# 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 2013 年 上半年 下午试卷 案例

（考试时间 150 分钟）

**试题一** 某慈善机构欲开发一个募捐系统，以跟踪记录为事业或项目向目标群体进行募捐而组织的集体性活动，该系统的主要功能如下所述。

(1）管理志愿者。根据募捐任务给志愿者发送加入邀请、邀请跟进、工作任务；管理志愿者提供的邀请响应、支援站信息、工作时长、工作结果等。

(2）确定募捐需求和收集所募捐赠(资金及物品)。根据需求提出的募捐任务、活动请求和募捐请求，获取所募集的资金和物品。

(3）组织募捐活动。根据活动请求，确定活动时间范围。根据活动时间，搜索场馆，即向场馆发送场馆可用性请求，获得场馆可用性。然后根据活动时间和地点推广募捐活动，根据相应的活动信息举办活动，从募款机构获取资金并向其发放赠品。获取和处理捐赠，根据捐赠要求，提供所募集的捐赠；处理与处理人之间的交互。即：

录入捐赠人信息，处理后存入捐赠人信息表；从捐赠人信息表中查询捐赠人信息，向捐赠人发送捐赠请求，并将已联系的捐赠人存入已联系的捐赠人表。根据捐赠请求进行募集，募得捐赠后，将捐赠记录存入捐赠表；对捐赠记录进行处理后，存入已处理捐赠表，向捐赠人发送致谢函。根据已联系的捐赠人和捐赠记录进行跟进，将捐赠跟进情况发送给捐赠人。

现采用机构化方法对募捐系统进行分析与设计，获得如图 1-1 、 1-2 和 1-3 所示分层数据流图。

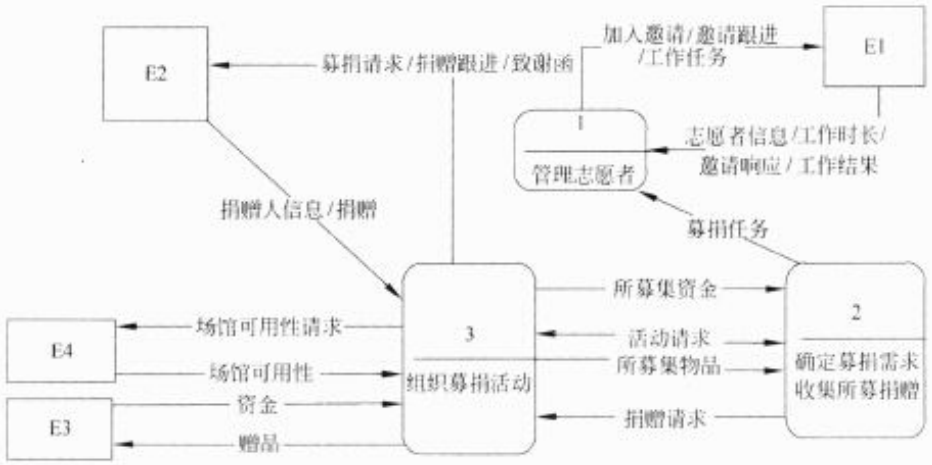



图 1-1　0 层数据流图

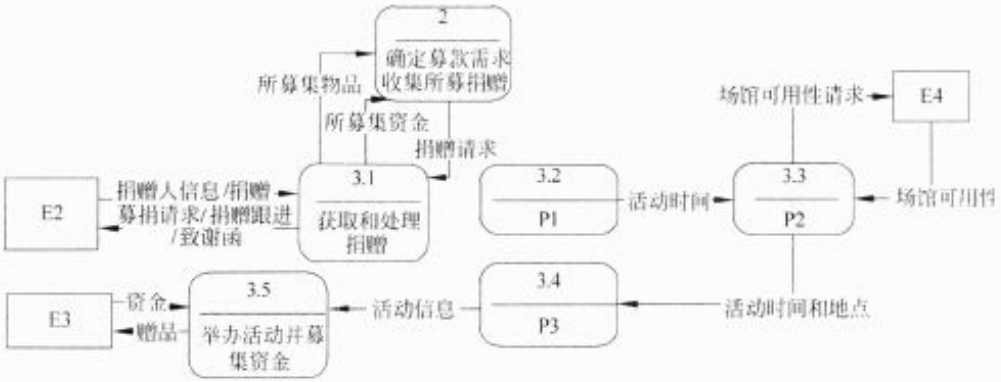



图 1-2　1 层数据流图

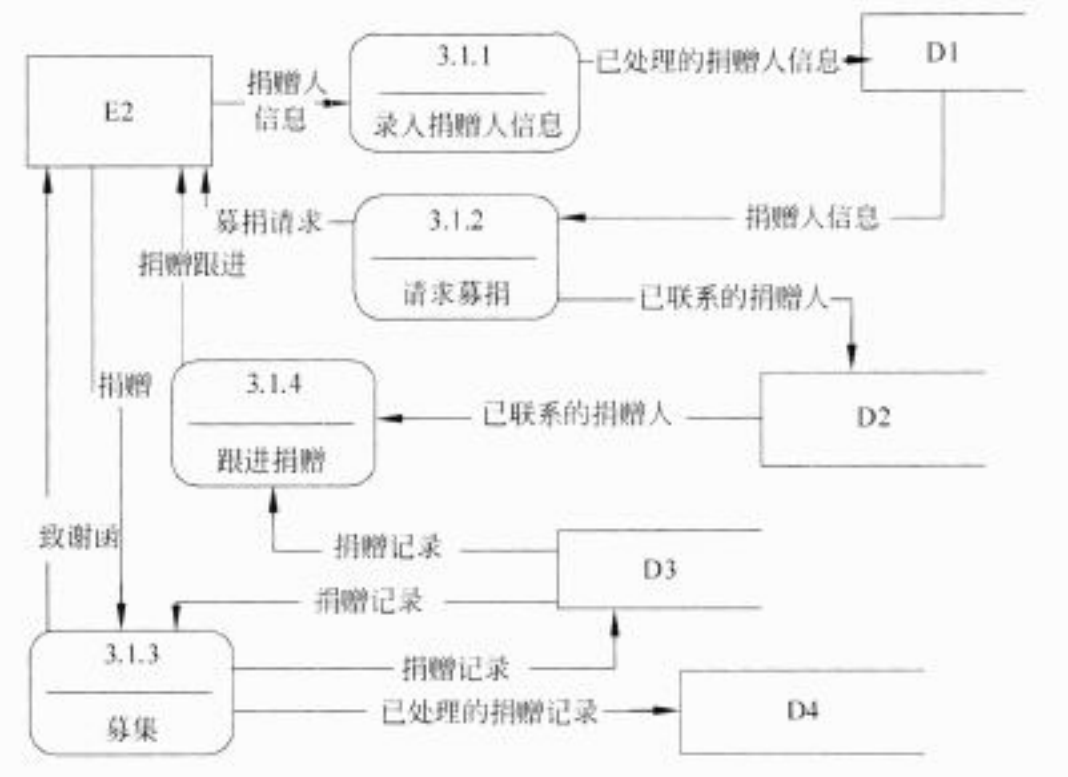



图 1-3　2 层数据流图

**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**

在建模 DFD 时，需要对有些复杂加工(处理)进行进一步精化，图 1-2 为图 1-1 中处理 3 的进一步细化的 1 层数据图，图 1-3 为图 1-2 中 3.1 进一步细化的 2 层数据流图。补全图 1-2 中加工 P1 、 P2 和 P3 的名称和图 1-2 与图 1-3 中缺少的数据流。

