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 2018 年 下半年 下午试卷 案例

（考试时间 150 分钟）

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。

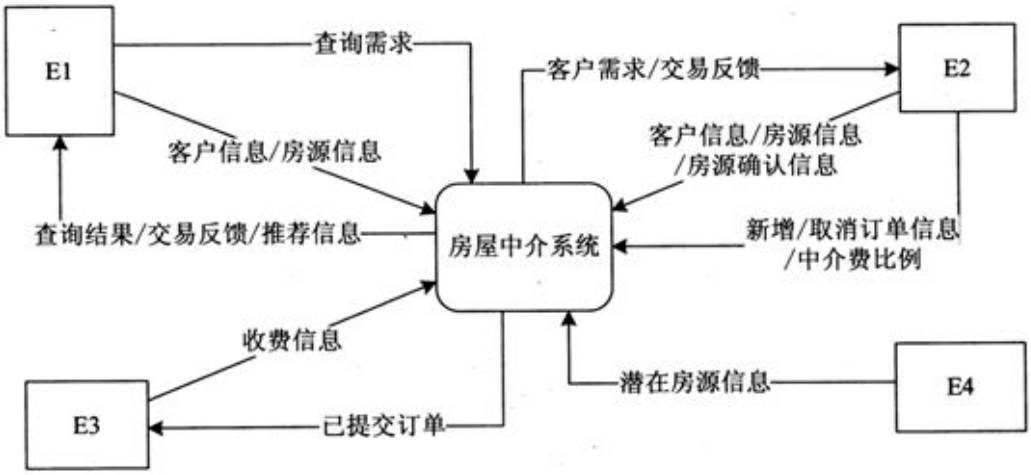

图 1-1 上下文数据流图

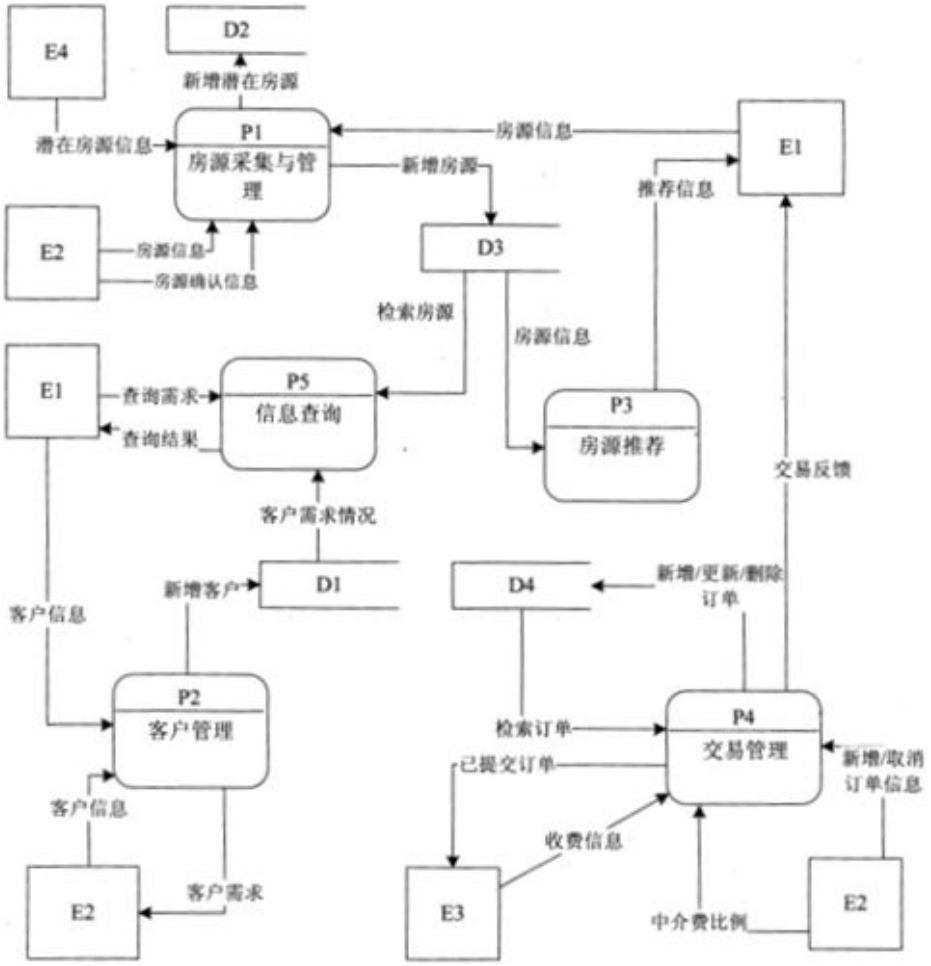

图 1-2 0 层数据流图

问题：1.1
(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：1.2
(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：1.3
(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：1.4
(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:

【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)

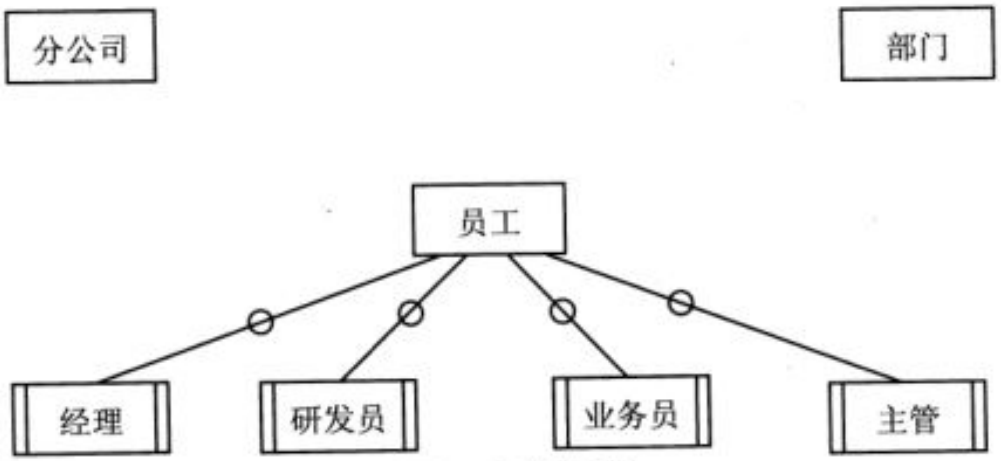

图 2-1 实体联系图

问题： 2.1
(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： 2.2
(5 分)
根据题意，将关系模式中的空（a)-(c）补充完整。
问题： 2.3
(4 分)
给出"部门"和"员工"关系模式的主键和外键。
问题： 2.4
(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?

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

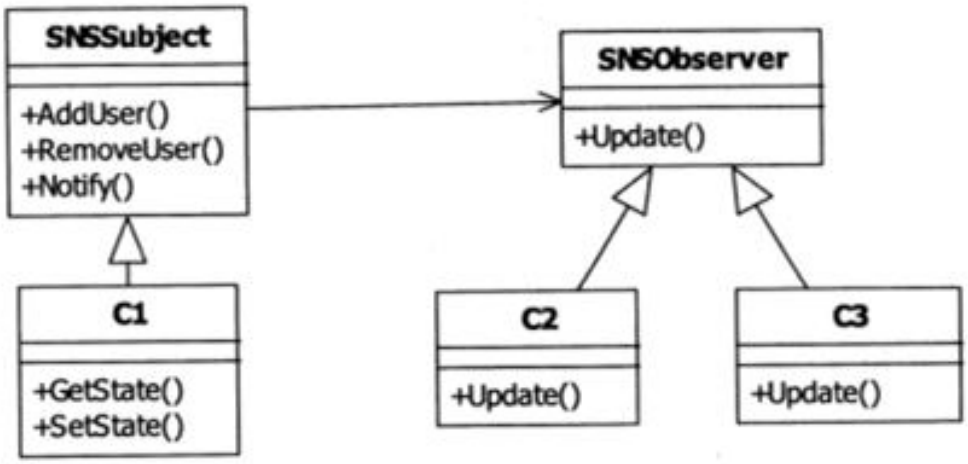

图 3-1　类图

问题：3.1
(6 分)
根据说明中的描述，给出图 3-1 中 C1C3 所对应的类名。
问题：3.2
(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
(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=$b_1b_2\ldots b_n$，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$i, j \in \{1,2,\ldots,n\}$，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 $b_ib_{i+1}\ldots b_j$ 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：

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

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：4.1

(8 分)

根据题干说明，填充 C 代码中的空(1)-(4)。

问题：4.2

(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：4.3

(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)

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。

问题：5.1

【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。

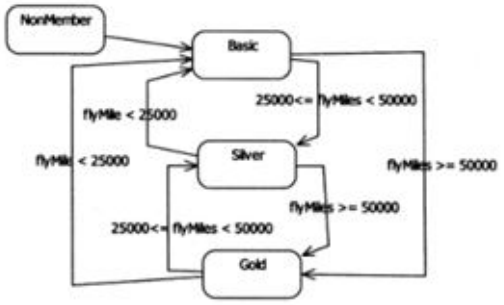

图 5-1 会员等级调整状态图

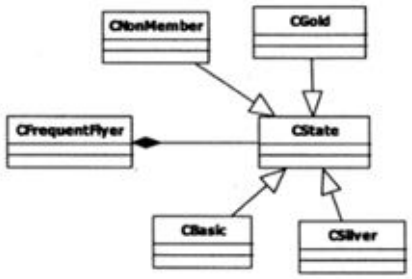

图 5-2 状态模式类图

【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:
        ____（1）____;      // 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。

