2009 年上半年软件设计师考试下午真题（参考答案）

● 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】

某集团公司拥有多个大型连锁商场，公司需要构建一个数据库系统以方便管理其业务运作活动。

【需求分析结果】

1. 商场需要记录的信息包括商场编号（编号唯一），商场名称，地址和联系电话。某商场信息如表 2-1 所示。

2-1 商场信息表

商场编号	商场名称	地址	联系电话
PS2101	淮海商场	淮海中路 918 号	021-64158818
PS2902	西大街商场	西大街时代盛典大厦	029-87283220
PS2903	东大街商场	碑林区东大街 239 号	029-87450287
PS2901	长安商场	雁塔区长安中路 38 号	029-85264953

2. 每个商场包含有不同的部门，部门需要记录的信息包括部门编号（集团公司分配），部门名称，位置分布和联系电话。某商场的部门信息如表 2-2 所示。

2-2 部门信息表

部门编号	部门名称	位置分布	联系电话
DT002	财务部	商场大楼六层	82504342
DT007	后勤部	商场地下副一层	82504347
DT021	安保部	商场地下副一层	82504358
DT005	人事部	商场大楼六层	82504446
DT001	管理部	商场裙楼三层	82504668

3. 每个部门雇用多名员工处理日常事务，每名员工只能隶属于一个部门（新进员工在培训期不隶属于任何部门）。员工需要记录的信息包括员工编号（集团公司分配），姓名，岗位，电话号码和工资。员工信息如表 2-3 所示。

2-3 员工信息表

员工编号	姓名	岗位	电话号码	工资
XA3310	周　超	理货员	13609257638	1500.00
SH1075	刘　飞	防损员	13477293487	1500.00
XA0048	江雪花	广播员	15234567893	1428.00
BJ3123	张正华	部门主管	13345698432	1876.00

4. 每个部门的员工中有一名是经理，每个经理只能管理一个部门，系统需要记录每个经理的任职时间。

【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：

【关系模式设计】

商场（商场编号，商场名称，地址，联系电话）
部门（部门编号，部门名称，位置分布，联系电话，（a））
员工（员工编号，员工姓名，岗位，电话号码，工资，（b））
经理（（c），任职时间）

【问题 1】

根据问题描述，补充四个联系，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，联系的类型分为 1:1、1:n 和 m:n。

【问题 2】

根据实体联系图，将关系模式中的空（a）~（c）补充完整，并分别给出部门、员工和经理关系模式的主键和外键。

【问题 3】

为了使商场有紧急事务时能联系到轮休的员工，要求每位员工必须且只能登记一位紧急联系人的姓名和联系电话，不同的员工可以登记相同的紧急联系人。则在图 2-1 中还需添加的实体是（1），该实体和图 2-1 中的员工存在（2 联系（填写联系类型）。给出该实体的关系模式。

● 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】

某银行计划开发一个自动存提款机模拟系统（ATM System）。系统通过读卡器（CardReader）读取 ATM 卡；系统与客户（Customer）的交互由客户控制台（CustomerConsole）实现；银行操作员（Operator）可控制系统的启动（System Startup）和停止（System Shutdown）；系统通过网络和银行系统（Bank）实现通信。当读卡器判断用户已将 ATM 卡插入后，创建会话（Session）。会话开始后，读卡器进行读卡，并要求客户输入个人验证码（PIN）。系统将卡号和个人验证码信息送到银行系统进行验证。验证通过后，客户可从菜单选择如下事务（Transaction）：

1. 从 ATM 卡账户取款（Withdraw）；
2. 向 ATM 卡账户存款（Deposit）；

3. 进行转账（Transfer）；

4. 查询（Inquire）ATM 卡账户信息。

一次会话可以包含多个事务，每个事务处理也会将卡号和个人验证码信息送到银行系统进行验证。若个人验证码错误，则转个人验证码错误处理（Invalid PIN Process）。每个事务完成后，客户可选择继续上述事务或退卡。选择退卡时，系统弹出 ATM 卡，会话结束。系统采用面向对象方法开发，使用 UML 进行建模。系统的顶层用例图如图 3-1 所示，一次会话的序列图（不考虑验证）如图 3-2 所示。消息名称参见表 3-

可能的消息名称列表

名　称	说　明	名　称	说　明
cardInserted()	ATM 卡已插入	performTransaction()	执行事务
performSession()	执行会话	readCard()	读卡
readPIN()	读取个人验证码	PIN	个人验证码信息
creat(atm, this, card, pin)	为当前会话创建事务	create(this)	为当前 ATM 创建会话
card	ATM 卡信息	doAgain	执行下一个事务
ejectCard()	弹出 ATM 卡		