**问题：1.3**

使用说明中的词语，给出图 1-3 中的数据存储 D1~D4 的名称。

**试题二　【说明】**

某电视台拟开发一套信息管理系统，以方便对全台的员工、栏目、广告和演播厅等进行管理。

【需求分析】

(1) 系统需要维护全台员工的详细信息、栏目信息、广告信息和演播厅信息等。员工的信息包括：工号、姓名、性别、出生日期、电话、住址等。栏目信息主要包括：栏目名称、

播出时间、时长等。广告信息主要包括：广告编号、价格等。演播厅信息包括：房间号、房间面积等。

(2）电视台根据调度单来协调各个栏目、演播厅和场务。一销售档栏目只会占用一个演播厅，但会使用多名场务来进行演出协调。演播厅和场务可以被多个栏目循环使用。

(3）电视台根据栏目来插播广告。每档栏目可以插播多条广告，每条广告也可以在多档栏目插播。

(4）一档栏目可以有多个主持人，但一名主持人只能主持一档栏目。

(5）一名编辑人员可以编辑多条广告，一条广告只能由一名编辑人员编辑。

【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，得出如下关系模式(不完整)；

演播厅(房间号，房间面积)

栏目(栏目名称，播出时间，时长)

广告(广告编号，销售价格，__(1)__ )

员工(工号，姓名，性别，出生日期，电话，住址)

主持人(主持人工号，__(2)__ )

插播单(__(3)__ ，播出时间)

调度单(__(4)__ )

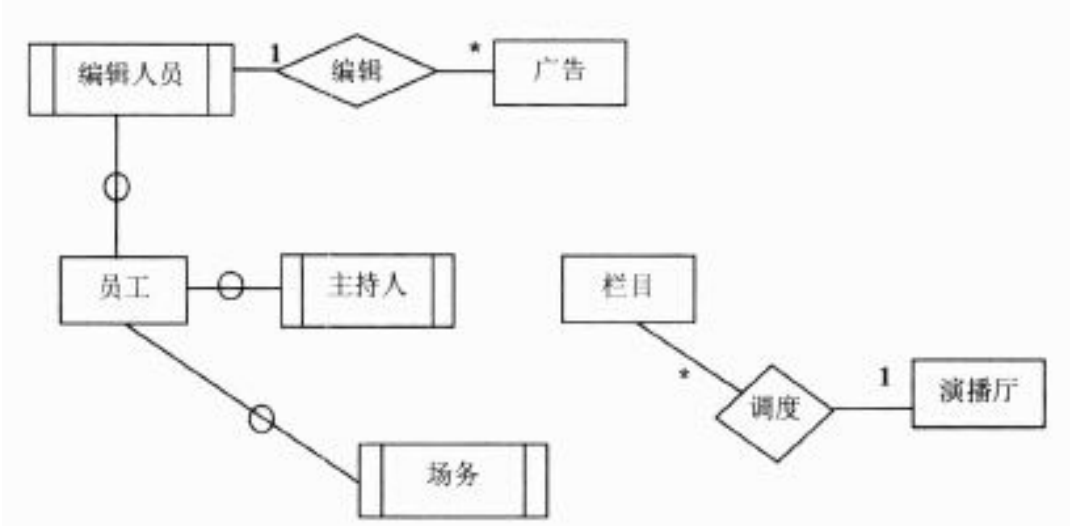



图 2-1　实体联系图

**问题：2.1**

补充图 2-1 中的联系和联系的类型。

**问题：2.2**

根据图 2-1，将逻辑结构设计阶段生成的关系模式中的空(1)~(4)补充完整，并用下划线指出空(1)~(4)所在关系模式的主键。

**问题：2.3**

现需要记录广告商信息，增加广告商实体。一个广告商可以提供多条广告，一条广告只由一个广告商提供。请根据该要求，对图 2-1 进行修改，画出修改后的实体间联系和联系的类型。

