

# 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 下半年 下午试卷 案例

（考试时间 150 分钟）

**试题一** 某大学欲开发一个基于 Web 的课程注册系统，该系统的主要功能如下：

### 1. 验证输入信息

(1) 检查学生信息：检查学生输入的所有注册所需信息。如果信息不合法，返回学生信息不合法提示；如果合法，输出合法学生信息。

(2) 检查学位考试结果：检查学生提供的学位考试结果。如果不合法，返回学位考试结果不合法提示；如果合法，检查该学生注册资格。

(3) 检查学生注册资格：根据合法学生信息和合法学位考试结果，检查该学生对欲选课程的注册资格。如果无资格，返回无注册资格提示；如果有注册资格，则输出注册学生信息（包含选课学生标识）和欲注册课程信息。

### 2. 处理注册申请

(1) 存储注册信息：将注册学生信息记录在学生库。

(2) 存储所注册课程：将选课学生标识与欲注册课程进行关联，然后存入课程库。

(3) 发送注册通知：从学生库中读取注册学生信息，从课程库中读取所注册课程信息，给学生发送接受提示；给教务人员发送所注册课程信息和已注册学生信息。

现采用结构化方法对课程注册系统进行分析与设计，获得如图 1-1 所示的 0 层数据流图和图 1-2 所示的 1 层数据流图。

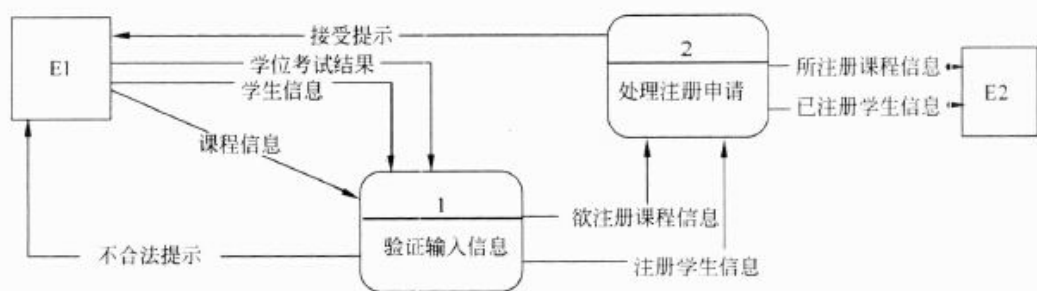


图 1-1 0 层数据流图

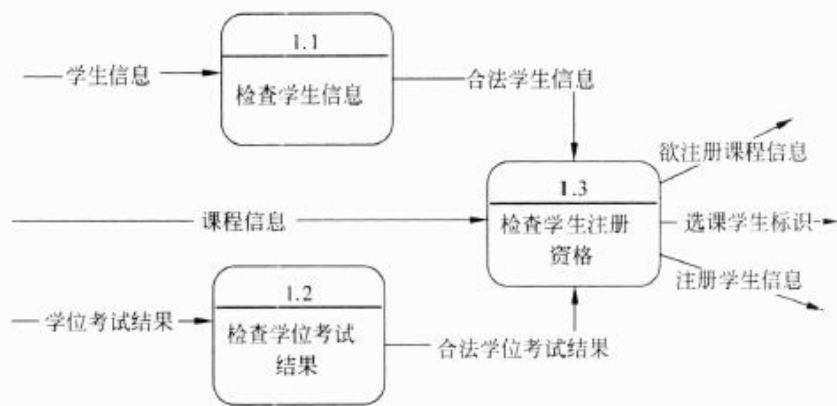


图 1-2 1 层数据流图

**问题：1.1**

使用说明中的词语，给出图 1-1 中的实体 E1 和 E2 的名称。

**问题：1.2**

使用说明中的词语，给出图 1-2 中的数据存储 D1 和 D2 的名称。

**问题：1.3**

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**

根据补充完整的图 1-1 和图 1-2, 说明上层的哪些数据流是由下层的哪些数据流组合而成。

**试题二** 某快递公司为了方便管理公司物品运送的各项业务活动，需要构建一个物品运送信息管理系统。

**【需求分析结果】**

(1) 快递公司有多多个分公司，分公司信息包括分公司编号、名称、经理、办公电话和地址。每个分公司可以有多个员工处理分公司的日常业务，每名员工只能在一个分公司工作。每个分公司由一名经理负责管理分公司的业务和员工，系统需要记录每个经理的任职时间。

(2) 员工信息包括员工号、姓名、岗位、薪资、手机号和家庭地址。其中，员工号唯一标识员工信息的每一个元组。岗位包括经理、调度员、业务员等。业务员根据客户提交的快件申请单进行快件受理事宜，一个业务员可以受理多个客户的快件申请，一个快件申请只能由一个业务员受理。调度员根据已受理的申请单安排快件的承运事宜，例如：执行承运的业务员、运达时间等。一个业务员可以执行调度员安排的多个快件的承运业务。

(3) 客户信息包括客户号、单位名称、通信地址、所属省份、联系人、联系电话、银行账号。其中，客户号唯一标识客户信息的每一个元组。当客户要寄快件时，先要提交快件申请单，申请号由系统自动生成。快件申请信息包括申请号、客户号、发件人、发件人电话、快件名称、运费、发出地、收件人、收件人电话、收件地址。其中，一个申请号对应唯一的一个快件申请，一个客户可以提交多个快件申请，但一个快件申请由唯一的一个客户提交。

**【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(图 2-1)和关系模式(不完整)如下：

**【关系模式设计】**

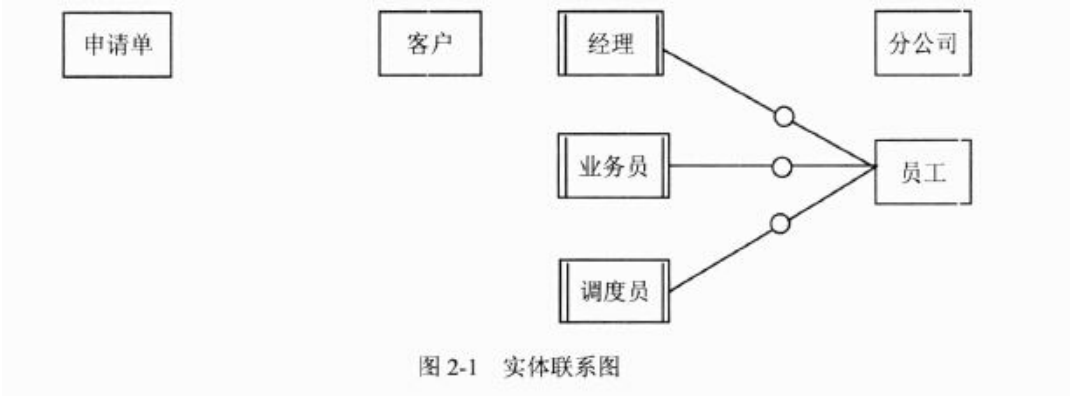
分公司(分公司编号，名称，经理，办公电话，地址)