1。

【问题 1】（7 分）

根据【说明 】中的描述，给出图 3-1 中 A1 和 A2 所对应的参与者，U1 至 U3 所对应的用例，以及该图中空 （1） 所对应的关系。（U1 至 U3 的可选用例包括：Session、Transaction、Insert CarD．Invalid PIN Process 和 Transfer）

【问题 2】（6 分）

根据【说明 】中的描述，使用表 3-1 中的英文名称，给出图 3-2 中 6～9 对应的消息。

【问题 3】（2 分）

解释图 3-1 中用例 U3 和用例 Withdraw、Deposit 等四个用例之间的关系及其内涵。

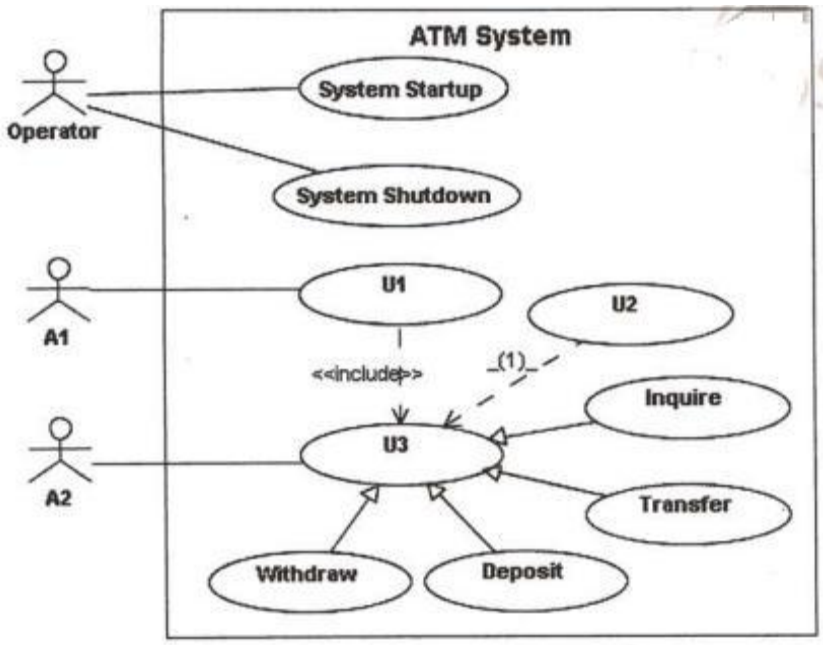


图 3-1　ATM 系统顶层用例图

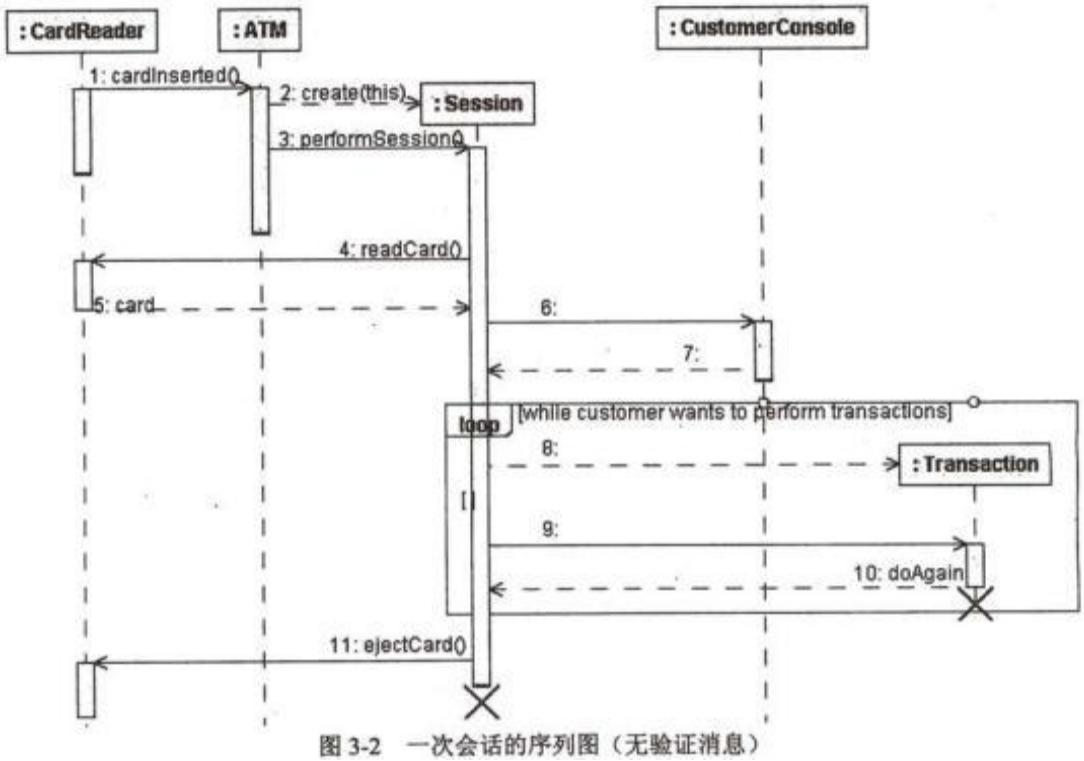


图 3-2　一次会话的序列图（无验证消息）

● 阅读下列说明，回答问题 1 和问题 2，将解答填入答题纸的对应栏内。

【说明】

现需在某城市中选择一个社区建一个大型超市，使该城市的其它社区到该超市的距离总和最小。用图模型表示该城市的地图，其中顶点表示社区，边表示社区间的路线，边上的权重表

示该路线的长度。现设计一个算法来找到该大型超市的最佳位置：即在给定图中选择一个顶点，使该顶点到其它各顶点的最短路径之和最小。算法首先需要求出每个顶点到其它任一顶点的最短路径，即需要计算任意两个顶点之间的最短路径；然后对每个顶点，计算其它各顶点到该顶点的最短路径之和；最后，选择最短路径之和最小的顶点作为建大型超市的最佳位置。

【问题 1】（12 分）

本题采用 Floyd-Warshall 算法求解任意两个顶点之间的最短路径。 已知图 G 的顶点集合为 V= {1，2，...，n }，W= ｛Wij｝n*n 为权重矩阵。设 d (k)ij=为从顶点 i 到顶点 j 的一条最短路径的权重。当 k = 0 时，不存在中间顶点，因此 d(0)ij=wij 当 k >0 时，该最短路径上所有的中间顶点均属于集合 {1，2， ...， k｝若中间顶点包括顶点 k ，则 d(k)ij=d(k-1)ik+d(k-1)kj 若中间顶点不包括顶点则 d(k-1)ij=d(k-1)i 于是得到如下递归式

$$d_{ij}^{(k)}=\begin{cases} w_{ij} & k=0 \\ \min(d_{ij}^{(k-1)}, d_{ik}^{(k-1)}+d_{kj}^{(k-1)}) & k>0 \end{cases}$$