**试题三**

【说明】

某城市拟开发一个基于 Web 的城市黄页，公开发布该城市重要的组织或机构(以下统称为客户)的基本信息，方便城市生活。该系统的主要功能描述如下：

(1) 搜索信息：任何使用 Internet 的网络用户都可以搜索发布在城市黄页中的信息，例如客户的名称、地址、联系电话等。

(2) 认证：客户若想在城市黄页上发布信息，需要通过系统的认证。认证成功后，该客户成为系统授权用户。

(3）更新信息：授权用户登录系统之后，可以更改自己的在城市黄页中的相关信息，例如变更联系电话等。

(4）删除客户：对于拒绝继续在城市黄页上发布信息的客户，由系统管理员删除该客户的相关信息。

系统采用面向对象方法进行开发，在开发过程中认定出如表 3-1 所示的类。系统的用例图和类图分别如图 3-1 和 3-2 所示。

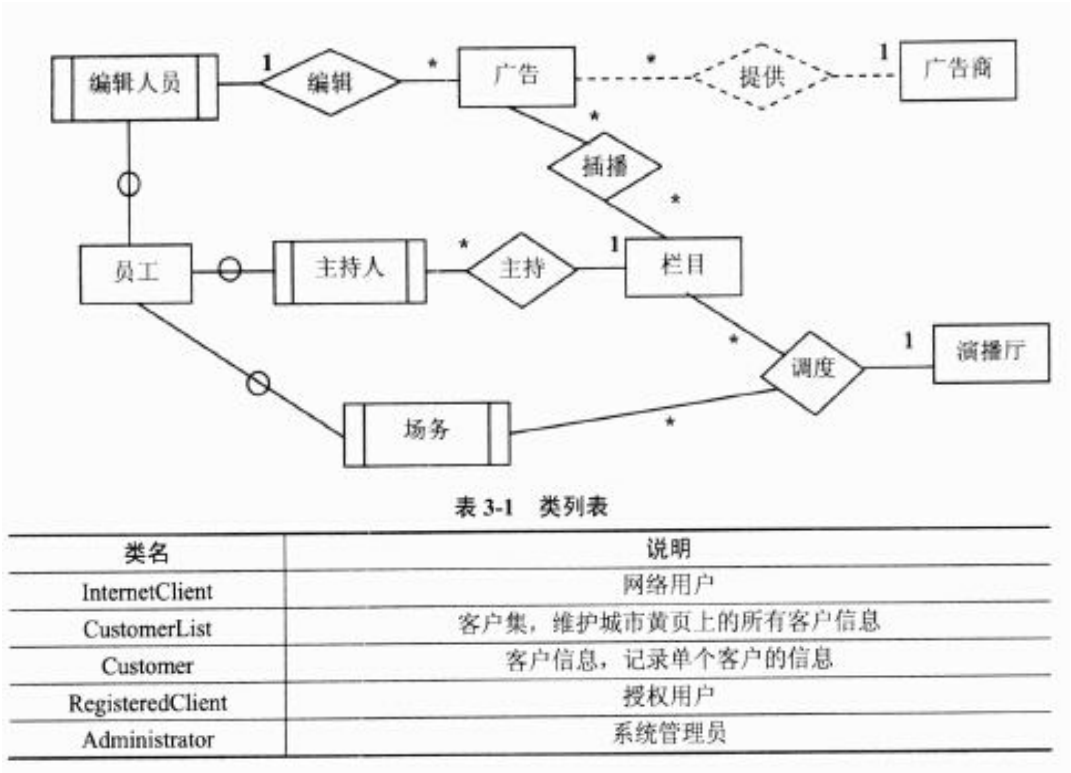



**表 3-1　类列表**

| 类名 | 说明 |
|---|---|
| InternetClient | 网络用户 |
| CustomerList | 客户集，维护城市黄页上的所有客户信息 |
| Customer | 客户信息，记录单个客户的信息 |
| RegisteredClient | 授权用户 |
| Administrator | 系统管理员 |

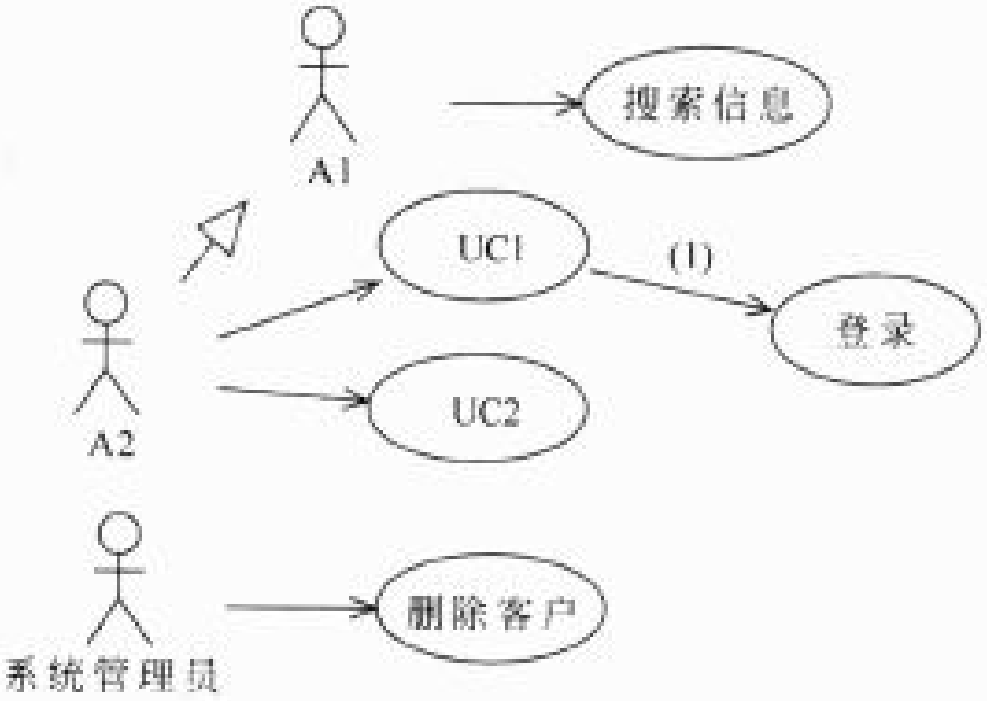



图 3-1　系统用例图

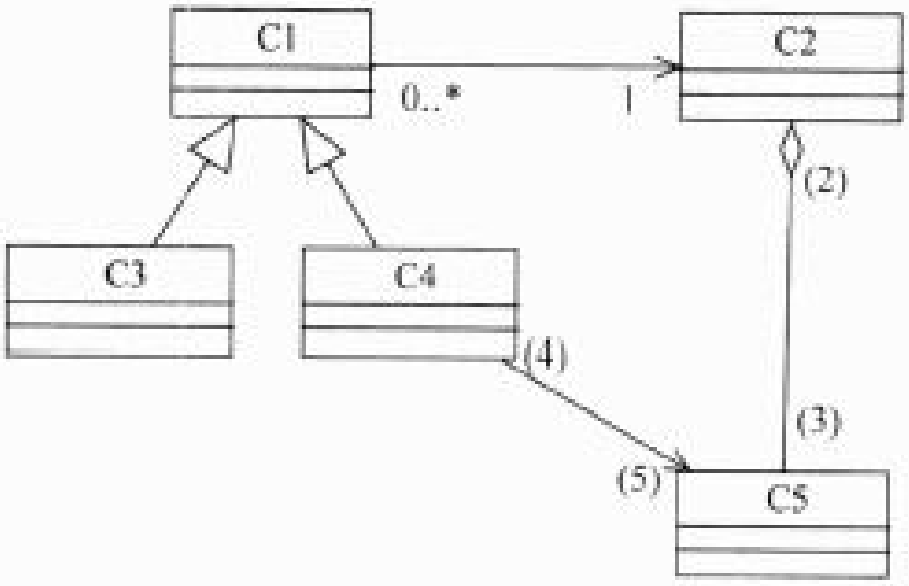



图 3-2　系统类图

**问题： 3.1**
根据说明中的描述，给出图 3-1 中的 A1 和 A2 处所对应的参与者、 UC1 和 UC2 处所对应的用例以及(1)处的关系。
**问题： 3.2**
根据说明中的描述，给出图 3-2 中的 C1~C5 所对应的类名(表 3-1 中给出的类名)和(2)~(5)处所对应的多重度。
**问题： 3.3**
认定类是面向对象分析中非常关键的一个步骤。一般首先从问题域中得到候选类集合，再根据相应的原则从该集合中删除不作为类的，剩余的就是从问题域中认定出来的类。简要说明选择题选类的原则，以及对候选类集合进行删除的原则。