员工(员工号，姓名，(a)，岗位，薪资，手机号，家庭地址)

客户(客户号，单位名称，通信地址，所属省份，联系人，联系电话，银行账号)申请单((

b) ,发件人, 发件人电话, 发件人地址, 快件名称, 运费, 收件人, 收件人电话, 收件地址, 受理标志, 业务员)

安排承运((c) , 实际完成时间, 调度员)



**问题： 2.1**

根据问题描述，补充五个联系，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、联系 4 和联系 5 代替，联系的类型分为 1:1、1:n 和 m:n (或 1:1、1:\* 和 \*:\*)。

**问题： 2.2**

(1) 根据实体联系图，将关系模式中的空(a) ( c)补充完整。(2) 给出员工、申请单和安排承运关系模式的主键和外键。

**问题： 2.3**

(1) 客户关系的通信地址可以进一步分为邮编、省、市、街道，那么该属性是否属于简单属性，为什么？请用 100 字以内的文字说明。(2) 假设分公司需要增设一位经理的职位，那么分公司与经理之间的联系类型应修改为(d) , 分公司的主键应修改为 (e) 。

**试题三** 某航空公司会员积分系统(CFrequentFlyer)的主要功能描述如下：

乘客只要办理该航空公司的会员卡，即可成为普卡会员(CBasic)。随着飞行里程数的积累，可以从普卡会员升级到银卡会员(CSilver)或金卡会员(CGold)。非会员(CNonMember)不能累积里程数。

每年年末，系统根据会员在本年度累积的里程数对下一年会员等级进行调整。

普卡会员在一年内累积的里程数若满 25,000 英里但不足 50,000 英里，则自动升级为银卡会员；若累积的里程数在 50,000 英里以上，则自动升级为金卡会员。银卡会员在一年内累积的里程数若在 50,000 英里以上，则自动升级为金卡会员。

若一年内没有达到对应级别要求的里程数，则自动降低会员等级。金卡会员一年内累积的里程数若不足 25,000 英里，则自动降级为普卡会员；若累积的里程数达到 25,000 英里，但是不足 50,000 英里，则自动降级为银卡会员。银卡会员一年内累积的里程数若不足 25,000 英里，则自动降级为普卡会员。