因为对于任意路径，所有的中间顶点都在集合{1，2， ...， n｝ 内，因此矩阵 D(n)=｛d(n)ij｝n*n 给出了任意两个顶点之间的最短路径，即对所有 i， j ∈V，d(n)ij 表示顶点 i 到顶点 j 的最短路径。

下面是求解该问题的伪代码，请填充其中空缺的 (1)至(6)处。 伪代码中的主要变量说明如下：

W：权重矩阵

n： 图的顶点个数

SP：最短路径权重之和数组，SP[i]表示顶点 i 到其它各顶点的最短路径权重之和，i 从 1 到 n

min_SP：最小的最短路径权重之和

min_v：具有最小的最短路径权重之和的顶点

i：循环控制变量

j：循环控制变量

k：循环控制变量

```
LOCATE -SHOPPINGMALL(W, n)
1     D(0)=W
2     for          (1)
3               for i = 1 to n
4                         for j = 1 to n
5                        if d(k-1)ij<=d(k-1)ik+d(k-1)kj
6                (2)
7               else
8                                                    (3)
9     for i = 1 to n
10              SP[i] = 0
11              for j = 1 to n
12                                   (4)
13     min_SP = SP[1]
14            (5)
15     for i = 2 to n
16            if min_SP > SP[i]
17                      min_SP = SP[i]
18                      min_v = i
19  return          (6)
```

【问题 2】（3 分）

【问题 1】中伪代码的时间复杂度为（7）用O 符号表示）。

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

【说明】

对二叉树进行遍历是二叉树的一个基本运算。遍历是指按某种策略访问二叉树的每个结点，且每个结点仅访问一次的过程。函数 InOrder__(9)__借助栈实现二叉树的非递归中序遍历运算。