**试题四** 阅读下列说明和 C 代码，回答问题 1 至问题 3, 将解答写在答题纸的对应栏内。

【说明】

设有 m 台完全相同的机器运行 n 个独立的任务，运行任务 i 所需要的时间为 $t_i$，要求确定一个调度方案，使得完成所有任务所需要的时间最短。

假设任务已经按照其运行时间从大到小顺序。算法基于最长运行时间作业优先的策略；按顺序先把每个任务分配到一台机器上，然后将剩余的任务依次放入空闲的机器。

【C 代码】

下面是算法的 C 语言实现。

(1) 常量和变量说明

m:机器数

n:任务数

t[]:输入数组，长度为 n，其中每个元素表示任务的运行时间，下标从 0 开始

s[][]:二维数组，长度为 m*n，下标从 0 开始，其中元素 s[i][j]表示机器 i 运行的任务 j 的编号

d[]:数组，长度为 m，其中元素 d[i]表示机器 i 的运行时间，下标从 0 开始

count[]:数组，长度为 m，下标从 0 开始，其中元素 count[i]表示机器 i 的运行任务数

i:循环变量

j:循环变量

k:临时变量

max:完成所有任务的时间

min:临时变量

(2）函数 schedule

```
void schedule(){

int i,j,k,max=0

for(i=0;i
d[i]=0;

for(j=0;j
s[i][j]=0;

}

}

for(i=0;i
s[i][0]=j;
```

(1) ;

count[i]=1;

}

for( (2) 、 i

int min=d[0];

k=0;

for(j=1;j

if(min>d[j]){

min=d[j];

k=j; // 机器 K 空闲

}

}

(3) ;

count[k]=count[k]+1;

d[k]=d[k]+t[i];

}

for(i=0;i

if( (4) ){

max=d[i];

```
}

}

}
```

**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，该问题采用了（5）算法设计策略，时间复杂度为（6）（用 0 符号表示）。

**问题： 4.3**

考虑实例 m=3(编号 0~2)， n=7(编号 0~6)，各任务的运行时间为{16,14,6,5,4,3,2}。则在机器 0 、 1 和 2 上运行的任务分别为(7)、(8)和(9)(给出任务编号)。从任务开始运行到完成所需要的时间为（10）。

**试题五 【说明】**

现要求实现一个能够自动生成求职简历的程序。简历的基本内容包括求职者的姓名、性别、年龄及工作经历等。希望每份简历中的工作经历有所不同，并尽量减少程序中的重复代码。

现采用原型(Prototype)模式来实现上述要求，得到如图 5-1 所示的类图。

【C++代码】

```
#include
using namespace std;
class Cloneable {
      public;
      (1);
} ;
class WorkExperience : public Cloneable { /* 工作经历 */
      private;
```

```
        string      workDate;
        string      company;
        public;
        Cloneable* Clone()
        {
              (2);
              obj->workDate     = this->workDate;
              obj->company      = this->company;
              return(obj);
        }


/* 其余代码省略 */
} ;
class Resume : public Cloneable { /* 简历 */
        private;
        string            name; string sex; string age;
        WorkExperience    * work;
        Resume( WorkExperience* work )
        {
              this->work = (3);
        }


        public;
        Resume( stringname )
/* 实现略 */
        {
        }


        voidSetPersonallnfo( stringsex, stringage; )                /*
实现略 */
        {
        }


        voidSetWorkExperience( stringworkDate, stringcompany )      /*
实现略 */
        {
        }


        Cloneable* Clone()
```

```
    {
        (4);
        obj->name   = this->name;
        obj->sex    = this->sex;
        obj->age    = this->age;
        return(obj);
    }
};
int main()
{
    Resume *a = newResume( “  ” );
        a->SetPersonallnfo( “ 男 ”, “ 29 ” );
        a->SetWorkExperience( “ 1998 ~2000 ”, “ XXX 公司 ” );
            Resume * b = (5);
    b->SetWorkExperience( “ 2001 ~2006 ”, “ YYY 公司 ” );
    return(0);
}
```

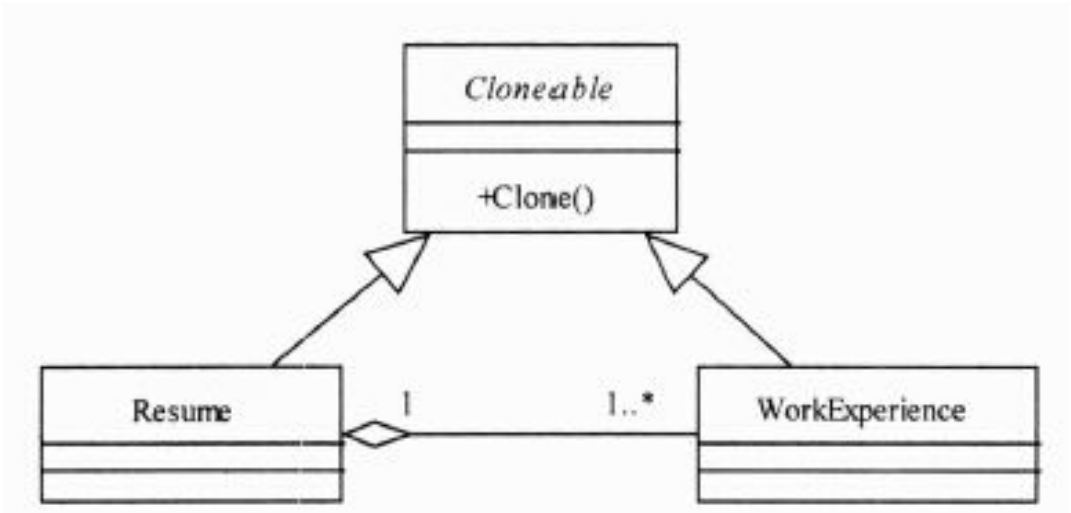



图 5-1　类图

**问题：5.1**

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。

**试题六** 【说明】

现要求实现一个能够自动生成求职简历的程序。简历的基本内容包括求职者的姓名、性别、年龄及工作经历等。希望每份简历中的工作经历有所不同，并尽量减少程序中的重复代码。

现采用原型(Prototype)模式来实现上述要求，得到如图 6-1 所示的类图。

【Java 代码】

```
class WorkExperience(1) Closeable { /* 工作经历 */
      private string    workDate;
      private string    company;
      publicObject Clone()
      {
            (2) ;
            obj.workDate      = this.workDate;
            obj.company = this.company;
            return(obj);
      }
}

class Resume(3) Closeable { /* 简历 */
      private string          name;
      privatestring           sex;
      private string          age;
      private WorkExperience  work;

      public Resume( string name )
      {
            this name = name; work = new SetWorkExperience();
      }


      private Resume( WorkExperience work )
      {
            this work = (4);
      }
```

```
      publicvoidSetPersonallnfo( stringsex, stringage; )
/* 代码略 */
      {
      }


      publicvoidSetWorkExperience( stringworkDate, stringcompany )
/* 代码略 */
      {
      }


      publicObject Clone()
      {
            Resume obj = (5);
/* 其余代码省略 */
            return(obj);
      }
}

classWorkResume {
      public static void main( string[] arg )
      {
            Resumea = newResume( “  ” ) ;
                  a.SetPersonallnfo( “ 男 ”, “ 29 ” ) ;
                  a.SetWorkExperience( “ 1998 ~2000 ”, “ XXX 公司
 ” ) ;

                     Resume b = (6);
            b.SetWorkExperience( “ 2001 ~2006 ”, “ YYY 公司 ” ) ;
      }
}
```

