

# 全国计算机技术与软件专业技术资格（水平）考试

## 2018 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

2018 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

### 【说明】

某房产中介连锁企业欲开发一个基于 Web 的房屋中介信息系统，以有效管理房源和客户，提高成交率。该系统的主要功能是：

1.房源采集与管理。系统自动采集外部网站的潜在房源信息，保存为潜在房源。由经纪人联系确认的潜在房源变为房源，并添加出售/出租房源的客户。由经纪人或客户登记的出售/出租房源，系统将其保存为房源。房源信息包括基本情况、配套设施、交易类型、委托方式、业主等。经纪人可以对房源进行更新等管理操作。

2.客户管理。求租/求购客户进行注册、更新，推送客户需求给经纪人，或由经纪人对求租/求购客户进行登记、更新。客户信息包括身份证号、姓名、手机号、需求情况、委托方式等。

3.房源推荐。根据客户的需求情况（求购/求租需求情况以及出售/出租房源信息），向已登录的客户推荐房源。

4.交易管理。经纪人对租售客户双方进行交易信息管理，包括订单提交和取消，设置收取中介费比例。财务人员收取中介费之后，表示该订单已完成，系统更新订单状态和房源状态，向客户和经纪人发送交易反馈。

5.信息查询。客户根据自身查询需求查询房屋供需信息。

现采用结构化方法对房屋中介信息系统进行分析与设计，获得如图 1-1 所示的上下文数据流图和图 1-2 所示的 0 层数据流图。

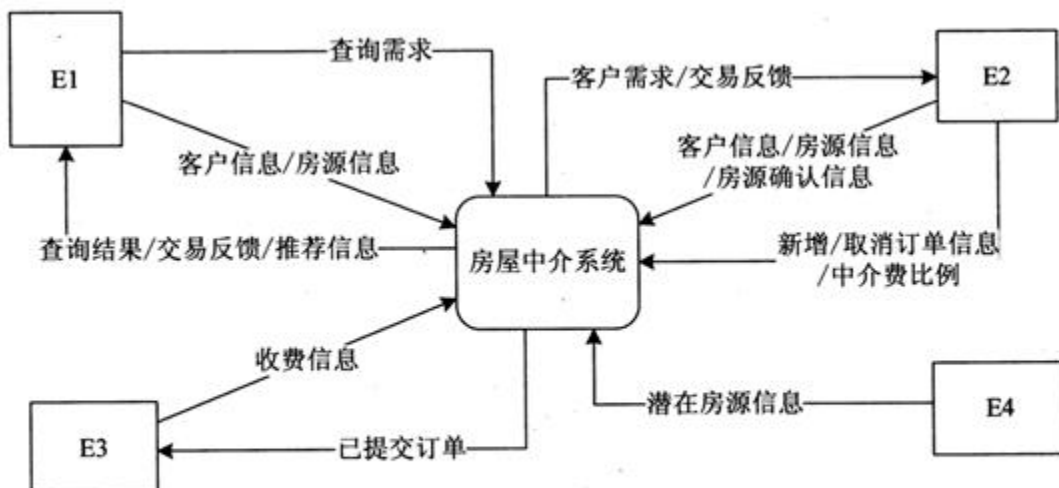


图 1-1 上下文数据流图

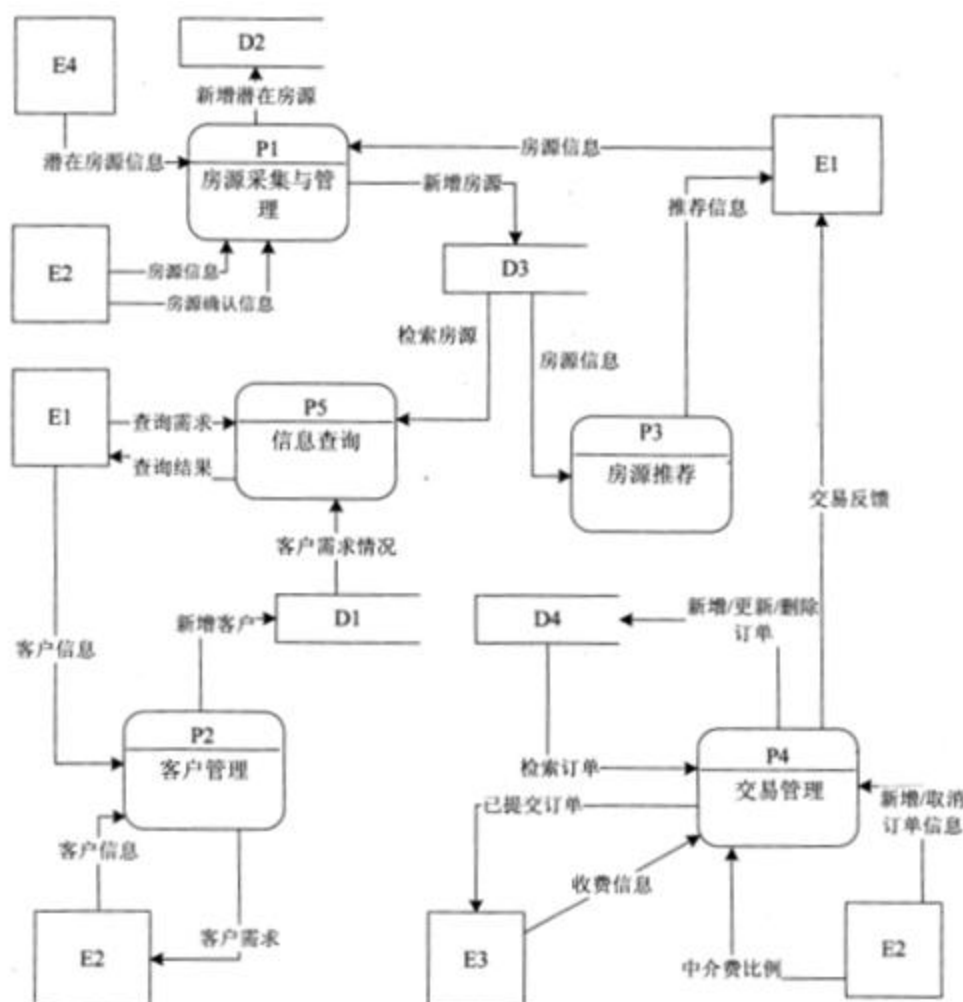


图 1-20 层数据流图

【问题 1】(4 分)

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】（4 分）**