设二叉树采用二叉链表存储，结点类型定义如下：

```
typedef struct BtNode{
ElemTypedata;/*结点的数据域，ElemType 的具体定义省略*/
struct BtNode *lchild，*rchild;/*结点的左、右孩子指针域*/
}BtNode， *BTree;
```

在函数 InOrder__(10)__中，用栈暂存二叉树中各个结点的指针，并将栈表示为不含头结点的单向链表（简称链栈），其结点类型定义如下：

```
typedef struct StNode{  /*链栈的结点类型*/
  BTree elem;  /*栈中的元素是指向二叉链表结点的指针*/
  struct StNode *link;
}StNode;
```

假设从栈顶到栈底的元素为 en、en-1、…、e1，则不含头结点的链栈示意图如图 5-1 所示。

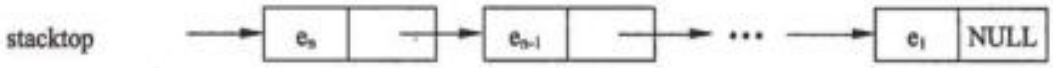


图 5-1 链栈示意图

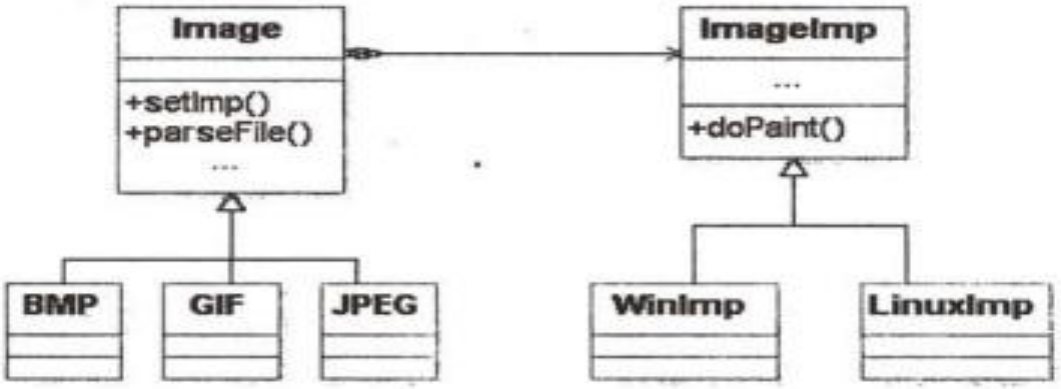
【C 函数】

```
int   InOrder(BTree root)                    /* 实现二叉树的非递归中序遍历   */
{
        BTree ptr;                           /* ptr 用于指向二叉树中的结点   */
        StNode *q;                           /* q 暂存链栈中新创建或待删除的结点指针*/
        StNode *stacktop = NULL;             /*  初始化空栈的栈顶指针 stacktop */
ptr = root;                                  /* ptr 指向二叉树的根结点   */
while (      (1)        || stacktop != NULL) {
        while (ptr != NULL) {
                q = (StNode *)malloc(sizeof(StNode));
                if (q == NULL)
return -1;
                q->elem = ptr;
                    (2)      ;
stacktop = q;                    /*stacktop 指向新的栈顶*/
ptr =       (3)      ;      /*进入左子树*/
        }
q = stacktop;
    (4)      ;                   /*栈顶元素出栈*/
visit(q);                                    /*visit 是访问结点的函数，其具体定义省略
*/
ptr =       (5)      ;                   /*进入右子树*/
free(q);                                     /*释放原栈顶元素的结点空间*/
}
return 0;
}/*InOrder*/
```

● 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】

现欲实现一个图像浏览系统，要求该系统能够显示 BMP、JPEG 和 GIF 三种格式的文件，并且能够在 Windows 和 Linux 两种操作系统上运行。系统首先将 BMP、JPEG 和 GIF 三种格式的文件解析为像素矩阵，然后将像素矩阵显示在屏幕上。系统需具有较好的扩展性以支持新的文件格式和操作系统。为满足上述需求并减少所需生成的子类数目，采用桥接（Bridge）设计模式进行设计所得类图如图 6-1 所示。