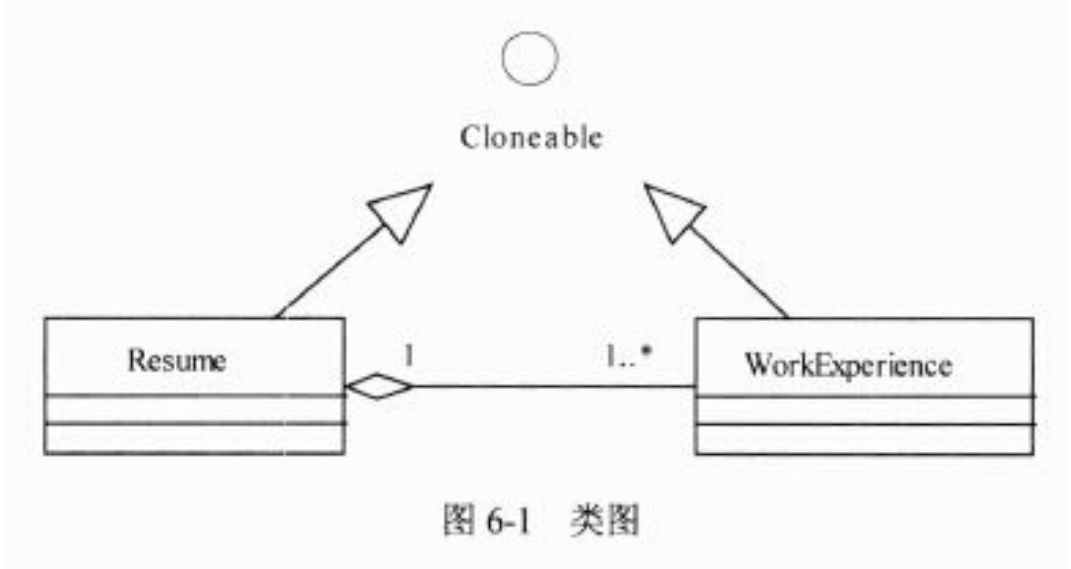


图 6-1　类图

**问题： 6.1**
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。

**试题一 答案：　　解析：**　E1 志愿者 E2 捐赠人 E3 募捐机构 E4 场馆

本题采用结构化方法进行系统分析与设计，主要考查数据流图(DFD)的应用，是比较传统的题目，要求考生细心分析题目中所描述的内容。

DFD 是一种便于用户理解、分析系统数据流程的图形化建模工具，是系统逻辑模型的重要组成部分。顶层 DFD一般用来确定系统边界，将待开发系统看作一个大的加工(处理)，然后根据系统从哪些外部实体接收数据流，以及系统将数据流发送到哪些外部实体，建模出的顶层图中只有唯一的一个加工和一些外部实体，以及这两者之间的输入输出数据流。 0 层 DFD 在顶层确定的系统外部实体以及与外部实体的输入输出数据流的基础上，将顶层 DFD 中的加工分解成多个加工，识别这些加工的输入输出数据流，使得所有顶层 DFD 中的输入数据流，经过这些加工之后变换成顶层 DFD 的输出数据流。根据 0 层 DFD 中的加工的复杂程度进一步建模加工的内容。

在建分层 DFD 时，根据需求情况可以将数据存储建模在不同层次的 DFD 中，注意在绘制下层数据流图时要保持父图与子图平衡。父图中某加工的输入输出数据流必须与它的子图的输入输出数据流在数量和名称上相同，或者父图中的一个输入(或输出)数据流对应于子图中几个输入(或输出)数据流，而子图中组成这些数据流的数据项全体正好是父图中的这一个数据流。

本问题给出 0 层 DFD，要求根据描述确定图中的外部实体。分析题目中描述，并结合已在

图中给出的数据流进行分析。从题目的说明中可以看出，与系统交互实体包括志愿者、捐赠人、募款机构和场馆，这四个作为外部实体。

P1：确定活动时间范围

P2：搜索场馆

P3：推广募捐活动

本题考查分层 DFD 的加工分解，以及父图与子图的平衡。图 1-2 中对图 1-1 的加工 3 进行进一步分解，根据说明(3) 中对加工 3 的描述对图 1-2 进行分析。首先需要确定活动时间范围，其输入数据流是活动请求，输出流为活动时间。然后是搜索场馆，其输入流为活动时间，输出活动时间和地点，同时向场馆发送的场馆可用性请求和获得的场馆可用性分别作为输入和输出数据流。在确定活动时间和地点的基础上推广募捐活动， 活动时间和地点是其输入流，活动信息作为其输出流，流向举办活动并募集资金，从募款机构获取资金并向其发放赠品，加工 2 收集募得的资金和物品，因此 3.5 还需要将所募集资金作为输出流。获取和处理捐赠(资金和物品)时以捐赠请求作为其输入流，输出流为所募集的捐赠，因为既有资金又有物品，而从募款机构募得的只有资金，将图 1-1 中加工 3 流向加工 2 的数据流，分为所募集资金和所募集物品，而 3.5 的输出流中只有所募集资金。

因此， P1 为确定活动时间范围， P2 为搜索场馆， P3 为推广募捐活动。图 1-2 中缺失了从 2 到 3.3 的活动请求和从 3.5 到 2 的所募集资金这两条数据流。

题目给出处理和捐赠人之间的交互进一步描述，对 3.1 进一步建模下层数据流图(图 1-3)。分解加工 3.1，确定相关数据流。其中根据加工 2 的捐赠请求进行募集，所募捐赠需要返回给加工 2 。

根据父图与子图的平衡原则，图 1-3 中此处也缺失了捐赠请求和所募集资金和所募集物品。

D1：捐赠人信息表 D2:已联系的捐赠人表

D3：捐赠表 D4:已处理捐赠表

本问题考查 2 层 DFD 中数据存储的确定。本案例中，数据存储的描述都是在这一部分描述给出，所以数据存储建模在此层体现。

| 数据流名称 | 起点 | 终点 |
| --- | --- | --- |
| 所募集资金 | 3.5 或 举办活动并募集资金 | 2 |
| 活动请求 | 2 | 3.2 或 确定活动时间范围 |
| 捐赠请求 | 2 | 3.1.3 募集 |
| 所募集捐赠 | 3.1.3 或 募集 | 2 |
| 或 | | |
| 所募集资金 | 3.1.3 或 募集 | 2 |
| 所募集物品 | 3.1.3 或 募集 | 2 |

