全国计算机技术与软件专业技术资格（水平）考试

2009 年上半年 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 7 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

2009 年上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“5 月 20 日”，故在答题纸的对应栏内写上“5”和“20”（参看下表）。

例题	解答栏
（1）	5
（2）	20

试题一至试题四是必答题

试题一

假设某大型商业企业由商品配送中心和连锁超市组成，其中商品配送中心包括采购、财务、配送等部门。为实现高效管理，设计了商品配送中心信息管理系统，其主要 功能描述如下：

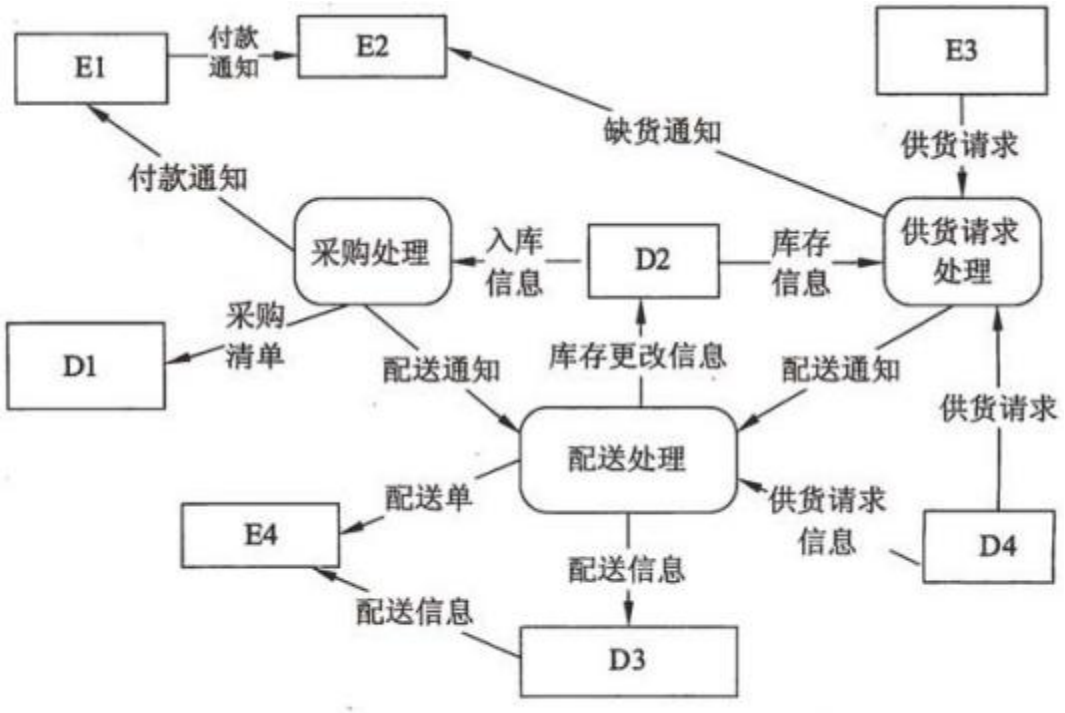
1. 系统接收由连锁超市提出的供货请求，并将其记录到供货请求记录文件。

2. 在接到供货请求后，从商品库存记录文件中进行商品库存信息查询。如果库存满足供货请求，则给配送处理发送配送通知：否则，向采购部门发出缺货通知。

3. 配送处理接到配送通知后，查询供货请求记录文件，更新商品库存记录文件，并向配送部门发送配送单，在配送货品的同时记录配送信息至商品配送记录文件。

4. 采购部门接到缺货通知后，与供货商洽谈，进行商品采购处理，合格商品入库，并记录采购清单至采购清单记录文件、向配送处理发出配送通知，同时通知财务部门给供货商支付货款。

该系统采用结构化方法进行开发，得到待修改的数据流图如下图所示。



数据流图

【问题 1】

使用【说明】中的词语，给出上图中外部实体 E1 至 E4 的名称和数据存储 D1 至 D4 的

名称。

【问题 2】

以上数据流图中存在四处错误数据流，请指出各自的起点和终点；若将上述四条错 误数据流删除，为保证数据流图的正确性，应补充三条数据流，请给出所补充数据流的 起点和终点。（起点和终点请采用上述数据流图中的符号或名称）