采用面向对象方法对会员积分系统进行分析与设计，得到如图 3-1 所示的状态图和图 3-2 所示的类图。

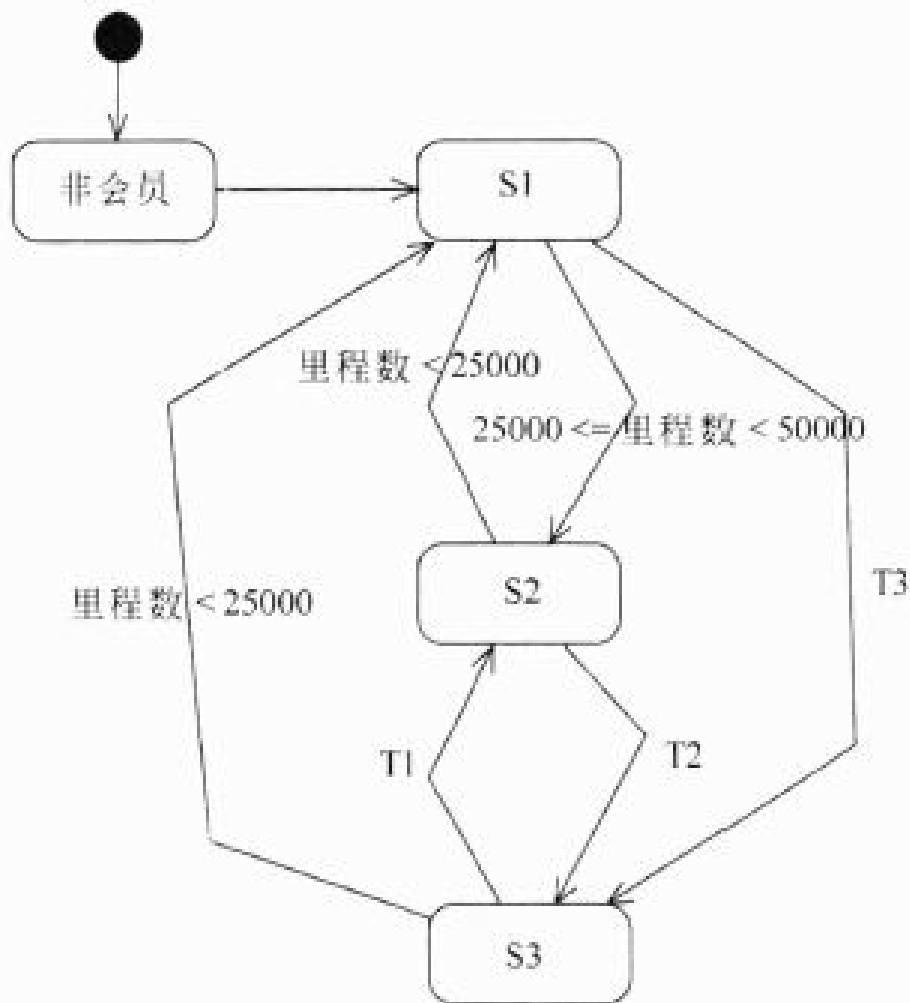


图 3-1 状态图

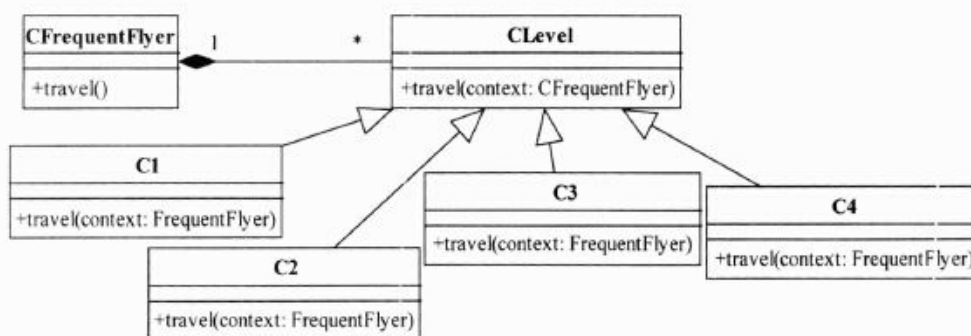


图 3-2 类图

### 问题： 3.1

根据说明中的描述，给出图 3-1 中 S1 S3 处所对应的状态以及 T1~T3 处所对应的迁移的名称。

**问题： 3.2**

根据说明中的描述，给出图 3-2 中 C1 C4 所对应的类名（类名使用说明中给出的英文词汇）。

**问题： 3.3**

图 3-2 所示的类图中使用了哪种设计模式？在这种设计模式下，类 CFrequentFlyer 必须具有的属性是什么？ C1 C4 中的 travel 方法应具有什么功能？

**试题四** 某工程计算中要完成多个矩阵相乘(链乘)的计算任务。

两个矩阵相乘要求第一个矩阵的列数等于第二个矩阵的行数，计算量主要由进行乘法运算的次数决定。采用标准的矩阵相乘算法，计算  $A_{m \times n} * B_{n \times p}$ ，需要  $m*n*p$  次乘法运算。

矩阵相乘满足结合律，多个矩阵相乘，不同的计算顺序会产生不同的计算量。以矩阵  $A_{10 \times 100}$ ， $A_{2_{100 \times 5}}$ ， $A_{3_{5 \times 50}}$  三个矩阵相乘为例，若按  $(A_1 * A_2) * A_3$  计算，则需要进行  $10*100*5+10*5*50=7500$  次乘法运算；若按  $A_1 * (A_2 * A_3)$  计算，则需要进行  $100*5*50+10*100*50=75000$  次乘法运算。可见不同的计算顺序对计算量有很大的影响。

矩阵链乘问题可描述为：给定  $n$  个矩阵，矩阵  $A_i$  的维数为  $p_M P_i$ ，其中  $i=1, 2, \dots, n$ 。确定一种乘法顺序，使得这  $n$  个矩阵相乘时进行乘法的运算次数最少。

由于可能的计算顺序数量非常庞大，对较大的  $n$ ，用蛮力法确定计算顺序是不实际的。经过对问题进行分析，发现矩阵链乘问题具有最优子结构，即若  $A_1 * A_2 * \dots * A_n$  的一个最优计算顺序从第  $k$  个矩阵处断开，即分为  $A_1 * A_2 * \dots, * A_k$  和  $A_{k+1} * A_{k+2} * \dots, * A_n$  两个子问题，则该最优解应该包含  $A_1 * A_2 * \dots, * A_k$  的一个最优计算顺序和  $A_{k+1} * A_{k+2} * \dots, * A_n$  的一个最优计算顺序。据此构造递归式，

其中， $cost[i][j]$  表示  $A_{i+1} * A_{i+2} * \dots * A_{j+1}$  的最优计算的计算代价。最终需要求解  $cost[0][n-1]$ 。

**【C 代码】**

算法实现采用自底向上的计算过程。首先计算两个矩阵相乘的计算量，然后依次计算 3 个矩阵、4 个矩阵…… $n$  个矩阵相乘的最小计算量及最优计算顺序。下面是该算法的 C 语言实现。

(1) 主要变量说明

$n$ : 矩阵数

$seq[]$ : 矩阵维数序列

$cost[][]$ : 二维数组, 长度为  $n*n$ , 其中元素  $cost[i][j]$  表示  $A_{i+1} * A_{i+2} * \dots * A_{j+1}$  的最优计算的计算代价

$trace[][]$ : 二维数组, 长度为  $n*n$ , 其中元素  $trace[i][j]$  表示  $A_{i+1} * A_{i+2} * \dots * A_{j+1}$  的最优计算对应的划分位置, 即  $k$

(2) 函数 `cmm`



$$\text{cost}[i][j] = \begin{cases} 0 & \text{if } i = j \\ \min_{i \leq k < j} \text{cost}[i][k] + \text{cost}[k+1][j] + p_i * p_{k+1} * p_{j+1} & \text{if } i < j \end{cases}$$

```

#define    N 100
int cost[N][N];
int trace[N][N];
int cmm(int n, int seq[]){
    int tempCost;
    int tempTrace;
    int i, j, k, p;
    int temp;
    for(i = 0; i < n; i++){ cost[i][i] = 0;    }
    for(p = 1; p < n; p++){
        for(i = 0; (1); i++){
            (2);
            tempCost = -1;
            for(k = i; k < j; k++){
                temp = (3);
                if(tempCost == -1 || tempCost > temp){
                    tempCost = temp;
                    (4);
                }
            }
            cost[i][j] = tempCost;
            trace[i][j] = tempTrace;
        }
    }
    return cost[0][n - 1];
}