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 分）**

根据说明和图中术语，补充图 1-2 中缺失的数据流及其起点和终点。

**【问题 4】（4 分）**

根据说明中术语，给出图 1-1 中数据流"客户信息"、"房源信息"的组成。

## 试题二

### 【说明】

某集团公司拥有多个分公司，为了方便集团公司对分公司各项业务活动进行有效管理，集团公司决定构建一个信息系统以满足公司的业务管理需求。

### 【需求分析】

1.分公司关系需要记录的信息包括分公司编号、名称、经理、联系地址和电话。分公司编号唯一标识分公司信息中的每一个元组。每个分公司只有一名经理，负责该分公司的管理工作。每个分公司设立仅为本分公司服务的多个业务部门，如研发部、财务部、采购部、销售部等。

2.部门关系需要记录的信息包括部门号、部门名称、主管号、电话和分公司编号。部门号唯一标识部门信息中的每一个元组。每个部门只有一名主管，负责部门的管理工作。每个部门有多名员工，每名员工只能隶属于一个部门。

3.员工关系需要记录的信息包括员工号、姓名、隶属部门、岗位、电话和基本工资。其中，员工号唯一标识员工信息中的每一个元组。岗位包括:经理、主管、研发员、业务员等。

### 【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）如图 2-1 所示：

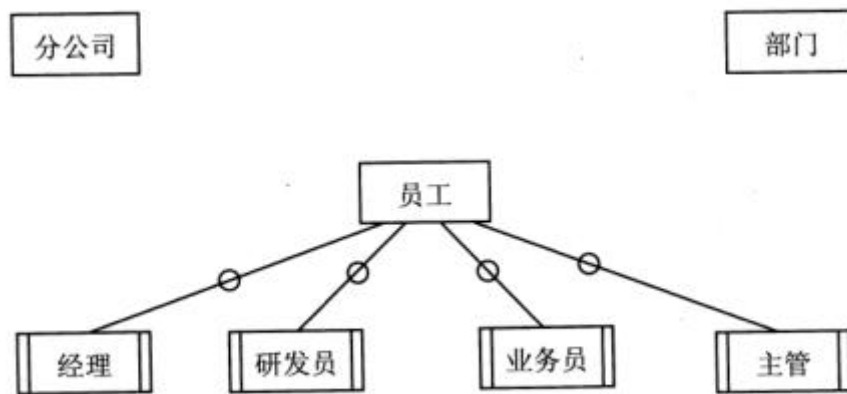


图 2-1 实体联系图

### 【关系模式设计】

分公司（分公司编号，名称，(a)，联系地址，电话）

部门（部门号，部门名称，(b)，电话）

员工（员工号，姓名 (c)，电话，基本工资）

### 【问题 1】（4 分）

根据问题描述，补充 4 个联系，完善图 2-1 的实体联系图。联系名可用联系 1、联系 2、联系 3 和联系 4 代替，联系的类型为 1:1、1:n 和 m:n （或 1:1、1:\*和\*:\*）。

