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 2021 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 试题一

阅读下列说明和图回答问题 1 至问题 4，将解答填入答题纸的对应栏内。

【说明】某停车场运营方为了降低运营成本，减员增效，提供良好的停车体验，欲开发无人值守停车系统，该系统的主要功能是：

1、 信息维护。管理人员对车位(总数、空余车位数等)计费规则等基础信息进行设置。

2、 会员注册。车主提供手机号、车牌号等信息进行注册，提交充值信息(等级、绑定并授权支付系统进行充值或交费的支付账号)不同级别和充值额度享受不同停车折扣点。

3、 车牌识别。当车辆进入停车场时，若有(空余车位数大于 1)，自动识别车牌号后进行道闸控制，当车主开车离开停车场时，识别车牌号，计费成功后，请求道闸控制。

4、 计费。更新车辆离场时间，根据计费规则计算出停车费用，若车主是会员，提示停车费用：若储存余额够本次停车费用，自动扣费，更新余额，若储值余额不足，自动使用授权缴费账号请求支付系统进行支付，获取支付状态。若非会员临时停车，提示停车费用，车主通过扫描费用信息中的支付码调用支付系统自助交费，获取支付状态。

5、 道闸控制。根据道闸控制请求向道闸控制系统发送时干发行指令和接收道闸执行状态。若道闸执行状态为正常放行时，对入场车辆，将车牌号及其入场时间信息存入停车记录，修改空余车位数；对出厂车辆更新停车状态，修改空余车位数。当因道闸重置系统出现问题(断网断电或是故障为抬杠等情况)，而无法在规定的时间内接收到其返回的执行状态正常放行时，系统向管理人员发送异常告警信息，之后管理人员安排故障排查处理，确保车辆有序出入停车场。