```

**问题： 4.1**

根据以上说明和 C 代码，填充 C 代码中的空(1) ( 4)。

**问题： 4.2**

根据以上说明和 C 代码，该问题采用了(5) 算法设计策略，时间复杂度为(6) (用 O 符号表示)。

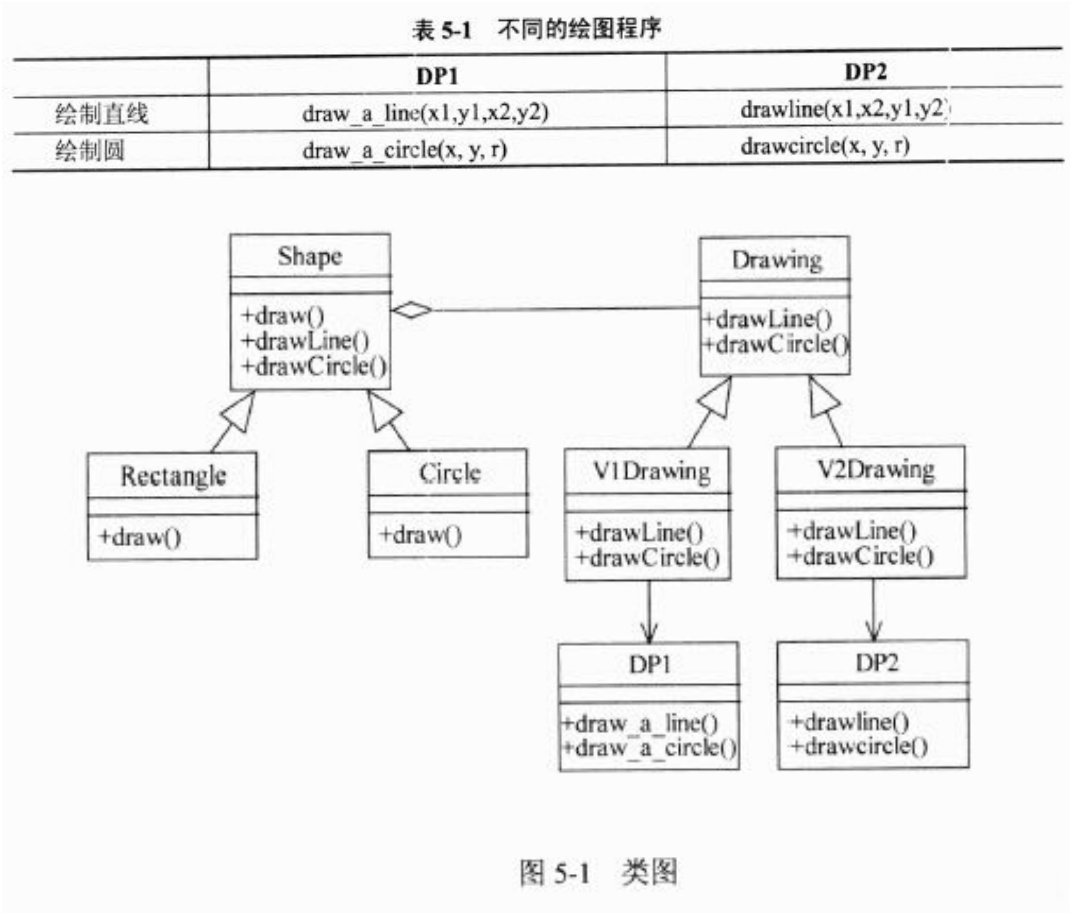
**问题： 4.3**

考虑实例 n=6, 各个矩阵的维数： A1 为 5\*10, A2 为 10\*3, A3 为 3\*12, A4 为 12\*5, A5 为 5\*50,

A6 为 50\*6，即维数序列为 5, 10, 3, 12, 5, 50, 6。则根据上述 C 代码得到的一个最优计算顺序为 (7) (用加括号方式表示计算顺序)，所需要的乘法运算次数为 (8)。

**试题五** 欲开发一个绘图软件，要求使用不同的绘图程序绘制不同的图形。以绘制直线和圆形为例，对应的绘图程序如表 5-1 所示。

该绘图软件的扩展性要求，将不断扩充新的图形和新的绘图程序。为了避免出现类爆炸的情况，现采用桥接(Bridge)模式来实现上述要求，得到如图 5-1 所示的类图。



### 【C++代码】

```
class DP1 {
public:
    static void draw_a_line(double x1, double y1, double x2, double y2) { /*
代码省略 */ }
    static void draw_a_circle(double x, double y, double r) { /* 代码省略
*/ }
};
class DP2 {
public:
    static void drawline(double x1, double x2, double y1, double y2) { /*
代码省略 */ }
    static void drawcircle(double x, double y, double r) { /* 代码省略 */ }
};
class Drawing {
public:
    (1);
    (2);
};
class V1Drawing : public Drawing {
public:
    void drawLine(double x1, double y1, double x2, double y2) { /* 代码
省略 */ }
    void drawCircle(double x, double y, double r) { (3); }
};
class V2Drawing : public Drawing {
public:
    void drawLine(double x1, double y1, double x2, double y2) { /* 代码
省略 */ }
    void drawCircle(double x, double y, double r) { (3); }
};
class V2Drawing : public Drawing {
public:
    void drawLine(double x1, double y1, double x2, double y2) { /* 代码
省略 */ }
    void drawCircle(double x, double y, double r) { (4); }
};
class Shape {
public:
    (5);
    Shape(Drawing *dp) { _dp = dp; }
    void drawLine(double x1, double y1, double x2, double y2) {
        _dp->drawLine(x1, y1, x2, y2); }
    void drawCircle(double x, double y, double r) { _dp->drawCircle(x, y,
r); }
private: Drawing *_dp;
};
```

```
class Rectangle : public Shape {
public:
    void draw() { /* 代码省略 */ }
    // 其余代码省略
};

class Circle : public Shape {
private: double _x, _y, _r;
public:
    Circle(Drawing *dp, double x, double y, double r) :__(6)___ { _x =
x; _y = y; _r = r; }
    void draw() { drawCircle(_x, _y, _r); }
};
```

问题： 5.1

阅读说明和 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 欲开发一个绘图软件，要求使用不同的绘图程序绘制不同的图形。以绘制直线和圆形为例，对应的绘图程序如表 6-1 所示。

该绘图软件的扩展性要求，将不断扩充新的图形和新的绘图程序。为了避免出现类爆炸的情况，现采用桥接(Bridge)模式来实现上述要求，得到如图 6-1 所示的类图。

表 6-1 不同的绘图程序		
	DP1	DP2
绘制直线	draw_a_line(x1,y1,x2,y2)	drawline(x1,x2,y1,y2)
绘制圆	draw_a_circle(x, y, r)	drawcircle(x, y, r)