类图

采用该设计模式的原因在于：系统解析 BMP、GIF 与 JPEG 文件的代码仅与文件格式相关，而在屏幕上显示像素矩阵的代码则仅与操作系统相关。

【C++代码】

```
class Matrix{     //各种格式的文件最终都被转化为像素矩阵
  //此处代码省略
};
class ImageImp{
public:
  virtual void doPaint(Matrix m) = 0;      //显示像素矩阵 m
};

class WinImp : public ImageImp{
public:
void doPaint(Matrix m){ /*调用 windows 系统的绘制函数绘制像素矩阵*/ }
};

class LinuxImp : public ImageImp{
public:
void doPaint(Matrix m){ /*调用 Linux 系统的绘制函数绘制像素矩阵*/   }
};
class Image {
public:
  void setImp(ImageImp *imp){      (1)          = imp;}
  virtual void parseFile(string fileName) = 0;
protected:
       (2)          *imp;
};

class      BMP : public Image{
public:
  void parseFile(string fileName){
    //此处解析 BMP 文件并获得一个像素矩阵对象 m
        (3)       ;// 显示像素矩阵 m

  }
};
class      GIF : public Image{
//此处代码省略
};

class      JPEG : public Image{
  //此处代码省略
};

void main(){
  //在 windows 操作系统上查看 demo.bmp 图像文件
  Image *image1 =        (4)       ;
  ImageImp *imageImp1 =        (5)       ;
       (6)       ;
```

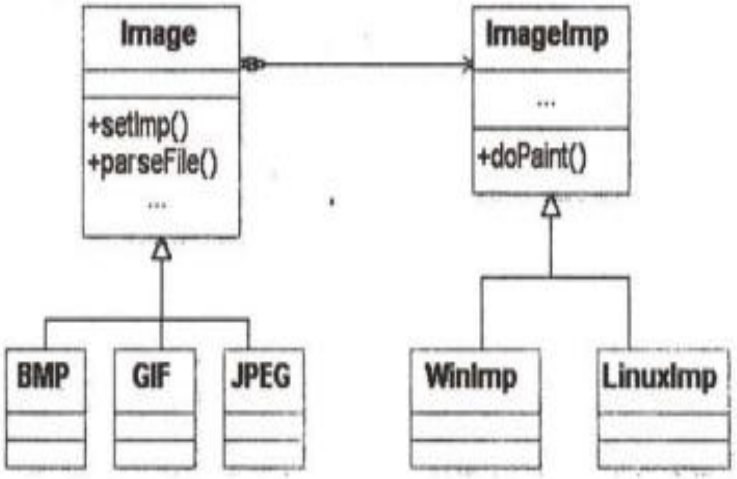
现假设该系统需要支持 10 种格式的图像文件和 5 种操作系统，不考虑类 Matrix，若采用桥接设计模式则至少需要设计（7）个类。

● __(19)__（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】

现欲实现一个图像浏览系统，要求该系统能够显示 BMP、JPEG 和 GIF 三种格式的文件，并且能够在 Windows 和 Linux 两种操作系统上运行。系统首先将 BMP、JPEG 和 GIF 三种格式的文件解析为像素矩阵，然后将像素矩阵显示在屏幕上。系统需具有较好的扩展性以支持新的文件格式和操作系统。为满足上述需求并减少所需生成的子类数目，采用桥接（Bridge）设计模式进行设计所得类图如图 7-1 所示



类图

采用该设计模式的原因在于：系统解析 BMP、GIF 与 JPEG 文件的代码仅与文件格式相关，而在屏幕上显示像素矩阵的代码则仅与操作系统相关。

【Java 代码】

```
class Matrix{     //各种格式的文件最终都被转化为像素矩阵
  //此处代码省略
};

abstract class ImageImp{
public   abstract void doPaint(Matrix m);      //显示像素矩阵 m
};

class WinImp extends ImageImp{
public void doPaint(Matrix m){ /*调用 windows 系统的绘制函数绘制像素矩阵*/ }
};

class LinuxImp extends ImageImp{
public void doPaint(Matrix m){/*调用 Linux 系统的绘制函数绘制像素矩阵*/}
};

abstract class Image {
public void setImp(ImageImp imp){
     （1）          =imp; }
public abstract void parseFile(String fileName);
protected         （2）              imp;
};

class      BMP extends Image{
public   void parseFile(String fileName){
     //此处解析 BMP 文件并获得一个像素矩阵对象 m
          （3）       ;// 显示像素矩阵 m

  }
};
class      GIF extends Image{
//此处代码省略
};

class      JPEG extends Image{
  //此处代码省略
};

public class javaMain{
public static void main(String[] args){
  //在 windows 操作系统上查看 demo.bmp 图像文件
  Image image1 =        （4）       ;
  ImageImp imageImp1 =        （5）       ;
       （6）       ;
  image1.parseFile("demo.bmp");
}
```

现假设该系统需要支持 10 种格式的图像文件和 5 种操作系统，不考虑类 Matrix，若采用桥接设计模式则至少需要设计（7）个类。