现采用结构化方法对无人值守停车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。

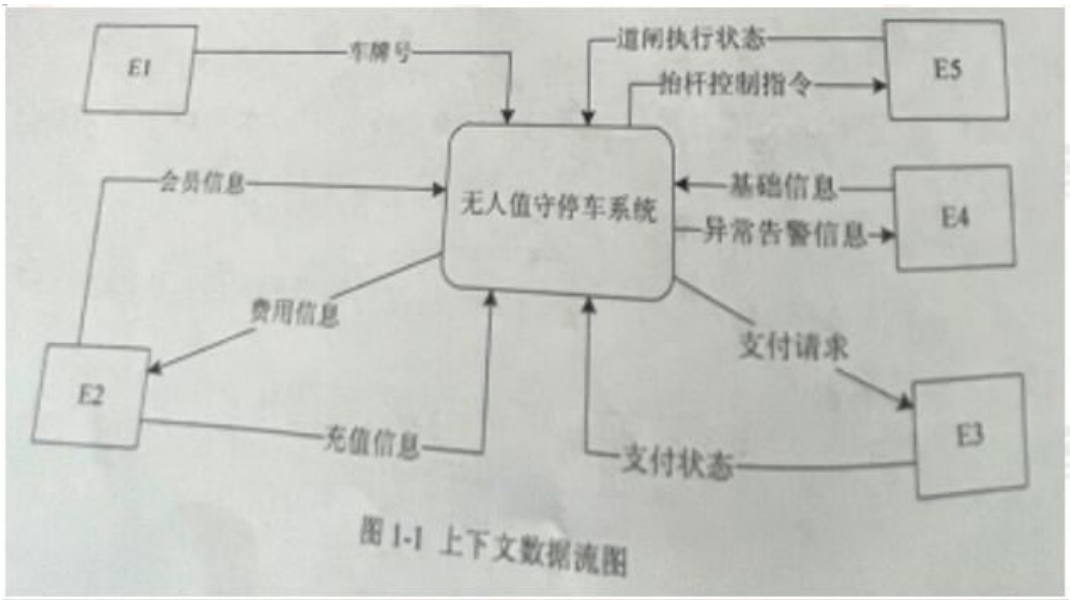

图 1-1 上下文数据流图

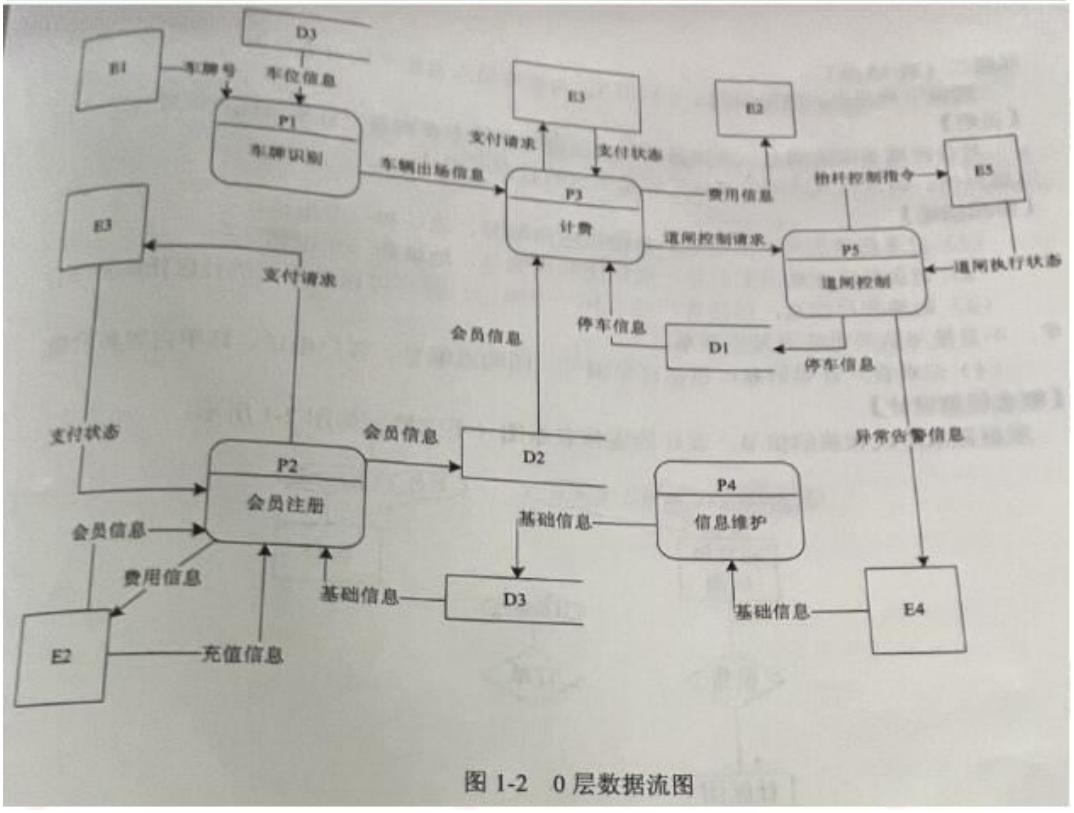

图 1-2 0 层数据流图

【问题 1 】(5 分)

使用说明中的词语，给出图 1-1 中的实体 E1~E5 的名称

【问题 2 】(3 分)

使用说明中的词语，给出图 1-2 中的数据存储 D1~D3 的名称

【问题 3 】(4 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 】(3 分)

根据说明，采用结构化语言对“道闸控制”的加工逻辑进行描述

试题二　试题二

阅读下列说明，回答问题 1 至问题 3 ，将解答填入答题纸的对应栏内。

【说明】

某社区蔬菜团购网站，为规范商品收发流程，便于查询客户订单情况，需要开发一个信息系统，请根据下列需求描述完成该系统的数据库设计。

【需求描述】

( ) 记录蔬菜供应商的信息，包括供应商编号，地址和一个电话

( ) 记录社区团购点的信息，包括团购点编号、地址和一个电话。

( ) 记录客户信息，包括包括客户姓名和一个电话，客户可以在不同的社区团购点订单，不直接与蔬菜供应商发生联系。

( ) 记录客户订单信息，包括订单编号、团购点编号、客户电话、订单内容和日期。

【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(不完整)如图 2-1 所示。

【逻辑结构设计】

根据概念模型设计阶段完成的实体联系图，得出如下关系模型(不完整)：

蔬菜供应商（供应商编号，地址，电话）

社区团购店点（团购点编号，地址，电话）

供货(供货商编号，(a))

客户(姓名，客户电话)