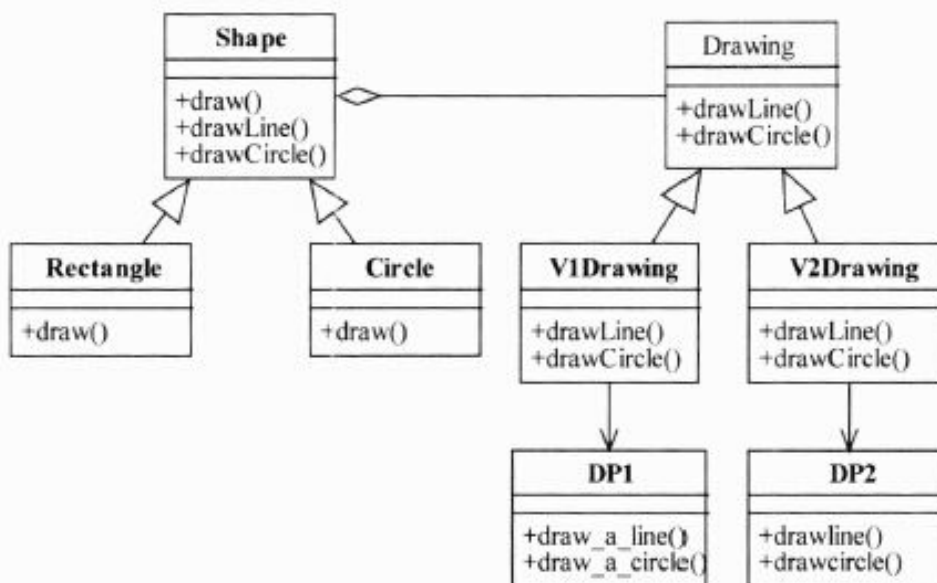


图 6-1 类图

【Java 代码】

```

(1)    Drawing {
(2)    ;
(3)    ;
}

class DP1{
    static public void draw_a_line(double x1, double y1, double x2, double
y2)
    { /*代码省略 */ }
    static public void draw_a_circle (double x, double y, double r) { /*
代码省略 */ }
}

class DP2{
    static public void drawline(double x1, double y1, double x2, double y2)
{ /*代码省略 */ }
    static public void drawcircle (double x, double y, double r) { /*代码
省略 */ }
}

class V1Drawing implements Drawing {
    public void drawLine(double x1, double y1, double x2, double y2) { /*
代码省略 */ }
}
  
```

```

        public void drawCircle(double x, double y, double r) {      (4)      ; }
    }

    class V2Drawing implements Drawing {
        public void drawLine(double x1, double y1, double x2, double y2) { /*
代码省略 */ }
        public void drawCircle(double x, double y, double r) {      (5)      ; }
    }

    abstract class Shape {
        private Drawing _dp;
        (6)      ;
        Shape(Drawing dp) { _dp = dp; }
        public void drawLine(double x1, double y1, double x2, double y2) {
            _dp.drawLine(x1, y1, x2, y2); }
        public void drawCircle(double x, double y, double r) { _dp.drawCircle(x,
y, r); }
    }

    class Rectangle extends Shape {
        private double _x1, _x2, _y1, _y2;
        public Rectangle (Drawing dp, double x1, double y1, double x2, double
y2)
        { /* 代码省略 */ }
        public void draw() { /* 代码省略 */ }
    }

    class Circle extends Shape {
        private double _x, _y, _r;
        public Circle(Drawing dp, double x, double y, double r) { /* 代码省略
*/ }
        public void draw() { drawCircle(_x, _y, _r); }
    }

```

### 问题： 6.1

阅读说明和 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：学生

E2：教务人员

本问题考查 0 层 DFD，要求确定外部实体。不难看出，在 0 层 DFD 中，系统主要功能“验证输入信息”和“处理注册申请”，涉及与系统交互的外部实体有“学生”提供输入信息，发送注册通知功能给“教务人员”发送所注册的课程信息和已注册的学生信息，从而即可确定 E1 为“学生”实体，E2 为“教务人员”实体。

D1：学生库

## D2：课程库

本问题要求确定 1 层数据流图中的数据存储。分析说明中和数据存储有关的描述，不难发现，说明 2. (1) 存储注册信息明确说明“将注册学生信息记录在学生库”，可知 D1 为学生库；说明 2. (2) 存储所注册课程中明确说明“然后存入课程库”，可知 D2 为课程库。

本问题要求补充缺失的数据流及其起点和终点。细心的考生可能会发现，对照图 1-1 和图 1-2 的输入数据流，数量和名称均相同，所以缺失的数据流是输出数据流或者处理之间的数据流。考查图 1-1 中输出至 E1 的数据流，有“接受提示”和“不合法提示”，而图 1-2 中没有这两条数据流，可以确定缺失的数据流包括这两条或者其分解的数据流。

考查说明 1 中的 3 个子功能，

1. (1) 检查学生信息完成检查学生输入的所有注册所需信息。如果信息不合法，返回学生信息不合法提示。

1. (2) 检查学位考试结果完成检查学生提供的学位考试结果。如果不合法，返回学位考试结果不合法提示。

1. (3) 检查学生注册资格完成根据合法学生信息和合法学位考试结果，检查该学生对欲选课程的注册资格。如果无资格，返回无注册资格提示。

对应图 1-1 中的处理 1 验证输入信息的输出数据流“不合法提示”，不难发现，在图 1-2 中，处理 1.1 缺少了到实体学生的输出数据流“学生信息不合法提示”；处理 1.2 缺少了到实体学生的输出数据流“无注册资格提示”；处理 1.3 缺少了到实体学生的输出数据流“学位考试结果不合法提示”。

再考查图 1-1 中处理 2，其输出数据流有三条，而图 1-2 中对图 1-1 中处理 2 的分解中，只包含了“所注册课程信息”和“已注册学生信息”两条数据流，缺失了“接受提示”。说明 2. (3) 中发送注册通知功能完成从学生库中读取注册学生信息，从课程库中读取所注册课程信息，给学生发送接受提示；给教务人员发送所注册课程信息和已注册学生信息。所以，缺失的“接受提示”的起点是处理 2.3 发送注册通知，终点是 E1 学生。

图 1-1 中不合法提示分解为图 1-2 中的三条数据流的组合：学生信息不合法提示、无注册资格提示、学位考试结果不合法提示。

图 1-1 中注册学生信息对应图 1-2 中注册学生信息和选课学生标识。

本问题考查数据流的分解与组合。仔细分析【说明】中的文字并与图 1-1 的对照，可以发现在图 1-1 中不合法提示在图 1-2 中没有出现。事实上，从前述【问题 3】缺失数据流的分析中，已经发现，图 1-2 中对于说明中的功能出现了“学生信息不合法提示”、“无注

册资格提示”和“学位考试结果不合法提示”三条数据流，说明图 1-1 中的数据流“不合法提示”是由这三条数据流组合而成。同样，2.(2) 存储所注册课程将选课学生标识与欲注册课程进行关联，然后存入课程库，图 1-1 中注册学生信息在图 1-2 中进一步分出注册学生信息和选课学生标识，即图 1-1 中注册学生信息是注册学生信息和选课学生标识的并集。

数 据 流	起 点	终 点
学生信息不合法提示	1.1 或 检查学生信息	E1 或 学生
无注册资格提示	1.3 或 检查学生注册资格	E1 或 学生
学位考试结果不合法提示	1.2 或 检查学位考试结果	E1 或 学生
接受提示	2.3 或 发送注册通知	E1 或 学生