问题：6.1

【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。

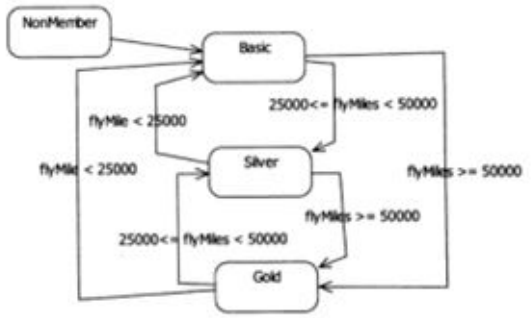

图 6-1 会员等级调整状态图

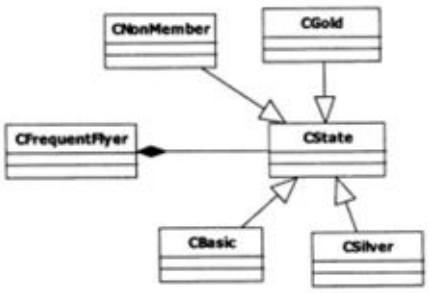

图 6-2 状态模式类图

【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;
    }
}
```

试题一 答案：　解析： E1：客户；E2：经纪人；E3：财务人员；E4：外部网站

题干说明中，自动采集潜在房源信息，并且无反馈信息的为外界网站，即 1-1 中的 E4；

只有经纪人可以确认潜在房源，因此 E2 为经纪人；

系统只向客户推送推荐房源，因此 E1 为客户。

财务管理人员收取中介费用，因此 E3 为财务人员。

D1 ： 客户记录； D2 ： 潜在房源记录； D3 ： 房源记录； D4 ： 订单记录

对于新增潜在房源，保存为潜在房源，即 D2 为潜在房源记录；

对于新增房源保存为房源，即 D3 为房源记录；

对于新增客户应该保存为客户信息，因此 D1 为客户记录或客户信息表；

对于订单检索的对象应该为订单记录，即 D4 为订单记录。

缺失数据流如下：

1 、交易反馈：起点-P4 交易管理，终点-E2

2 、客户需求：起点-D1 ，终点-P3 房源推荐

3 、房源状态：起点-P4 交易管理，终点-D3

客户信息：身份证号，姓名，手机号，需求情况，委托方式。

房源信息：基本情况，配套设施，交易类型，委托方式，业主等。

试题二 答案： 解析： 联系 1 ：分公司：经理， 1 ： 1

联系 2 ：分公司：部门， 1 ：*

联系 3 ：部门：主管， 1 ： 1

联系 4 ：部门：员工， 1 ：*

(a)经理工号

(b)分公司编号，主管号

(c)隶属部门，岗位

部门

主键：部门号；外键：分公司编号，主管号

员工

主键：员工号；外键：隶属部门

不需要增加新的实体，对于任职情况，可以将部门与主管的联系单独形成关系模式，联系(部门号，主管工号，任职时间，任职年限)，考虑到同一个员工可能会在不同的时间多次担任主管，因此，对于该关系模式，可以将(部门号，主管工号，任职时间)作为组合主键。

试题三 答案： 解析： C1 ： SNSGroup ； C2 ： SNSUser ； C3 ： SNSAdmin 。

(其中 C2 、 C3 可以互换)

采用的设计模式：观察者模式

意图：当被观察者(群组主页)发生改变时，可以通知所有的观察者(群组主页内容的关注者)随之改变，以达到联动的效果。

使用场合：观察者模式是行为型模式，定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并自动更新。

新增一个被观察者对象群组 B 的主页，对于观察者，新增一个方法，加入群组 B，加入之后，可以接收被观察者群组 B 的主页变动所发送的通知。

试题四 答案：　解析：　(1) max=C[i][j-1]

(2) t=i

(3) isMatch(b[t],b[j])，或 isMatch(B[t],B[j])==1，或与其等价的形式

(4)max 或 C[i-1][j]

采用的算法策略：动态规划法

时间复杂度：O(nsup3)

最大字符对数：2

试题五 答案：　解析：　(1) virtual double travel(int miles,FrequentFlyer* context)=0

(2) context->setState(context->silver)

(3) context->setState(context->gold)

(4) context->setState(context-> silver)

(5) context->setState(context->basic)

试题六 答案：　解析：　(1) abstract double travel(int miles,FrequentFlyer context)

(2) context.setState(new CSilver())

(3) context.setState(new CGold ())

(4) context.setState(new CSilver())

(5) context.setState(new CBasic())