订单(订单编号，团购点编号，(b)，订单内容，日期)

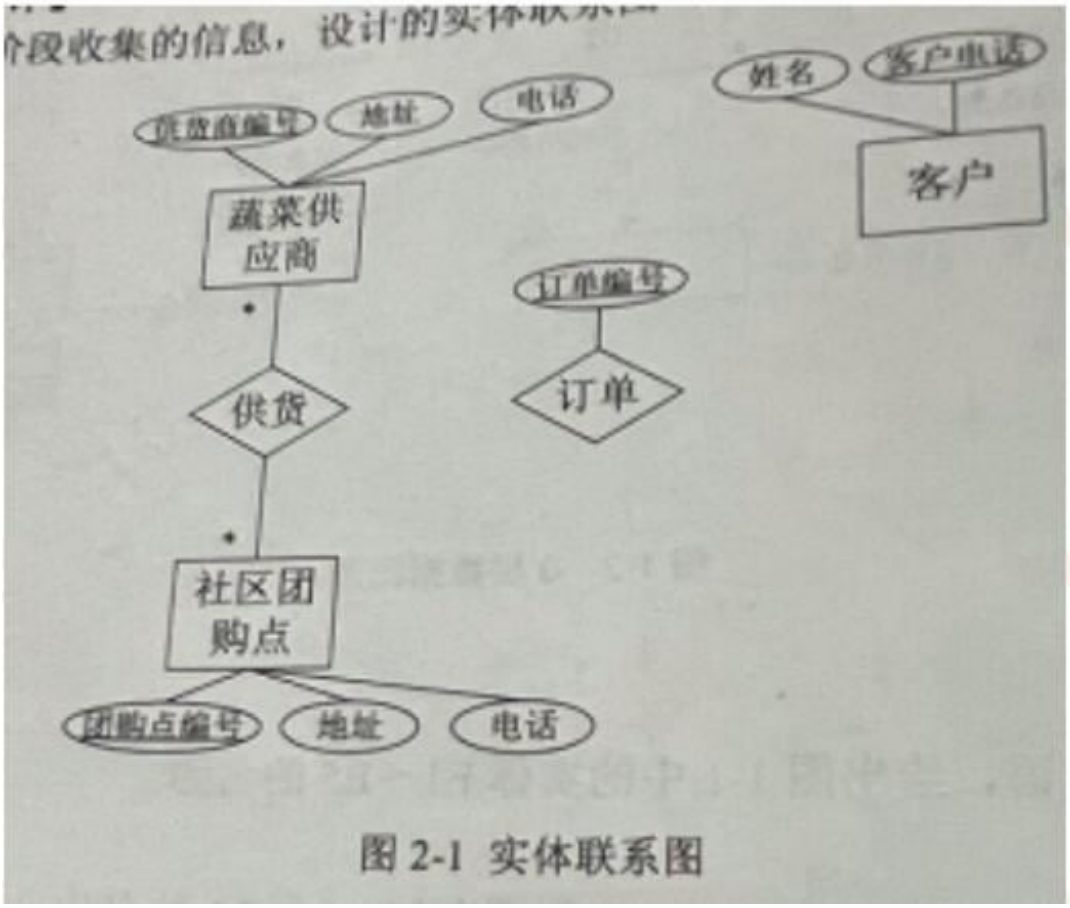

图 2-1 实体联系图

【问题 1 】(6 分)

根据问题描述，补充 2-1 的实体联系图

【问题 2 】(4 分)

补充逻辑结构设计结构中的(a)(b)两处空缺及完整性的约束关系

【问题 3 】(5 分)

若社区蔬菜团购网站还将有代收快递的业务，请增加新的“快递”实体，并给出客户实体和快递实体之间的“收取”联系，对图 2-1 进行补充，“快递”关系模式包括快递编号、客户电话和日期。