**试题二 答案：** **解析：** 图中的\*可表示为 m 或 n，对联系名称可不作要求，但不能出现重名。

由“每个分公司有一位经理”可知分公司与经理之间的管理联系类型为 1:1;由“每个分公司有多名员工处理日常事务，每个员工属于一个分公司”可知分公司与员工间的所属联系类型为 1:\*;并且员工是经理的超类型，经理是员工的子类型。

由“一个客户可以有多个快件申请，但一个快件申请对应唯一的一个客户”可知，客户与申请单之间的提交联系类型为 1:\*。

由“业务员根据客户提交的快件申请单进行快件受理事宜，一个业务员可以受理多个客户的快件申请，一个快件申请只能由一个业务员受理”可知业务员与申请单之间的受理联系类型为 1:\*。

由“调度根据已受理的申请单安排快件的承运事宜，例如：执行承运的业务员、运达时间等；一个业务员可以执行调度安排的多个快件的承运业务。”可知，调度、业务员和申请单之间的承运联系类型为 1:1:\*。

- (1) (a) 分公司编号
- (b) 申请号，客户号
- (c) 申请号，业务员

(1) 完整的模式如下：

分公司(分公司编号，名称，办公电话，地址)

员工(员工号，姓名，分公司编号，岗位，薪资，手机号，家庭地址)

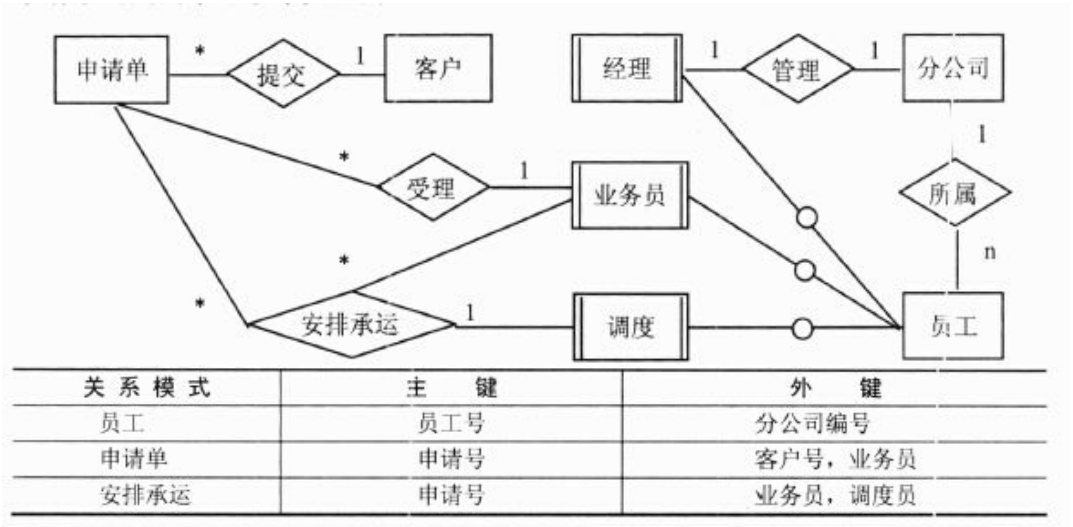


客户(客户号, 单位名称, 通信地址, 所属省份, 联系人, 联系电话, 银行账号)  
申请单(申请号, 客方号, 发件人, 发件人电话, 发件人地址, 快件名称, 运费, 收件人, 收件人电话, 收件地址, 受理标志, 业务员)  
安排承运(申请号, 业务员, 实际完成时间, 调度员)

(2) 员工、申请单和安排承运关系模式的主键和外键的分析如下：  
在申请单信息中，申请号由系统自动生成，不会重复，可作为申请单的主键属性，外键为客户号，业务员；在员工信息中，员工号唯一标识员工信息的每一个元组，故为员工关系的主键属性，外键为分公司编号；安排承运关系模式的主键为申请号，外键为业务员和调度员。

- (1) 该属性不属于简单属性。因为简单属性是原子的、不可再分的，复合属性是可以细分为更小的部分(即划分为别的属性)，本题客户关系的通信地址可以进一步分为邮编、省、市、街道，所以属于复合属性。
- (2) (d) 1:n
- (e) 分公司编号, 经理

- (1) 客户的通信地址属性不属于简单属性。因为根据题意，客户关系的通信地址可以进一步分为邮编、省、市、街道，而简单属性是原子的、不可再分的，复合属性可以细分为更小的部分(即划分为别的属性)。由于客户的通信地址可以进一步分为邮编、省、市、街道，故属于复合属性。
- (2) 根据题意，分公司需要增设一位经理的职位，那么分公司与经理之间的联系类型应修改为 1:n，分公司的主键应修改为分公司编号，经理。



**试题三 答案： 解析：** S1：普卡、普卡会员

S2：银卡、银卡会员

S3：金卡、金卡会员

T1: 25000<=里程数<50000

T2: 里程数>=50000

T3: 里程数>=50000

UML 中的状态图主要用于描述一个对象在其生存期间的动态行为，表现一个对象所经历的装填序列，引起状态转移的事件以及因状态转移而伴随的动作。图中给出的是会员的状态图。图中要求填充 S1、S2、S3 这三个状态以及它们之间的变迁关系。本题中会员有三种状态：普卡、金卡和银卡。根据说明，办理会员卡之后即可成为普卡会员，所以 S1 可以判定为普卡会员。当“里程数满 25,000 英里但不足 50,000 英里，则自动升级为银卡会员”，所以 S2 应为银卡会员，那么 S3 就应该是金卡会员。T1、T2 就是 S2 和 S3 之间的转换原则。T3 是 S1->S2 的转换原则。由说明可知，S2->S3(T2)：里程数在 50,000 英里以上；S3->S3(T1)：里程数达到 25,000 英里，但是不足 50,000 英里；S1->S3(T3)：累积的里程数在 50,000 英里以上。

C1：CNonMember

C2：CBasic

C3：CSilver

C4：CGold

(C1 C4 的次序可以互换)

由图 3-2 可知，需要补充的是继承结构中的子类。根据题目说明，能够具有一般/特殊关系的只有不同级别的会员。所以 C1~C4 依次应该是：CNonMember、CBasic、CSilver、CGold。

