立于它的产品的创建、组合和表示时；

(2) 一个系统要由多个产品系统中的一个来配置时；

(3) 当要强调一系列相关的产品对象的设计一边进行联合使用时；

(4) 提供一个产品类库，而只想显示它们的接口而不是实现时。

题目利用抽象工厂模式来解决在同一个软件系统中支持多种不同数据库的问题，这也是软件开发中比较常见的情形。其中的类 IFactory 相当于上图中的类 AbstractFactory；类 IDepartment 相当于上图中的类 ProdcutA 。本题中只给出了一个产品类。

下面来分析程序。

第(1)、(2)空出现类 IDepartment 的定义中。类 IDepartment 的作用是为一类产品对象声明一个接口，在 Java 中通常都采用 Inteface 来定义这种抽象操作接口。要确定接口的原型，需要去考察实现接口 IDepartment 的类。由类图 6-1 可知，实现接口 IDepartment 的类分别是 SqlserverDepartment 和 AccessDepartment 。至此可以提前确定(3)、(4)空的内容了。(3)、(4)空都应该填写 implementsIDepartment 。我们看类 SqlserverDepartment 中的方法，分别为： Insert 和 GetDepartment ，而在类 AccessDepartment 中也出现了这两个方法，且接口完全一致。所以这两个方法就应该是 IDepartment 中所定义的抽象接口。由此可知，(1)空应该填写 voidInsert(Department department)，(2)空应该填写 DepartmentGetDepartment(intid)。

空(5)和(6)分别缺失在类的名称以及该类中的方法。由图 6-1 和代码可知，缺少类 IFactory 的定义。由代码 “classSqlServerFactoryimplementsIFactory” 和 “class AccessFactoryimplementsIFactory” 可知， IFactory 也是一个接口。所以(5)空处应该填写 InterfaceIFactory 。那么类 IFactory 应包含的方法是什么？类 IFactory 的作用是声明一个创建抽象产品对象的操作接口，这个接口一定会同时出现在类 SqlServerFactory 和 AccessFactory 中，即 CreateDepartment 。(6)处同样应该是一个纯虚拟函数，所以(6)空处应该填写 IDepartmentCreateDepartment() 。