试题三　试题三

阅读下列说明和图，回答问题 1 至问题 3，将解答填入答题纸的对应栏内。

【说明】

某中医医院拟开发一套线上抓药 APP，允许患者凭借该医院医生开具的处方线上抓药，并提供免费送药上门服务。该系统的主要功能描述如下：

( ) 注册。患者扫描医院提供的二维码进行注册，注册过程中，患者需提供其病历号，系统根据病历号自动获取患者基本信息

( ) 登录。已注册的患者可以登录系统进行线上抓药，未注册的患者系统拒绝其登陆。

( ) 确认处方。患者登录后，可以查看医生开具的所有处方。患者选择需要抓药的处方和数量(需要抓几副药)，同时说明是否需要煎制。选择取药方式：自行到店取药或者送药上门，若选择送药上门，患者需要提供提供收贷人姓名、联系方式和收货地址。系统自动计算本次抓药的费用，患者可以使用微信或支付宝等支付方式支付费用。支付成功之后，处方被发送给药师进行药品配制。

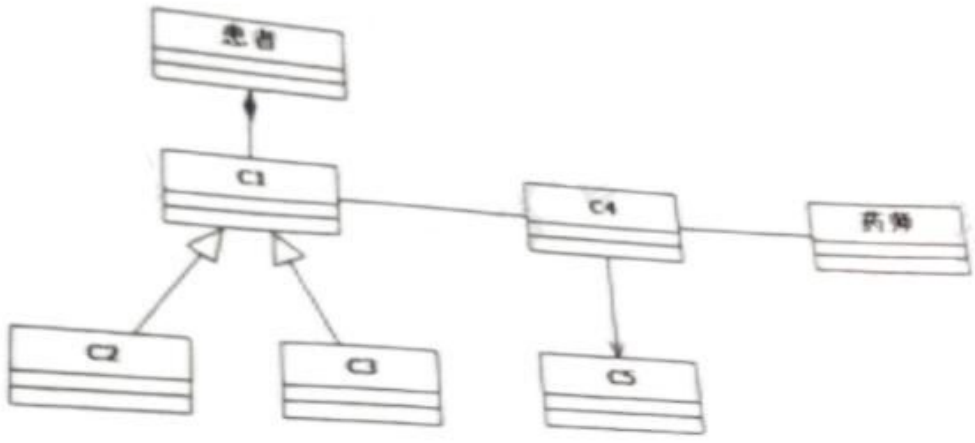

图 3-2　类图

【问题 1 】(7 分)

根据说明中的描述，给出图 3-1 中 A1~A3 所对应的参与者名称和 U1~U4 处所对应的用例名称。

【问题 2 】(5 分)

根据说明中的描述，给出图 3-2 中 C1~C5 所对应的类名。

【问题 3 】(3 分)

简要解释用例之间的 include 、 extend 和 generalize 关系的内涵。

试题四 试题四(共 15 分)

凸多边形是指多边形的任意两点的连线均落在多边形的边界或者内部。相邻的点连线落在多边形边上，称为边，不相邻的点连线落在多边形内部。称为弦。假设任意两点连线上均有权重，凸多边形最优三帮剂分问题定义为：求将凸多边形划分为不相交的三角形集合，且各三角形权重之和最小的剖分方案。每个三角形的权重为三条边权重之和。

假设 N 个点的凸多边形点编号为 V1,V2,……,VN,若在 VK 处将原凸多边形划分为一个三角形 V1VkVN ，两个子多边形 V1,V2,…,Vk 和 Vk,Vk+1,…VN ，得到一个最优的剖分方案，则该最优剖分方案应该包含这两个子凸边形的最优剖分方案。用 m[i][j]表示带你 Vi-1,Vi,…Vj 构成的凸多边形的最优剖分方案的权重，S[i][j]记录剖分该凸多边形的 k 值。