使用了 State 模式(状态模式)。

类 CFrequentFlyer 必须具有的属性：CLevel 的对象。

travel 方法的功能：计算飞行里程数，根据里程数判断是否需要调整会员级别(跳转到不同的状态)。

本题在设计类时使用到了状态模式。

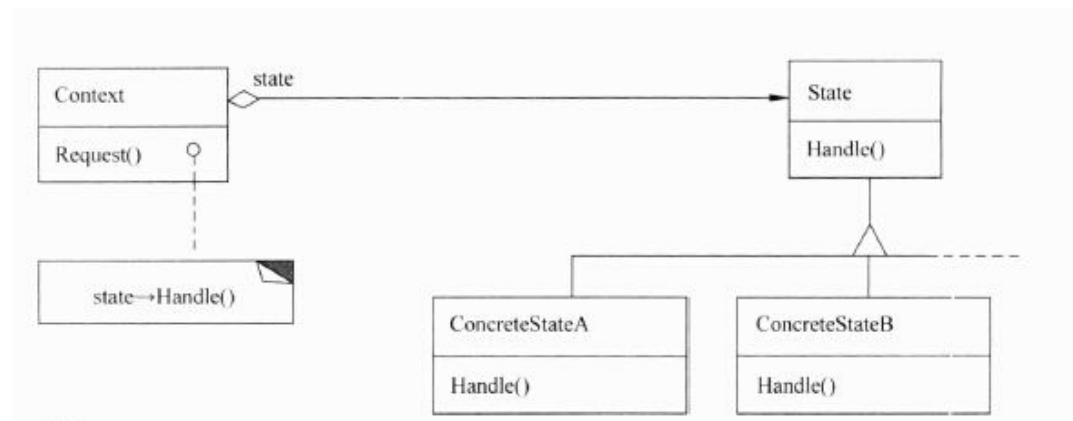
状态模式允许对象在内部状态变化时，变更其行为，并且修改其类。状态模式的类图如下所示。

其中：

- 环境类(Context):定义客户感兴趣的接口。维护一个 ConcreteState 子类的实例，这个实例定义当前状态。
- 抽象状态类(State):定义一个接口以封装与 Context 的一个特定状态相关的行为。
- 具体状态类(ConcreteState):每一子类实现一“与 Context 的一个状态相关的行为。

图 3-2 中的类 CFrequentFlyer 对应上图中的环境类，因此类 CFrequentFlyer 应该有一个 CLevel 类的对象。

travel 方法的功能：计算飞行里程数，根据里程数判断是否需要调整会员级别(跳转到不同的状态)。



试题四 答案： 解析： (1) i

(2)  $j = i + p$

(3)  $\text{cost}[i][k] + \text{cost}[k+1][j] + \text{seq}[i] * \text{seq}[k+1] * \text{seq}[j+1]$

(4)  $\text{tempTrace} = k$

本问题考查算法的实现。C 程序中主要部分是三重循环，循环变量  $p$  定义了求解问题的规模，因为是从底向上，因此， $p$  的值应该是从 1 到  $n-1$ ，即从规模为 1 的问题一直到规模为  $n-1$  的问题。循环变量  $i$  是要求解的子问题的起始，从 0 开始，最大为  $n-p-1$ ，故(1)处应填  $n-p$ 。确定了  $i$  和  $p$  之后，下来就要确定  $j$  了，显然，空(2)处为  $j = i + p$ 。循环变量  $k$  是问题  $A_i * A_{i+1} * \dots * A_j$  的划分位置，对每一个  $k$ ，都要计算需要的计算成本，可以根据递归式来填写，空(3)处为  $\text{cost}[i][k] + \text{cost}[k+1][j] + \text{seq}[i] * \text{seq}[k+1] * \text{seq}[j+1]$ 。确定每个问题  $A_i * A_{i+1} * \dots * A_j$  的划分位置  $k$  之后，要把这个  $k$  值记住，放在变量

tempTrace 中，即空(4)处填写 tempTrace=k。

(5) 动态规划

(6)  $O(n^3)$

本问题考查算法的设计策略和时间复杂度，从题干说明可以很容易看出，问题具有最优子结构和重叠子问题，采用自底向上的方法求解，这些都是动态规划的典型特点，因此采用的是动态规划设计策略。从上述 C 程序很容易分析出，程序中没有递归，存在三重循环，故时间复杂度为  $O(n^3)$ 。

(7) ((A1A2)((A3A4)(A5A6)))

(8) 2010

本问题考查算法的应用。通过一个具体实例可以更容易理解问题和求解方法。可以根据问题 1 中的程序执行来求解。启发式的思路是先把维度最大的消掉，如 A<sub>5</sub>\*A<sub>6</sub> 相乘之后，维度 50 就没有了，所以考虑这两个矩阵先相乘；然后是 A<sub>3</sub>\*A<sub>4</sub> 相乘之后，维度 12 就没有了，所以考虑这两个矩阵相乘；接着，A<sub>1</sub>\*A<sub>2</sub> 相乘之后，维度 10 就没有了，所以考虑这两个矩阵相乘 这样可以确定相乘的顺序 ((A1A2)((A3A4)(A5A6)))，需要的计算开销分别是  $5*50*6=1500$ ,  $3*12*5=180$ ,  $5*10*3=150$ ,  $3*5*6=90$ ， $5*3*6=90$ ，把上述值相加，即  $1500+180+150+90+90=2010$ 。

试题五 答案： 解析： (1) virtual void draw Line

(double x1, double y1, double x2, double y2) = 0

(2) virtual void draw Circle (double x, double y, double r) = 0

(3) DP1::draw\_a\_circle(x, y, r)

(4) DP2::draw\_circle(x, y, r)

(5) virtual void draw() =0

(6) Shape(dp)

本题考查桥接(Bridge)模式的概念及应用。

Bridge 模式可以将复杂的组件分成两个独立的、

的抽象和内部实现。改变组件的这两个层次结构很简单，以至于它们可以相互独立地变

化。当具有抽象的层次结构和相应的实现层次结构时，Bridge 模式是非常有用的。除了可以将抽象和实现组合成许多不同的类，该模式还可以动态组合的独立类的形式实现这些抽象和实现。下图所示是 Bridge 模式的类图。