**【问题 2】（5 分）**

根据题意，将关系模式中的空 （a） - （c） 补充完整。

**【问题 3】（4 分）**

给出"部门"和"员工"关系模式的主键和外键。

**【问题 4】（2 分）**

假设集团公司要求系统能记录部门历任主管的任职时间和任职年限，那么是否需要在数据库设计时增设一个实体?为什么?

本文档由微信号:ruankaopass，一手整理，通过他人购买的，拒绝售后。本人专业提供软考历年真题

试题三

【说明】

社交网络平台（SNS）的主要功能之一是建立在线群组，群组中的成员之间可以互相分享或挖掘兴趣和活动。每个群组包含标题、管理员以及成员列表等信息。

社交网络平台的用户可以自行选择加入某个群组。每个群组拥有一个主页，群组内的所有成员都可以查看主页上的内容。如果在群组的主页上发布或更新了信息，群组中的成员会自动接收到发布或更新后的信息。

用户可以加入一个群组也可以退出这个群组。用户退出群组后，不会再接收到该群组发布或更新的任何信息。

现采用面向对象方法对上述需求进行分析与设计，得到如表 3-1 所示的类列表和如图 3-1 所示的类图。

表 3-1 类列表

类名	描述
<b>SNSSubject</b>	群组主页的内容
<b>SNSGroup</b>	社交网络平台中的群组（在主页上发布信息）
<b>SNSObserver</b>	群组主页内容的关注者
<b>SNSUser</b>	社交网络平台用户/群组成员
<b>SNSAdmin</b>	群组的管理员

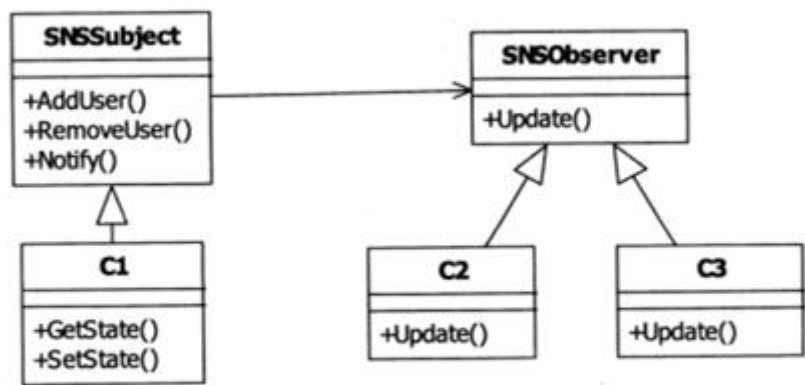


图 3-1 类图

【问题 1】（6 分）

根据说明中的描述，给出图 3-1 中 C1 C3 所对应的类名。

**【问题 2】（6 分）**

图 3-1 中采用了哪一种设计模式?说明该模式的意图及其适用场合。

**【问题 3】（3 分）**

现在对上述社交网络平台提出了新的需求:一个群体可以作为另外一个群体中的成员，例如群体 A 加入群体 B 。那么，群体 A 中的所有成员就自动成为群体 B 中的成员。

若要实现这个新需求，需要对图 3-1 进行哪些修改?（以文字方式描述）



#### 试题四

##### 【说明】

给定一个字符序列  $B=b_1b_2\dots b_n$ , 其中  $b_i \in \{A, C, G, U\}$ 。B 上的二级结构是一组字符对集合  $S=\{(b_i, b_j)\}$ , 其中  $i, j \in \{1, 2, \dots, n\}$ , 并满足以下四个条件:

- (1) S 中的每对字符是 (A,U), (U,A), (C,G) 和 (G,C) 四种组合之一;
- (2) S 中的每对字符之间至少有四个字符将其隔开, 即  $i < j - 4$ ;
- (3) S 中每一个字符 (记为  $b_k$ ) 的配对存在两种情况:  $b_k$  不参与任何配对;  $b_k$  和字符  $b_t$  配对, 其中  $t < k - 4$ ;
- (4) (不交叉原则) 若  $(b_i, b_j)$  和  $(b_k, b_l)$  是 S 中的两个字符对, 且  $i < k$ , 则  $i < k < j < l$  不成立。

B 的具有最大可能字符对数的二级结构 S 被称为最优配对方案, 求解最优配对方案中的字符对数的方法如下:

假设用  $C(i, j)$  表示字符序列  $b_i b_{i+1} \dots b_j$  的最优配对方案 (即二级结构 S) 中的字符对数, 则  $C(i, j)$  可以递归定义为:

$$C(i, j) = \begin{cases} \max(C(i, j-1), \max(C(i, t-1) + 1 + C(t+1, j-1))) & \text{若 } b_t \text{ 和 } b_j \text{ 匹配且 } i < j-4 \\ 0 & \text{否则} \end{cases}$$

下面代码是算法的 C 语言实现, 其中

n: 字符序列长度  
B[]: 字符序列  
C[][]: 最优配对数量数组

##### 【C 代码】

```
#include <stdio.h>
#include <stdlib.h>
#define LEN 100

/*判断两个字符是否配对*/
int isMatch(char a, char b){
    if((a == 'A' && b == 'U') || (a == 'U' && b == 'A'))
        return 1;
    if((a == 'C' && b == 'G') || (a == 'G' && b == 'C'))
        return 1;
    return 0;
}
```

```

/*求最大配对数*/
int RNA_2(char B[LEN], int n){
    int i, j, k, t;
    int max;
    int C[LEN][LEN] = {0};

    for(k = 5; k <= n - 1; k++){
        for(i = 1; i <= n - k; i++){
            j = i + k;
            _____(1)_____ ;
            for(_____(2)_____; t <= j - 4; t++){
                if(_____(3)_____ && max < C[i][t - 1] + 1 + C[t + 1][j - 1])
                    max = C[i][t - 1] + 1 + C[t + 1][j - 1];
            }
            C[i][j] = max;
            printf("c[%d][%d] = %d--", i, j, C[i][j]);
        }
    }
    return _____(4)_____ ;
}

```

**【问题 1】（8 分）**

根据题干说明，填充 C 代码中的空（1） - （4）。

**【问题 2】（4 分）**

根据题干说明和 C 代码，算法采用的设计策略为（5）

算法的时间复杂度为（6），（用 O 表示）。

**【问题 3】（3 分）**

给定字符序列 ACCGGUAGU，根据上述算法求得最大字符对数为（7）

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

阅读下列说明和 C++代码，将应填入（n）处的字句写在答题纸的对应栏内。

#### 【说明】

某航空公司的会员积分系统将其会员划分为:普卡（Basic）、银卡（Silver）和金卡（Gold）三个等级。非会员（NonMember）可以申请成为普卡会员。会员的等级根据其一年内累积的里程数进行调整。描述会员等级调整的状态图如图 5-1 所示。现采用状态（State）模式实现上述场景，得到如图 5-2 所示的类图。