则

其中， w(V-1x)=W-1.k+W4./+Wj,.1-1 为三角形 V,-1VV,的权重， W1-1,k,Wk,j,Wj.1-1 分别

为该三角形三条边的权重。

求解凸多边形的最优剖分方案，即求解最小剖分的权重及对应的三角形集。

```
#include

#define N 6
```

```
/* 凸多边形规模 */

int m[N + 1] [N + 1];    /* m[i][j]表示多边形 Vi-1 到 Vj 最优三角剖分的
权值 */

int S[N + 1] [N + 1];    /* S[i][j]记录多边形 Vi-1 到 Vj 最优三角剖分的
k 值 */

int W[N + 1] [N + 1];    /* 凸多边形的权重矩阵，在 main 函数中输入 */

/*三角形的权重 a， b， c，三角形的顶点下标*/

int get_ triangle_weight( int a int b int c )
{
      return(W[a][b] + W[b][c] + W[c][a]);
}


/*求解最优值*/

void triangle_partition()
{
      int i, r, k, j;

      int temp;

  /*初始化*/
      for ( i = 1; i <= N; i++ )
      {
            m[i][i] = 0;
      }

  /*自底向上计算 m， S*/
      for ( r = 2; (1); r++ )
      {
    /*r 为子问题规模*/
    /* r<=N */
            for ( i = 1; k <= N - r + 1; i++ )
            {
                  (2);
      /* int j=i+r-1 */
                  m[i][j] = m[i][j] + m[i + 1][j] +
get_triangle_weight( i - 1, i, j );
```

```
      /*k=j*/
                  S[i][j] = i;

                  for ( k = j + 1; k

        temp = m[i][k] + m[k + 1][j] + ge_triangle_ weight( i - 1, k,
j );

        if ( (3) )
        { /*判断是否最小值*/ /* temp */
          m[i][j] = temp;

          S[i][j] = k;
        }
      }
    }
  }
}

/*输出剖分的三角形 i， j：凸多边形的起始点下标*/
void print_triangle( int i, int j )
{
  if ( i == j )
    return;

  print_triangle( i, S[i][j] );

  print_triangle( (4) ); /* s[i][j]+1,j */

  print( " V % d - -V % d --V % d \ n ", i - 1, S[i][j], j );
}
```

$$m[i][j]=\begin{cases}0 & i\geq j\\ \min\limits_{i\leq k<j}\{m[i][k]+m[k+1][j]+w(v_{i-1}v_kv_j)\} & i<j\end{cases}$$

【问题 1】(8 分)

根据说明和 C 代码，填充 C 代码中的空( )~( )

【问题 2 】(7 分)

根据说明和 C 代码，该算法采用的设计策略为( )

算法的时间复杂度为( )，空间复杂度为( )(用 0 表示)。

试题五 试题五(共 15 分)

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】

层叠买单是留口风格的软件系统中经常采用的一种系统功能组织方式。层叠菜单(如

到 5-1 示例)中包含的可能是一个菜单项(直接对应某个功能)，也可能是一个子菜单。现采用组合(Compasite)设计模式实现层叠菜单，得到如图 5-2 所示的类图。

【 C++代码】

```
# include <list>

# include <iostream>

# include <string>

using namespace std;

class MenuComponent{//构成层叠菜单的元素

( ) :

string name;  / /菜单项或子菜单名称

public:

void printMenu0 { cout < <name;}

( )
```

```
virtual void removeMenuElement (MenuComponent*element) =0;

( )

}

class Menultem:public MenuComponent {

public:

Menultem (string name) { this->name-name;}

void addMenuElement (MenuComponent*element) { retum;}

void removeMenuElement (MenuComponent*element) {retum;}

list <MenuComponent*> *getElemento { retum NULL; }

 }:

class Menu:public MenuComponent{

private:

( )

public:

Menu (string name) {this->name = name;}

void addMenuElement (MenuComponent*element) |elementList.push_back
{element) ; }

void removeMenuElement (MenuComponent*element)
{elementList.remove{element);}

list <MenuComponent*> *getElement0 { return&elementList;}

} ;

int main0( ) {

MenuComponent*mainMenu = new Menu ("Insert") ;

MenuComponent*subMenu=new Menu ("Chart") ;
```

```
MenuComponent*element=new Menultem ("On This Sheet") ;

( )

subMenu-> addMenuElement (element) ;

return 0;

}
```

