

【软考达人】

# 软考资料免费获取

- 1、最新软考题库
- 2、软考备考资料
- 3、考前压轴题



**微信扫一扫，立马获取**



**6W+ 免费题库**



**免费备考资料**

PC版题库: [ruankaodaren.com](http://ruankaodaren.com)

# 全国计算机技术与软件专业技术资格（水平）考试

## 2012 年下半年 软件设计师 下午试卷

（考试时间 14:00～16:30 共 150 分钟）

请按下述要求正确填写答题纸

- 1.在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。
- 2.在答题纸的指定位置填写准考证号、出生年月日和姓名。
- 3.答题纸上除填写上述内容外只能写解答。
- 4.本试卷共 6 道题，试题一至试题四是必答题，试题五至试题六选答 1 道。每题 15 分，满分 75 分。
- 5.解答时字迹务必清楚，字迹不清时，将不评分。
- 6.仿照下面例题，将解答写在答题纸的对应栏内。

### 例题

2012 年下半年全国计算机技术与软件专业技术资格（水平）考试日期是（1）月（2）日。

因为正确的解答是“11 月 4 日”，故在答题纸的对应栏内写上“11”和“4”（参看下表）。

| 例题  | 解答栏 |
|-----|-----|
| （1） | 11  |
| （2） | 4   |

试题一至试题四是必答题

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 Crystal Reports 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

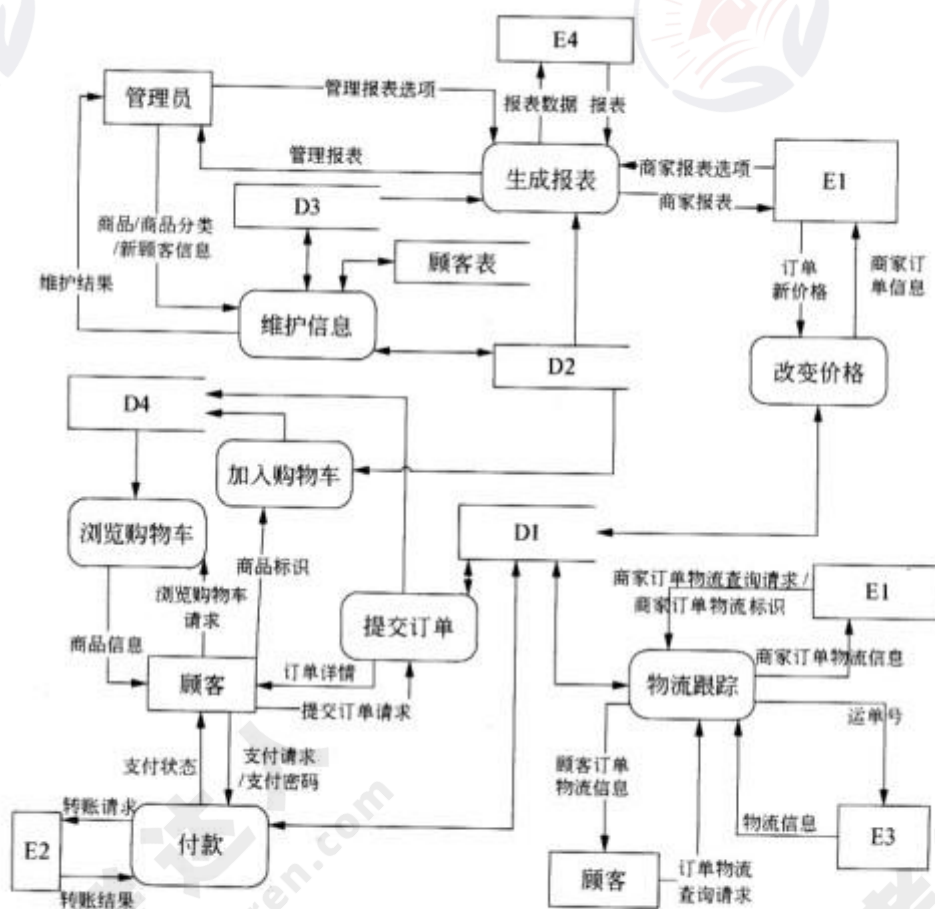


图 1-2 0 层数据流图

### 【问题 1】

使用说明中的词语，给出图 1-1 中的实体 E1~E4 的名称。

### 【问题 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 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 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：

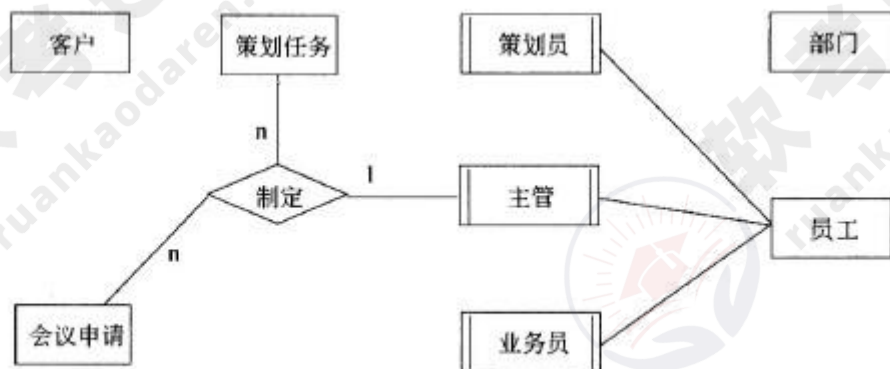


图 2-1 实体联系图

### 【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）

### 【问题 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】

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 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   |