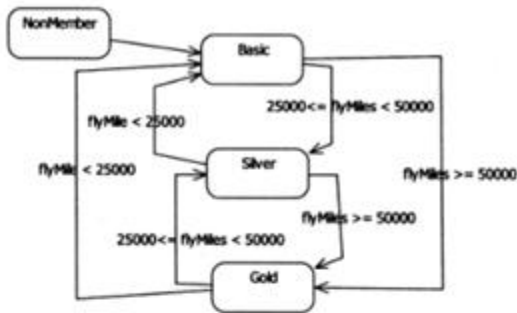


图 5-1 会员等级调整状态图

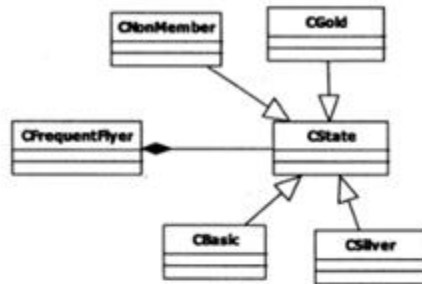


图 5-2 状态模式类图

#### 【C++代码】

```
#include <iostream>
using namespace std;
class FrequentFlyer; class CBasic; class CSilver; class CGold; class CNoCustomer; // 提前引用
class CState {
private: int flyMiles; // 里程数
public:
    _____; // 根据累积里程数调整会员等级
};
class FrequentFlyer {
friend class CBasic; friend class CSilver; friend class CGold;
private:
    CState *state; CState *nocustomer; CState *basic; CState *silver; CState *gold;
    double flyMiles;
public:
    FrequentFlyer(){ flyMiles = 0; setState(nocustomer); }
    void setState(CState *state){ this->state = state; }
    void travel(int miles) {
```

```

        double bonusMiles = state->travel(miles, this);
        flyMiles = flyMiles + bonusMiles;
    }
};

class CNoCustomer : public CState {    // 非会员
public:
    double travel(int miles, FrequentFlyer* context) {    // 不累积里程数
        cout << "Your travel will not account for points\n";    return miles;
    }
};

class CBasic : public CState {    // 普卡会员
public:
    double travel(int miles, FrequentFlyer* context) {
        if(context->flyMiles >= 25000 && context->flyMiles < 50000)
            _____ (2) _____;
        if(context->flyMiles < 25000) _____ (3) _____;
        return miles + 0.5*miles;    // 累积里程数
    }
};

class CGold : public CState {    // 金卡会员
public:
    double travel(int miles, FrequentFlyer* context) {
        if(context->flyMiles >= 25000 && context->flyMiles < 50000)
            _____ (4) _____;
        if(context->flyMiles < 25000) _____ (5) _____;
        return miles + 0.5*miles;    // 累积里程数
    }
};

class CSilver : public CState {    // 银卡会员
public:
    double travel(int miles, FrequentFlyer* context) {
        if(context->flyMiles < 25000)
            context->setState(context->basic);
        if(context->flyMiles >= 50000)
            context->setState(context->gold);
        return (miles + 0.25*miles);
    }
};

```

## 试题六

阅读下列说明和 Java 代码，将应填入 (n) 处的字句写在答题纸的对应栏内。

### 【说明】

某航空公司的会员积分系统将其会员划分为:普卡 (Basic)、银卡 (Silver) 和金卡 (Gold) 三个等级。非会员 (NonMember) 可以申请成为普卡会员。会员的等级根据其一年内累积的里程数进行调整。描述会员等级调整的状态图如图 6-1 所示。现采用状态 (State) 模式实现上述场景，得到如图 6-2 所示的类图。

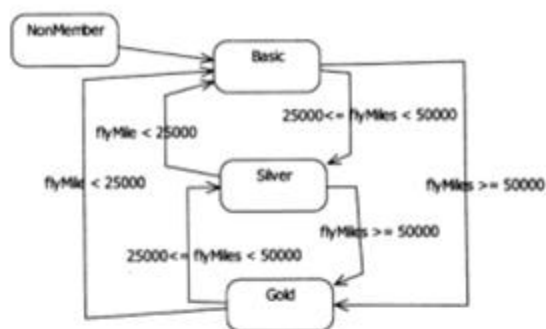


图 6-1 会员等级调整状态图

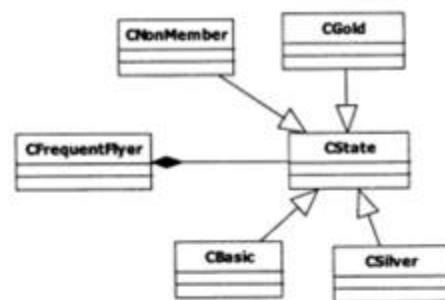


图 6-2 状态模式类图

### 【Java 代码】

```
import java.util.*;

abstract class CState {
    public int flyMiles;    // 里程数
    public _____ (1) _____; // 根据累积里程数调整会员等级
}

class CNoCustomer extends CState {    // 非会员
    public double travel(int miles, FrequentFlyer context) {
        System.out.println("Your travel will not account for points");
        return miles;    // 不累积里程数
    }
}

class CBasic extends CState {    // 普卡会员
    public double travel(int miles, FrequentFlyer context) {
        if(context.flyMiles >= 25000 && context.flyMiles < 50000)
            _____ (2) _____;
        if(context.flyMiles >= 50000)
            _____ (3) _____;
        return miles;
    }
}
```

```

    }
}

class CGold extends CState {    // 金卡会员
    public double travel(int miles, FrequentFlyer context) {
        if(context.flyMiles >= 25000 && context.flyMiles < 50000)
            (4);
        if(context.flyMiles < 25000)
            (5);
        return miles + 0.5*miles;    // 累积里程数
    }
}

class CSilver extends CState {    // 银卡会员
    public double travel(int miles, FrequentFlyer context) {
        if(context.flyMiles <= 25000)
            context.setState(new CBasic());
        if(context.flyMiles >= 50000)
            context.setState(new CGold());
        return (miles + 0.25*miles);    // 累积里程数
    }
}

class FrequentFlyer {
    CState state;
    double flyMiles;
    public FrequentFlyer(){
        state = new CNoCustomer();
        flyMiles = 0;
        setState(state);
    }
    public void setState(CState state){    this.state = state;    }
    public void travel(int miles) {
        double bonusMiles = state.travel(miles, this);
        flyMiles = flyMiles + bonusMiles;
    }
}

```