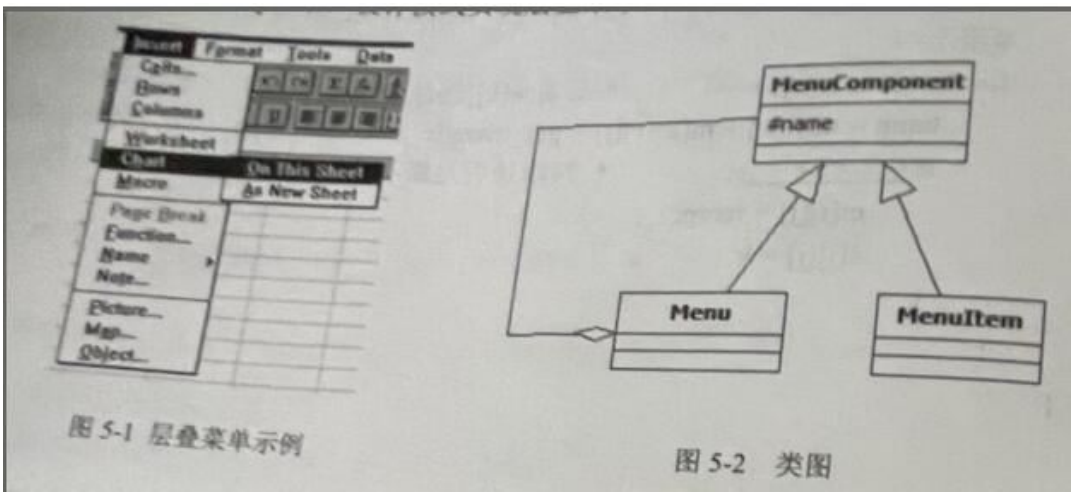

图 5-1 层叠菜单示例　　图 5-2　类图

试题六　试题六(共 15 分)

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】

层叠菜单是窗口风格的软件系统中经常采用的一种系统功能组织方式。层叠菜单(如图 6-1 示例)中包含的可能是一个菜单项(直接对应某个功能)，也可能是一个子菜单，现在采用组合(composite)设计模式实现层叠菜单，得到如图 6-2 所示的类图

【 Java 代码】

```
import java.util. *;



abstract class MenuComponent{  //构成层叠菜单的元素
```

```
( )Stringname;                    //菜单项或子菜单名称



public void printName( ) {  System.out.println(name);}

Public  ( )

public abstract boolean removeMenuElement (MenuComponent element) ;

Public  ( )

}

class Menultem extends MenuComponent

public Menultem (String name) { this.name=name;)

public boolean addMenuElement (MenuComponent element) { return false;}

public boolean removeMenuElement (MenuComponent element) { retum
false;}

public List <MenuComponentgetElement( ) { return null; }

}

class Mcnu extends MenuComponent {

Private  ( )

public Menu (String name) {

thi.name-name;

this.elementList-new ArmayList <MenuComponent O.

}

public boolean addMenuElement (MenuComponent element) {

retum elementList.add (element) ;
```

```
}
public boolean removeMenuElement (MenuComponent element) (
retum elementL.ist.remove (element) ;
}
public List <MenuComponent> getElement0 {return elementList; )
}


class Composite Test{
public static void main (String [] args) {
MenuComponent mainMenu=new Menu ("Insert") ;
MenuComponent subMenu = new Menu ("Chart") ;
MenuComponent element=new Menultem ("On This Sheet") ;
( )
subMenu.addMenuElement (element) ;
printMenus (mainMenu) ,
}
private static void printMenus (MenuComponent ifile) {
          ifile.printName (;
List <MenuComponent> children=ifile.getElement0;
if (children=null)    return;
for (MenuComponent element:children) {
printMenus (element) ;
```

}

}

试题一 答案： 解析：

问题 1(5 分)

E1:车辆 E2:车主 E3:支付系统 E4:管理人员 E5:道闸控制系统

问题 2(3 分)

D1:停车记录表 D2 ：会员信息表 D3 ：基础信息表

问题 3(4 分)

计费规则信息 D3--p3

道闸控制请求 P1--P5

更新车位信息 P5--D3

更新余额 P3--D2

问题 4(3 分)

"道闸控制"加工过程

IF(道闸执行状态正常)

IF(车辆入场) THEN

将车牌号及其入场时间信息存入停车记录，修改空余车位数

ELSEIF(车辆出场)THEN

更新停车状态，修改空余车位数

ENDIF

ELSESIF(未在规定的时间内接收到其返回的执行状态正常放行） THEN

向管理人员发送异常告警信息

ENDIF

试题二 答案：　解析：　问题 1：

问题 2：
(A)团购点编号。主键:供货商编号，团购点编号；外键:供货商编号，团购点编号.
(B)客户电话.主键:订单编号；外键:团购点编号，客户电话.