注：数据流没有次序要求；表中 2 处可以是“确定募捐需求收集所募捐赠”。

**试题二 答案： 解析：**

说明：*填写为 m 和 n 均可。

本题考查数据库设计，属于比较传统的题目，考查点也与往年类似。

本问题考查数据库的概念结构设计，题目要求补充完整实体联系图中的联系和联系的类型。

根据题目的需求描述可知，一个栏目可以插播多条广告，而多条广告也可以在多个栏目中播放，因此栏目和广告之间存在“插播”联系，联系的类型为多对多(*:*，或 m:n)。

根据题目的需求描述可知，一个栏目可以有多个主持人，而一个主持人只能主持一档栏目，因此栏目和主持人之间存在“主持”联系，联系的类型为一对多(1:*，或 1:n)。

根据题目的需求描述可知，一个栏目需要使用多名场务来进行演出协调，场务可以被多个栏目循环使用，因此演播厅、栏目和场务之间存在“调度”联系，联系的类型为 1 对多对多(1:*:*，或 1:m:n)。

本问题考查数据库的逻辑结构设计，题目要求补充完整各关系模式，并给出各关系模式的主键。

根据实体联系图和需求描述，广告记录广告编号、销售价格和编辑人员工号。所以， 对于“广告”关系模式，需补充属性“广告编号”。广告编号为广告的主键。

根据实体联系图和需求描述，主持人记录主持人工号和所属的栏目名称。所以，对于“主持人”关系模式，需补充属性“主持人工号”。主持人工号为主持人的主键。

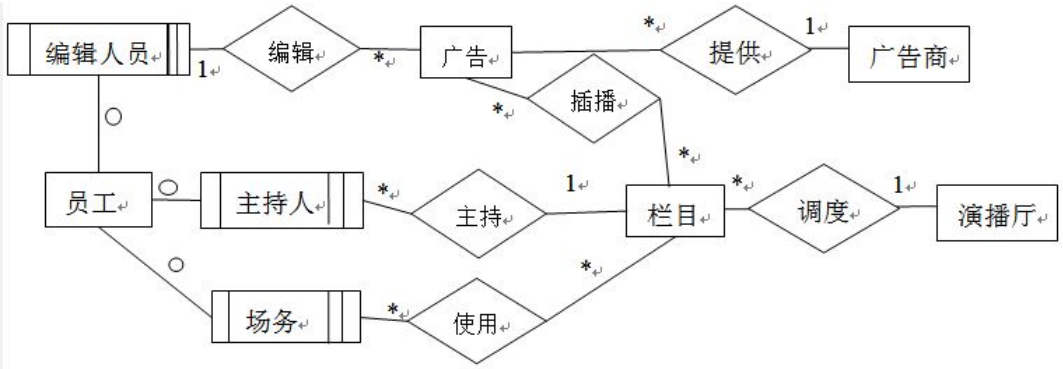
根据实体联系图和需求描述，插播单需要记录栏目名称、广告编号和播出的时间。 所以，对于“插播单”关系模式，需补充属性“栏目名称”和“广告编号”。栏目名称和广告编号联合作为插播单的主键。

根据实体联系图和需求描述，调度单需要记录栏目名称、房间号和参与的场务工号。 所以，对于“调度单”关系模式，需补充属性“栏目名称”、“房间号”和“场务工号”。

栏目名称、房间号和场务工号联合作为插播单的主键。

本问题考查数据库的概念结构设计，根据新增的需求增加实体联系图中的实体的联系和联系的类型。

根据问题描述，一个广告商可以提供多条广告，一条广告只由一个广告商提供。则须在广告商实体和广告实体之间存在“提供”联系，联系的类型为 1 对多(1:*，或 l:n)

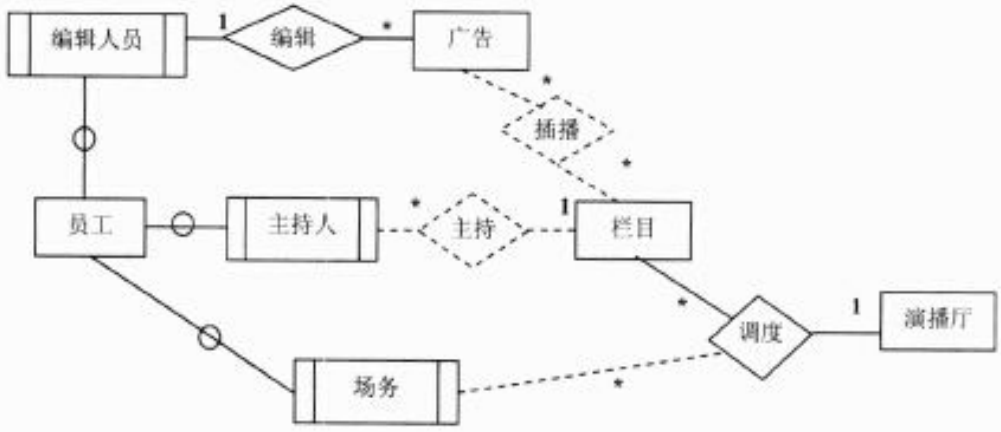



广告（广告编号，销售价格，编辑人员工号）

主持人（主持人工号，栏目名称）

插播单（栏目名称，广告编号，播出时间）

调度单（栏目名称，房间号，场务工号）




说明：*填写为 m 和 n 均可。

**试题三 答案：　解析：**　A1 ：网络用户或 IntemetClientA2：授权用户或

RegisteredClient

UC1：更新信息 UC2:认证

(1)：«include»

本题属于经典的考题，主要考查面向对象分析方法以及 UML 的用例图和类图的相关应用。

本问题要求将图 3-1 所给出的用例图补充完整。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。

本题背景描述简单，再结合图 3-1 中给出的两个用例“搜索信息”和“删除客户”，很容易确定出“认证”、“更新信息”就是需要补充的用例。下面只需要确定参与者 A1 、 A2 以及这两个用例与 A1 、 A2 之间的关系即可将图补充完整。在说明中出现了两类角色：客户和系统管理员。系统管理员已经作为参与者明确标识在 3-1 中了。那么 A1 和 A2 只能从“客户”这个角色中产生。明显地，“客户”在说明中被细分成了两类：网络用户和授权用户，而认证用户是一种特殊的网络用户。因此， A1 应该对应“网络用户”， A2 应该对应“授权客户”。要成为授权用户，必须首先经过认证。成为授权用户之后，不仅可以搜索信息，还具有更新信息的权限。更新信息时要求首先登录系统，因此“登录” 是“更新信息”操作中所包含的一个必须步骤。由此可以确定 UC1 对应“更新信息”， UC2 对应“认证”，而(1) 处的关系应该是«include»