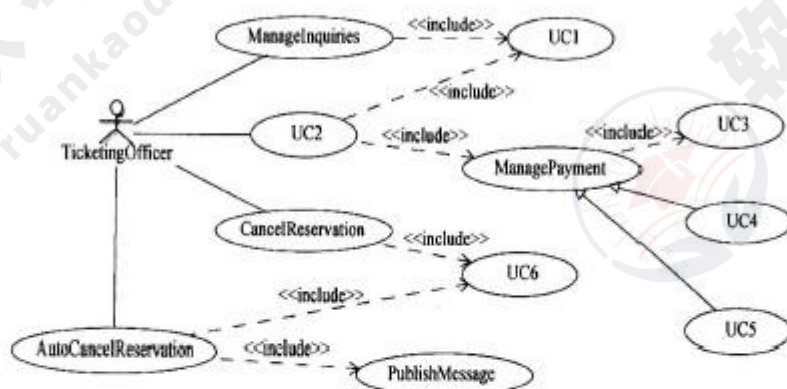


图 3-1 用例图

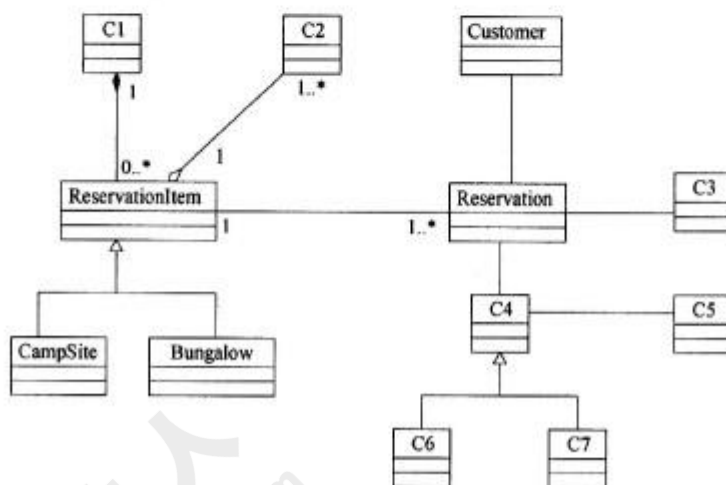


图 3-2 类图

### 【问题 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 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】**

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  $\{s_1, s_2, \dots, s_n\}$ ，且有  $s_i < C (1 \leq i \leq n)$ 。为节省运输成本，用尽可能少的集装箱来装运这  $n$  个货物。

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

$\text{temp}$ : 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;
}

```

```

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 1】

根据【说明】和【C 代码】，填充 C 代码中的空 (1)～(4)。

#### 【问题 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 O 符号表示)。

#### 【问题 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）

从下列的 2 道试题（试题五至试题六）中任选 1 道解答。  
如果解答的试题数超过 1 道，则题号小的 1 道解答有效。

### 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 SQL Server 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。

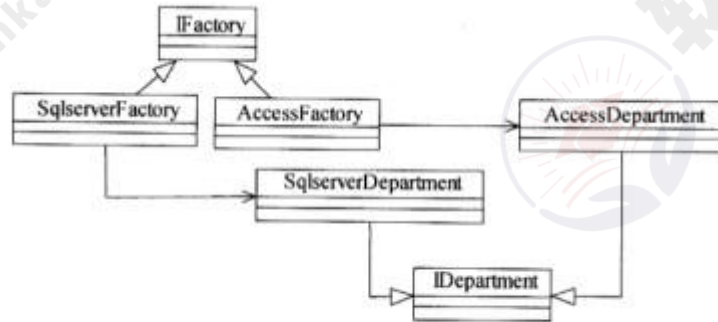


图 6-1 类图

### 【问题 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 SQL Server 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。

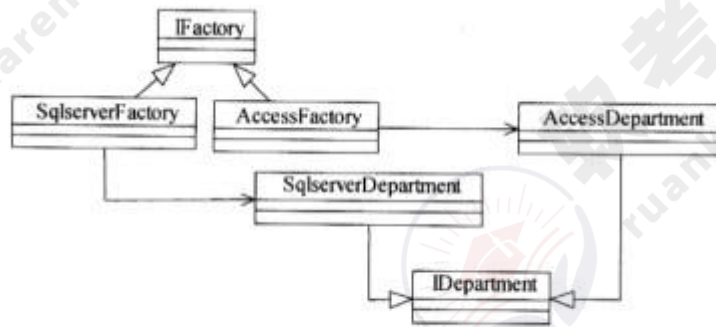


图 6-1 类图

## 【问题 1】

```

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  
}
```