问题 3：

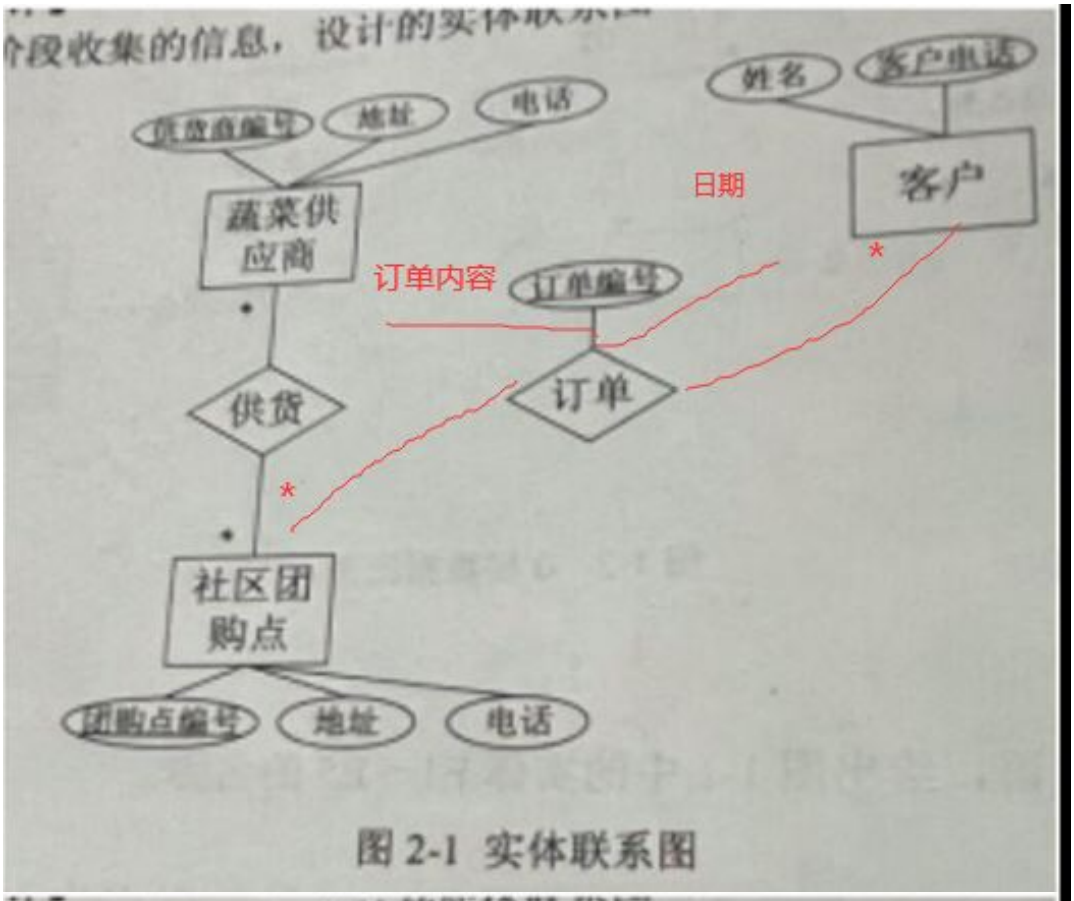

图 2-1 实体联系图

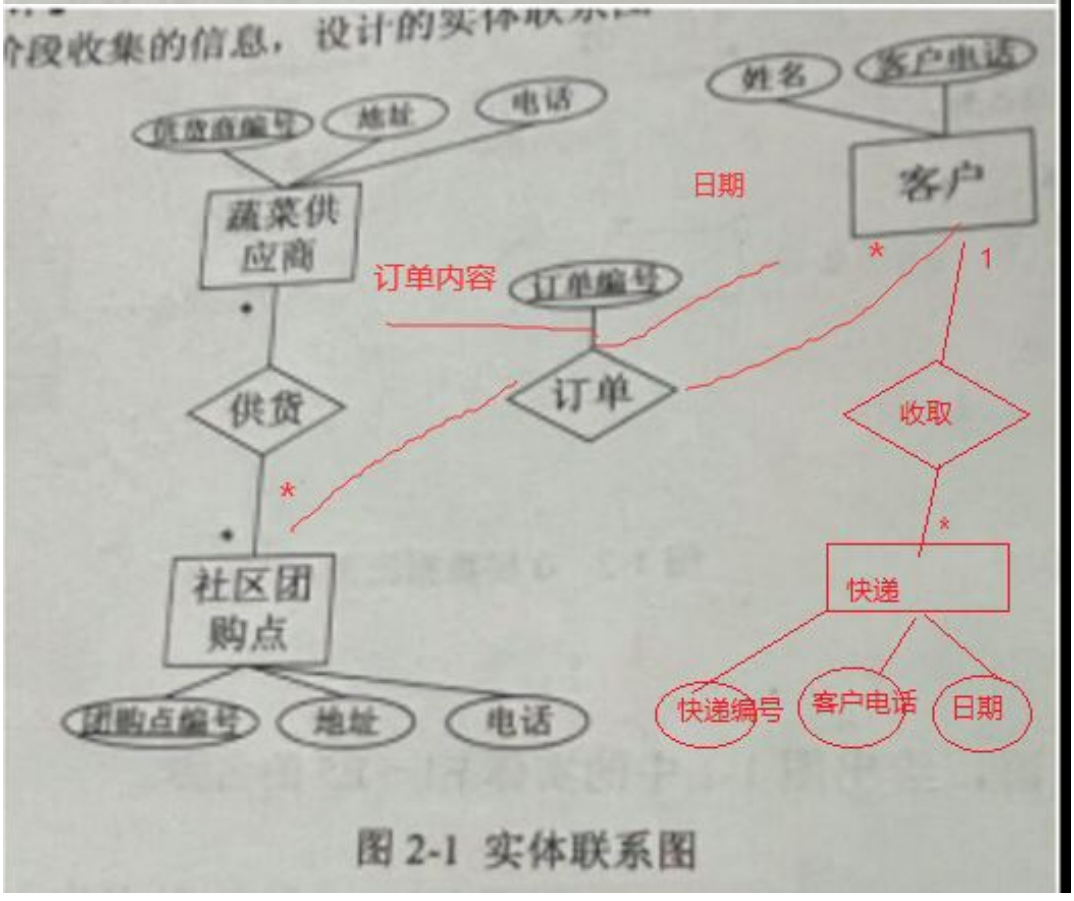

图 2-1 实体联系图

试题三 答案： 解析：

问题 1(7 分)

A1:患者:A2:快递人员 A3:药师

U1:确认处方 u2:支付方式 U3:微信支付 U4:支付宝支付(U3U4 可以互换)

问题 2(5 分)

C1:支付方式 c2:微信支付 c3:支付宝支付 C4:处方 C5:药品(C2c3 可以互换)

问题 3(3 分)

泛化(generalization)：泛化关系是一种继承关系，子用例将继承基用例的所有行为，关系和通信关系，也就是说在任何使用基用例的地方都可以用子用例来代替。泛化关系在用例图中使用空心的箭头表示，箭头方向从子用例指向基用例。

扩展(extend)： extend 关系是对基用例的扩展，基用例是一个完整的用例，即使没有子用例的参与，也可以完成一个完整的功能。

包含(include)： include 为包含关系，当两个或多个用例中共用一组相同的动作，这时可以将这组相同的动作抽出来作为一个独立的子用例，供多个基用例所共享。

试题四 答案： 解析：

问题 1(8 分)

(1)i<=N (2)j=i+r-1 (3)temp<m[i][j] (4)S[i][j]+1,j

问题 2(7 分)

(6) 0(n3)　　(7) 0(n2)

试题五 答案：　解析：

(1)protected

(2)virtual void addMenuElement(MenuComponent *element)=0;

(3)virtual list<MenuComponent*> getElement()=0;

(4)list<MenuComponent *> elementList;

(5)mainMenu->addMenuElement(subMenu);

试题六 答案：　解析： (1)protected (2)abstract boolean addMenuElement(Memucomponent element); (3)abstract List<MenuComponent>

getElement(); (4)Arraylist<MenuComponent> elementList; (5)
mainMenu.addMenuElement(subMenu);