Cl： IntemetClientC2： CustomerListC3： RegisteredClient

C4:AdministratorC5： Customer

(2) 1 (3) 0..* (4) 0..1 (5) 0..1

本问题考查类建模。表 3-1 已经给出了类列表，这样对照图 3-2 寻找对应关系即可。图 3-2 中有两个明显的结构：继承(Cl 、 C3 和 C4 之间)和聚集(C2 和 C5 之间)。

先确定聚集关系相关的类。由表 3-1 可以找出两个明显具有部分整体概念的类：CustomerList 和 Customer,由类的说明可以看出， CustomerList 表示整体概念，Customer 表示部分概念。由此可以确定 C2 对应 CustomerList，C5 对应 Customer 。同时可以确定出这两个类之间的多重度，即(2) 处应为 1 ，(3) 处为 0..*。

下面来确定继承关系相关的类。图 3-1 中已经出现了一个继承关系(A1 和 A2 之间)， 这就给出了一个明显的提示：图 3-2 中的继承关系与这两类角色相关。回到表 3-1 中，发现了 3 个与角色相关的类： IntemetClient (网络用户)、 RegisteredClient (授权用户)和 Administrator (系统管理员)。由于已经确定了 C5 是 Customer,而能够对 Customer 进行操作的只有系统管理员。因此具有继承关系的这 3 个类应分别是： C1-网络用户， C3-授权用户， C4-系统管理员。(4)、(5) 处的多重度也可以确定下来，均为 0..1 。

选择候选类时通常考虑的是问题域中自然存在的名词。

具有下列特征的候选类需要删除：含义相近(冗余)、含义不明确的对象、暗示实 现方式的、表示属性或特征、有动词含义的名词(表示行为和方法)。

本问题考查面向对象分析过程中认定类/对象的过程。通常分为两个步骤进行，首先将问题域(需求分析)中所有自然存在的名词都选出来，构成候选类集合。然后针对这个候选类集合，将满足以下原则的名词从候选类集合中删除：含义相近(冗余)、含义不明确的对象、暗示实现方式的、表示属性或特征、有动词含义的名词(表示行为和方法)。 最后剩余的就是所认定的类/对象。

**试题四 答案： 解析：** (1) d[i] = t[i] (2) i=m (3) s[k][count[k]] =i (4) max

本题考查算法设计与分析技术以及算法的C语言实现，是比较传统的题目，要求考生细心分析题目中所描述的内容。

根据题中说明和代码注释，算法首先初始化数组d和s中的元素；然后将m个任务分配到m台机器上，此时将任务0， 1，...， m-1分别分配到机器0， 1，…， m-1上，同时设置d、 s和count数组中的相关元素的值，故空格(1) 填写d[i] = t[i];接下来将剩下的n-m个任务分配到m台机器上，从任务m开始，因此空格(2) 填写i=m，确定首先空闲的机器k，将当前尚未分配的第一个任务分配到机器k上，并设置d、 s和count数组中的相关元素的值，故空格(3) 填写s[k][c0unt[k]] = i;最后确定从任务开始到结束所需要的时间，从所有机器的运行时间中选择运行时间最长的机器的运行时间，即最大的d[i]，因此空格(4) 填写max

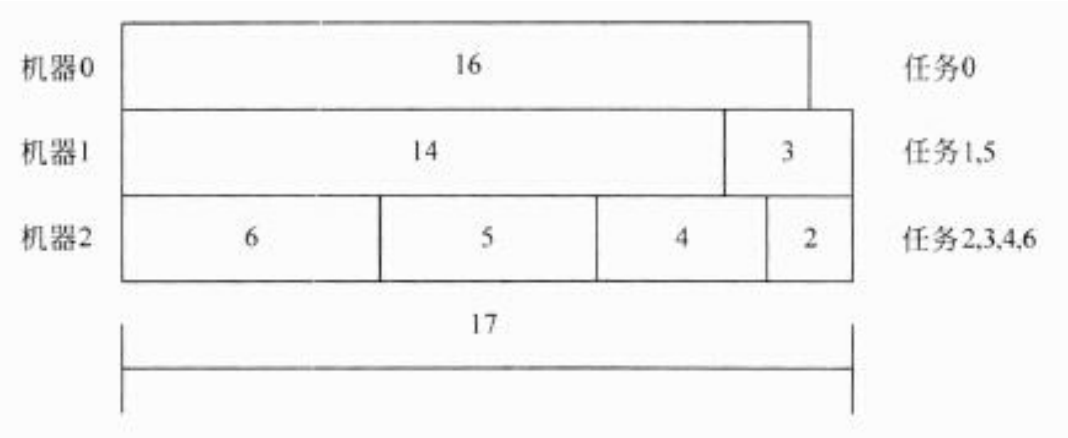
(5) 贪心 (6) O(mn)

根据上述C代码，算法有两处两重循环，时间复杂度为O(mn)，有两处一重循环，时间复杂度为O(m)，因此整个算法的时间复杂度为O(mn)。

(7) 0 (8) 1,5 (9) 2,3,4,6 (10) 17

根据说明中的算法思想和C代码，首先将任务0、 1和2分配到机器0、 1和2上运行，运行时间分别为16、 14和6。由于任务2的时间最短，故任务3在机器2上运行，机器2上的运行时间为6+5=11，仍然是时间最短，任务4继续在机器2上运行，机器2上的运行时间为11+4=15。此时机器1上的运行时间最短，任务5在机器1上运行，机器1上的运行时间为14+3=17。此时机器2上的运行时间最短，任务6在机器2上运行， 机器2上的运行时间为15+2=17。所以任务分配完成。根据此分配，在机器0上运行任务0,运行时

间为 16;机器 1 上运行任务 1 和 5, 运行时间为 17;在机器 2 上运行任务 2，3，4 和 6, 运行时间为 17 。因此从任务开始到结束的时间为 17 。




**试题五 答案：　解析：**　(1) virtual Cloneable* Clone() = 0

(2) WorkExperience *obj = new WorkExperience()

(3) (WorkExperience *)work->Clone()

(4) Resume *obj = new Resume(this->work)

(5) (Resume *)a->Clone()

本题考查原型(Prototype)模式的概念及应用。

Prototype 模式是一种对象创建型模式。 Prototype 模式通过给出一个原型对象来指明所要创建的对象的类型，然后通过复制这个原型对象的方法，创建出更多同类型的对象。原型模式又可以分为两种：浅克隆和深克隆。浅克隆仅仅复制所考虑的对象，而不复制它所引用的对象，也就是其中的成员对象并不复制；深克隆除了对象本身被复制外，对象包含的引用也被复制，即成员对象也被复制。