错误数据流

起点	终点

补充的数据流

起点	终点

试题三

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。 某商场信息如下表所本。

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下表所示。

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。员工信息如下表所示。

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】

实体联系图

【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，竞善图 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。

试题三

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(Customer-Console)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。

【问题1】

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】

根据【说明】中的描述，使用消息名称列表中的英文名称，给出图3-2中6～9对应的消息。

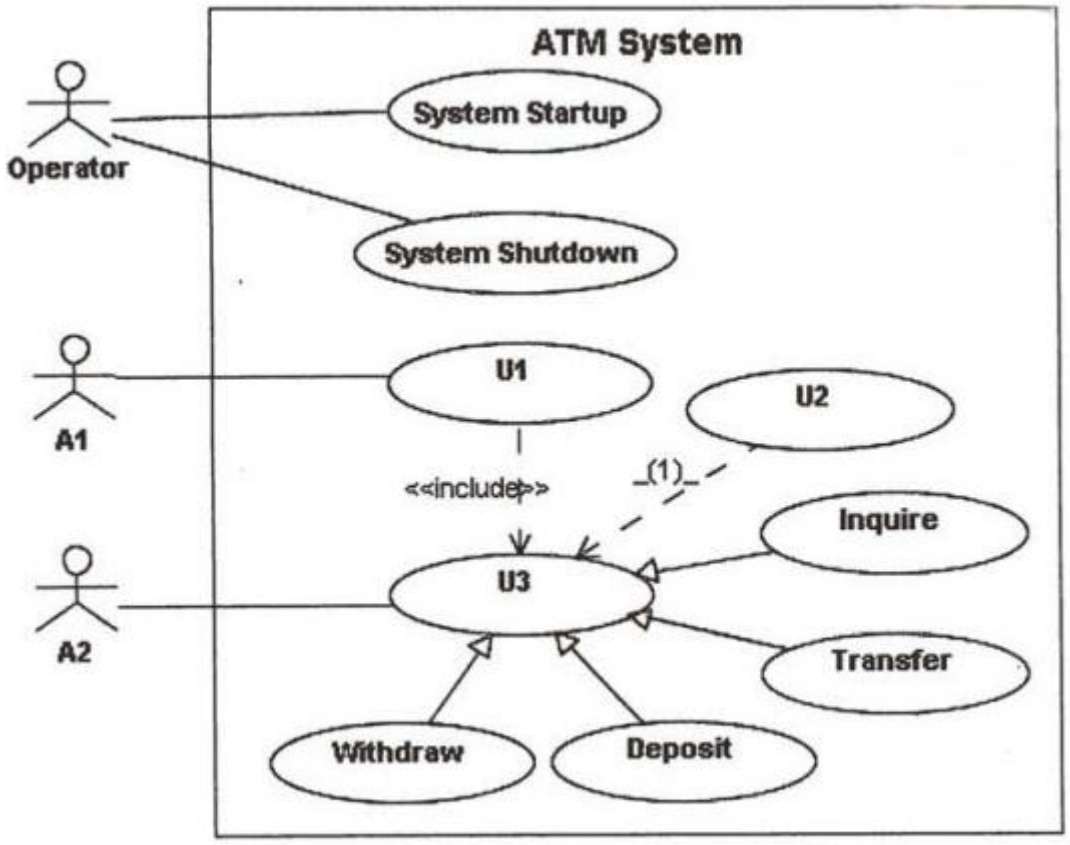


图 3-1　ATM 系统顶层用例图

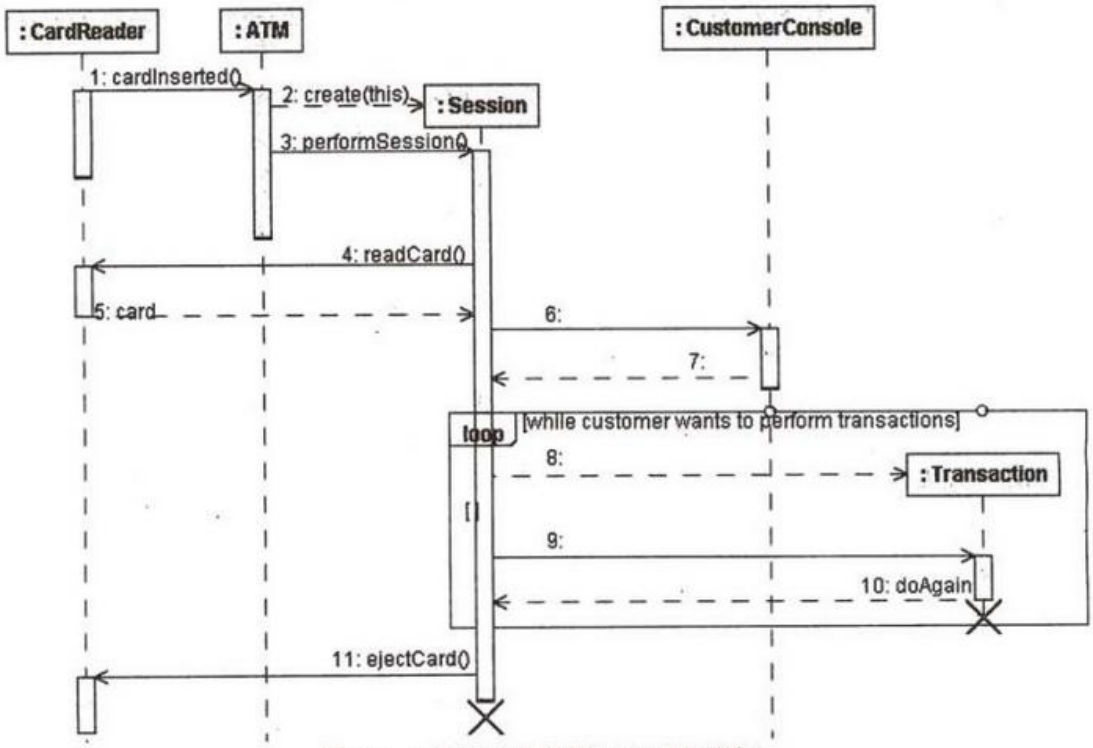


图 3-2　一次会话的序列图（无验证消息）

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		

【问题 3】

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。

试题四

现需在某城市中选择一个社区建一个大型超市，使该城市的其他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，使该顶点到其他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他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，计算其他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】

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 $V=\{1,2,\cdots,n\}$，$W=\{w_{ij}\}_{n*n}$ 为权重矩阵。设 $d_{ij}^{(k)}$ 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 $d_{ij}^{(0)}=w_{ij}$；当 $k>0$ 时，该最短路径上所有的中间顶点均属于集合 $\{1,2,\cdots,k\}$。若中间顶点包括顶点 k，则 $d_{ij}^{(k)}=d_{ik}^{(k-1)}+d_{kj}^{(k-1)}$；若中间顶点不包括顶点 k，则 $d_{ij}^{(k)}=d_{ij}^{(k-1)}$。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 $\{1,2,\cdots,n\}$ 内，因此矩阵 $D^{(n)}=\{d_{ij}^{(n)}\}_{n*n}$ 给出了任意两个顶点之间的最短路径，即对所有 $i,j\in V$，$d_{ij}^{(n)}$ 表示顶点 i 到顶点 j 的最短路径。

```
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_ij^(k-1) ≤ d_ik^(k-1) + d_kj^(k-1)
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
```

【问题 2】

【问题 1】中伪代码的时间复杂度为（7)(用 0 符号表示)。

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 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节点，且每个节点仅访问一次的过程。函数 InOrder()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，节点类型定义如下：