在以下情况中，应该使用 Bridge 模式：

- 想避免在抽象及其实现之间存在永久的绑定；
- 抽象及其实现可以使用子类进行扩展；
- 抽象的实现被改动应该对客户端没有影响，也就是说，不需要重新编译代码。

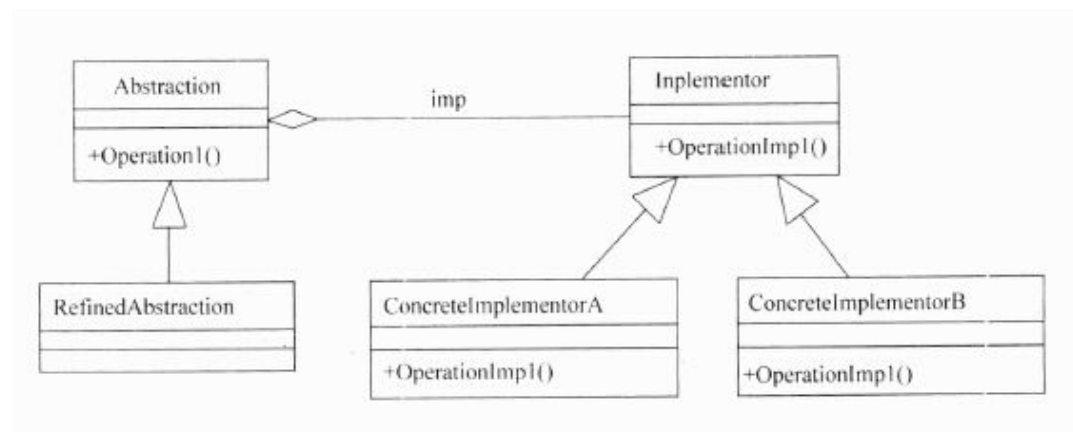
本题中，类 Shape 对应上图中的 Abstraction，表示抽象部分；类 Drawing 对应

Implementor，表示实现部分。这两个类的子类分别表示具体的抽象部分和实现部分。在 C++ 中，Drawing 可以用抽象类来实现，将其中的方法定义为纯虚拟函数。因此 (1)、(2) 分别应为 “virtual void drawLine(double x1, double y1, double x2, double y2)=0”、’ ‘  
virtual void drawCircle(double x, double y, double r)=0”。

VDrawing 是绘图实现类之一，它采用的绘图程序由是 DPI 所提供的。DPI 中的方法均为静态方法，必须用类名来引用。因此 (3) 处应为 DPI::draw\_\_a\_circle(x, y, r)。同理 (4) 处应为 “DP2::drawcircle(x, y, r)”。

由类图可以看出，Shape 类中定义的方法 draw 在其子类中被重置了，而 Shape 表示的是抽象部分，可以将 draw 方法定义为纯虚拟函数。所以，(5) 应该为 “virtual void draw() = 0”。

空 (6) 处考查继承结构中子类构造函数的定义。构造子类对象时，需要调用基类的构造函数，这可以通过初始化列表显式指明需要调用的基类的构造函数。在本题中，Shape 类只定义了一个构造函数，因此 (6) 应该为 “Shape(dp)”。



试题六 答案： 解析： (1) interface

(2) void drawLine(double x1, double y1, double x2, double y2)

(3) void drawCircle(double x, double y, double r)

(4) DPI.draw\_a\_circle(x, y, r)

(5) DP2.draw\_circle(x, y, r)

(6) public abstract void draw()

本题考查桥接(Bridge)模式的概念及应用。

Bridge 模式可以将复杂的组件分成两个独立的但又相关的继承层次结构：功能性的抽象和内部实现。改变组件的这两个层次结构很简单，以至于它们可以相互独立地变化。当具有抽象的层次结构和相应的实现层次结构时，Bridge 模式是非常有用的。除了可以将抽象和实现组合成许多不同的类，该模式还可以以动态组合的独立类的形式实现这些抽象和实现。下图所示是 Bridge 模式的类图。

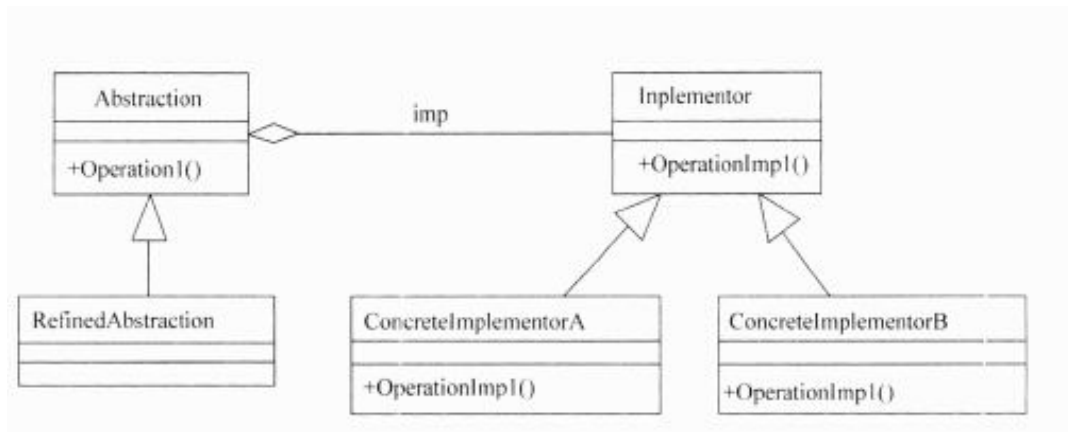
在以下情况中，应该使用 Bridge 模式：

- 想避免在抽象及其实现之间存在永久的绑定；
- 抽象及其实现可以使用子类进行扩展；
- 抽象的实现被改动应该对客户端没有影响，也就是说，不需要重新编译代码。

本题中，类 Shape 对应上图中的 Abstraction, 表示抽象部分；类 Drawing 对应 Implementor, 表示实现部分。这两个类的子类分别表示具体的抽象部分和实现部分。类 Drawing 为具体的实现类提供统一接口，在 Java 中可以使用接口来实现。因此(1)、(2)、(3) 分别应为“interface”、“void drawLine (double x1, double y1, double x2, double y2)”、“void drawCircle (double x, double y, double r)”。

VI Drawing 是绘图实现类之一，它采用的绘图程序由是 DPI 所提供的。3 此(4) 处应为 “DPI.draw\_a\_circle(x, y, r)” 。同理(5) 处应为 “DP2.draw\_circle(x, y, r)”。

由类图可以看出，Shape 类中定义的方法 draw 在其子类中被重置了，而 Shape 表示的是抽象部分，可以将 draw 方法定义为抽象函数。所以，(6) 应该为 “abstract public void draw()”。



苹果 扫码或应用市场搜索“软考  
真题”下载获取更多试卷



安卓 扫码或应用市场搜索“软考  
真题”下载获取更多试卷