原型模式的优点是：

(1) 向客户隐藏制造新实例的复杂性；

(2) 提供让客户能够产生未知类型对象的选项；

(3) 在某些环境下，复制对象比创建新对象更有效。

在一个复杂的类层次中，当系统必须从其中的许多类型创建新对象时，可以考虑原型模式。使用原型模式的缺点是：对象的复制有时相当复杂。

Prototype 模式的结构如下图所示。

题目利用原型模式来实现自动生成多份求职简历，而且每份简历的工作经历有所差别。题目中使用的是原型模型中的深克隆。题目中的类 Cloneable 对应上图中的类 Prototype，Resume 和 WorkExperience 分别是两个具体克隆类。由于 WorkExperience 是 Resume 的一部分，所以在 Resume 中有一个 WorkExperience 类型的成员对象。

下面来分析程序。

第(1) 空出现在类 Cloneable 的定义中。类 Cloneable 的作用是声明一个克隆自身的接口，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;

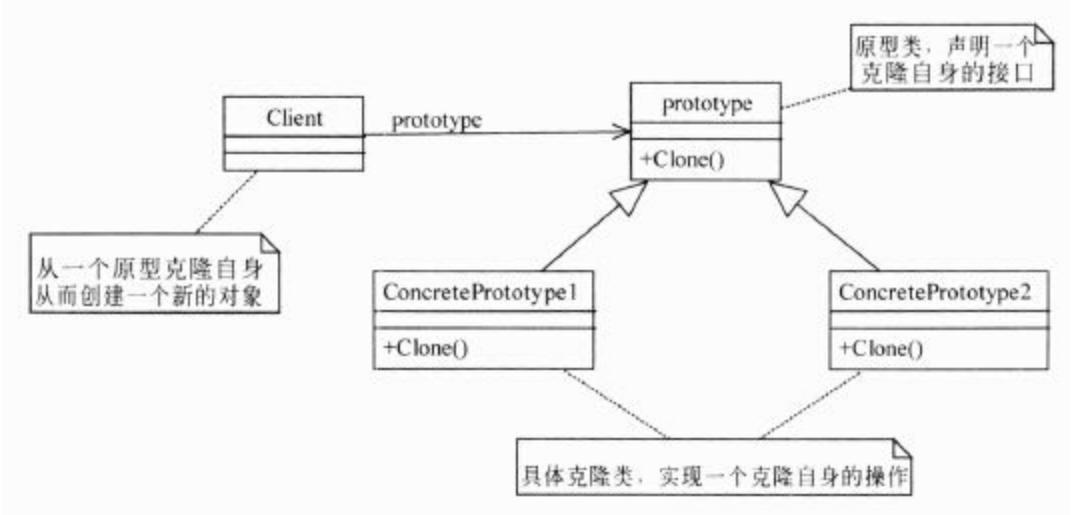
下面来确定纯虚拟函数的原型。由类图 5-1 以及 Cloneable 的子类 WorkExperience 中定义的成员函数 Clone() 可知，(1) 空应该填写 virtualCloneable* Clone() = 0 。空(2)出现在类 WorkExperience 成员函数 Clone() 的定义中。 WorkExperience 作为具体克隆类，应实现继承自父类的纯虚拟函数 Ck>ne() 。根据已给出的代码的提示，在 Clone() 方法中，应再创建一个 WorkExperience 对象，而这个对象的名称就是 obj 。因此

(2) 空处应填 WorkExperience *obj = newWorkExperience() 0

本题采用的是深克隆，除了对象本身被复制外，成员对象也被复制。 work 是 Resume 中的成员对象，所以(3) 空是对成员对象的克隆，因此应调用 work 对象的克隆操作， 即( WorkExperience *)work->Clone() 。

第(4)空出现在 Resume 的 Clone() 方法中，功能与第(2) 空类似，完成 Resume 对象的克隆，所以第(4)空应填 Resume *obj = newResume(this->work)。

第(5) 空是对原型模式的应用。用简历 a 克隆出简历 b, 即调用 a 的克隆操作，所以第(5)空应填(Resume *)a->Clone() 0



**试题六 答案：　解析：**　(1) implements

(2) WorkExperience obj = new WorkExperience()

(3) implements

(4) (WorkExperience)work.Clone()

(5) new Resume(this.work)

(6) (Resume)a.Clone()

本题考查原型(Prototype)模式的概念及应用。

Prototype 模式是一种对象创建型模式。 Prototype 模式通过给出一个原型对象来指明所要创建的对象的类型，然后通过复制这个原型对象的方法，创建出更多同类型的对象。 原型模式又可以分为两种：浅克隆和深克隆。浅克隆仅仅复制所考虑的对象，而不复制它所引用的对象，也就是其中的成员对象并不复制；深克隆除了对象本身被复制外，对象包含的引用也被复制，即成员对象也被复制。

原型模式的优点是：

(1) 向客户隐藏制造新实例的复杂性；

(2) 提供让客户能够产生未知类型对象的选项；

(3) 在某些环境下，复制对象比创建新对象更有效。

在一个复杂的类层次中，当系统必须从其中的许多类型创建新对象时，可以考虑原型模式。使用原型模式的缺点是：对象的复制有时相当复杂。

Prototype 模式的结构如下图所示。

题目利用原型模式来实现自动生成多份求职简历，而且每份简历的工作经历有所差别。题目中使用的是原型模型中的深克隆。题目中的类 Cloneable 对应上图中的类 Prototype，Resume 和 WorkExperience, 分别是两个具体克隆类。由于 WorkExperience 是 Resume 的一部分，所以在 Resume 中有一个 WorkExperience 类型的成员对象。

下面来分析程序。

Cloneable 是 Java 中的一个接口，其中已经定义 fClone() 接口，可以直接在程序中使用。对于两个具体克隆类 Resume 和 WorkExperience 来说，只要实现 Cloneable 即可。 因此第(1) 、(3) 空均填写 implements 。

空(2) 出现在类 WorkExperience 成员函数 Clone() 的定义中。 WorkExperience 作为具体克隆类，应实现 Cloneable 中的方法 Clone() 。根据已给出的代码的提示，在 Clone() 方法中，应再创建一个 WorkExperience 对象，而这个对象的名称就是 obj 。因此(2) 空处应填 WorkExperienceobj = newWorkExperience() 。

本题采用的是深克隆，除了对象本身被复制外，成员对象也被复制。 work 是 Resume 中的

成员对象，所以(4)空是对成员对象的克隆，因此应调用 work 对象的克隆操作， 即(WorkExperience)work.Clone() 。

第(5) 空出现在 Resume 的 Clone() 方法中，功能与第(2) 空类似，完成 Resume 对象的克隆，所以第(5) 空应填 newResume(this.work)。

第(6) 空是对原型模式的应用。用简历 a 克隆出简历 b, 即调用 a 的克隆操作，所以第(6)空应填(Resume)a.Clone() 。