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

在函数 InOrder()中，用栈暂存二叉树中各个节点的指针，并将栈表示为不含头节点的单向链表（简称链栈），其节点类型定义如下：

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

假设从栈顶到栈底的元素为 e_n、e_{n-1}、…、e_1，则不含头节点的链栈示意图如图 5-1 所示。

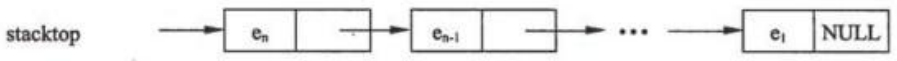


图 5-1　链栈示意图

【问题 1】

【C 函数】

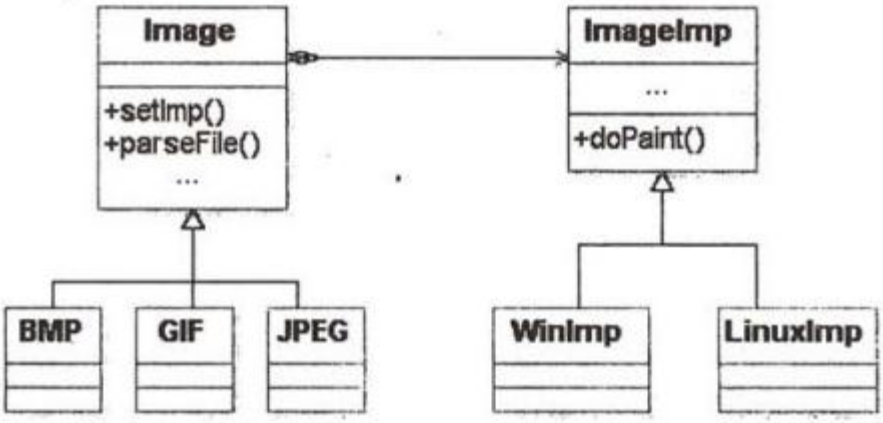
```
int  InOrder(BTree root)          /*实现二叉树的非递归中序遍历*/
{
    BTree ptr;                    /*ptr 用于指向二叉树中的节点*/
    StNode *q;                    /*q 暂存链栈中新创建或待删除的节点指针*/
    StNode *stacktop = NULL;      /*初始化空栈的栈顶指针 stacktop*/
    ptr = root;                   /*ptr 指向二叉树的根节点*/
    while ( __(1)__ || 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 是访问节点的函数，其具体定义省略*/
       ptr = __(5)__;             /*进入右子树*/
       free(q);                   /*释放原栈顶元素的节点空间*/
  }
  return 0;
}/*InOrder*/
```

试题六

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，所得类图如下图所示。



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*/ }
};
```

【问题1】

```
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) ;

    image1->parseFile("demo.bmp");

}
```

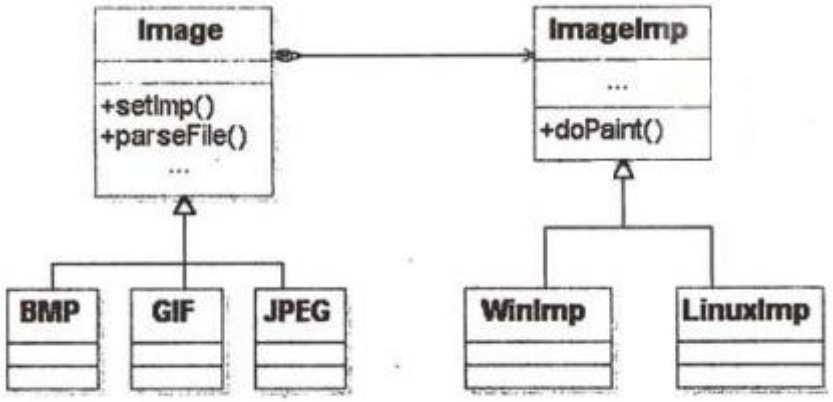
现假设该系统需要支持 10 种格式的图像文件和 5 种操作系统，不考虑类 Matrix，若采用桥接设计樽式则至少需要设计(7) 个类。

试题七

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，所得类图如下图所示。



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。

【问题 1】

【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");
    }
}
```