

希赛网, 专注于软考、PMP、通信考试的专业 IT 知识库和在线教育平台。希赛网在线题库, 提供历年考试真题、模拟试题、章节练习、知识点练习、错题本练习等在线做题服务, 更有能力评估报告, 让你告别盲目做题, 针对性地攻破自己的薄弱点, 更高效的备考。

希赛网官网: <http://www.educity.cn/>

希赛网软件水平考试网: <http://www.educity.cn/rk/>

希赛网在线题库: <http://www.educity.cn/tiku/>

2013 年下半年软设案例分析真题答案与解析: <http://www.educity.cn/tiku/tp1423.html>

2013 年下半年软件设计师考试下午真题 (参考答案)

- 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) 检查学位考试信息: 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 and 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。

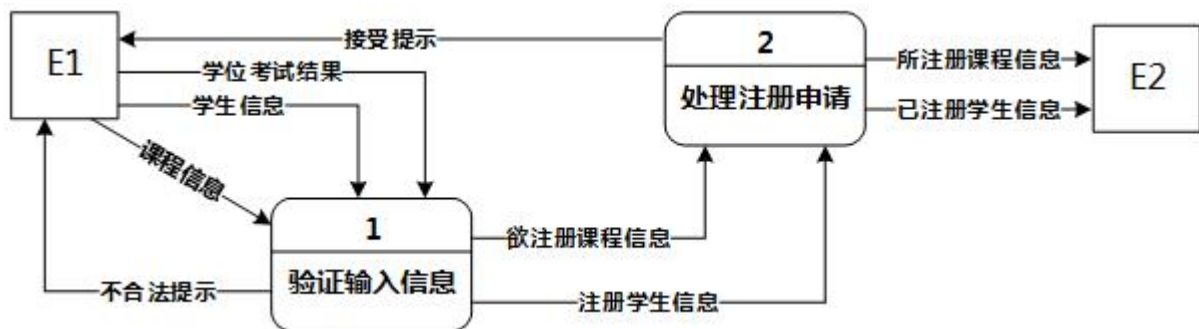


图 1-1 0 层数据流

图

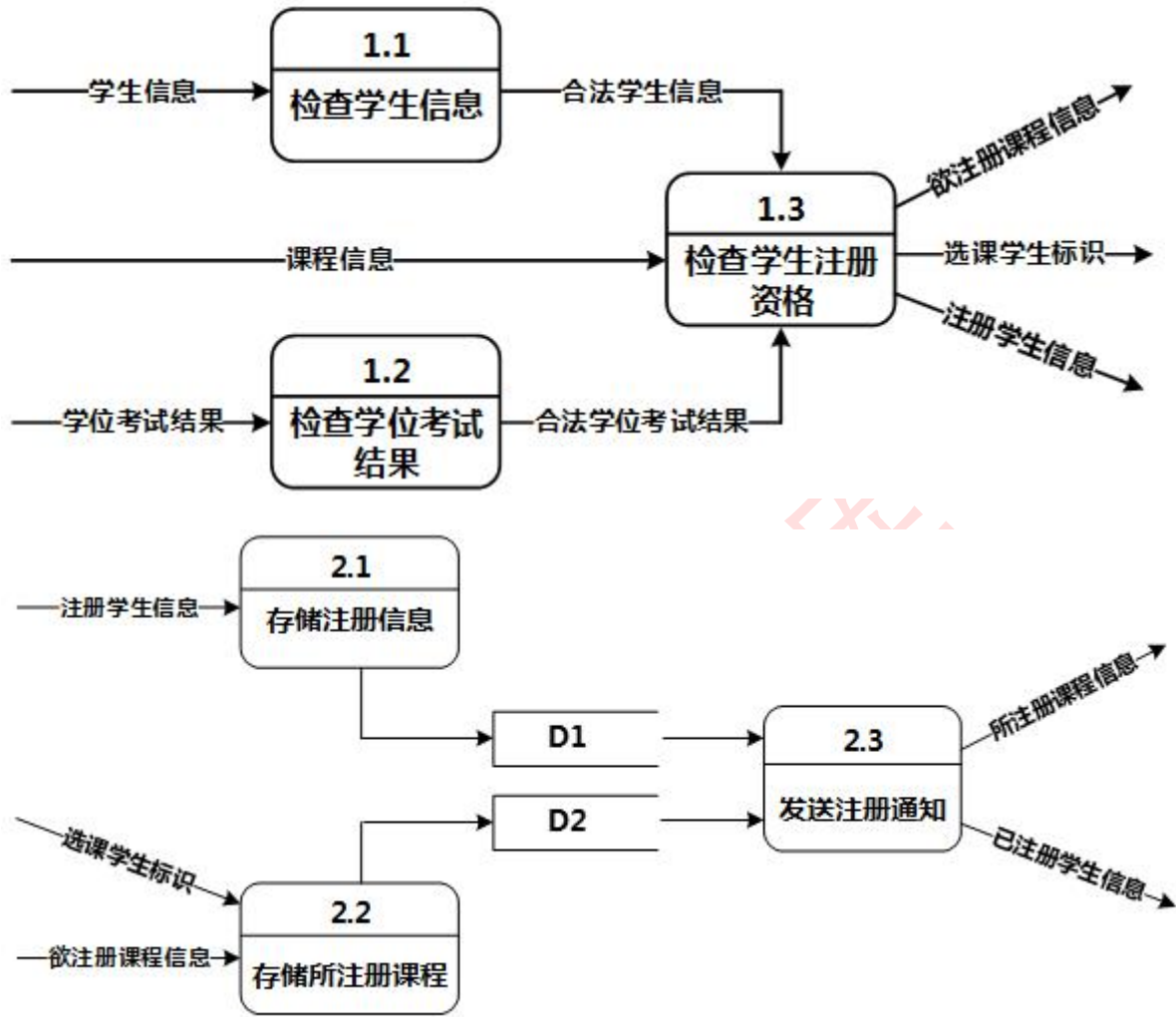


图 1-2 1 层数据流图

【问题 1】(2 分)

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 2】(2 分)

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 3】(8 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。

- 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名员工处理分公司的日常业务, 每位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、运送时间等。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) 和关系模型 (不完整) 如下:

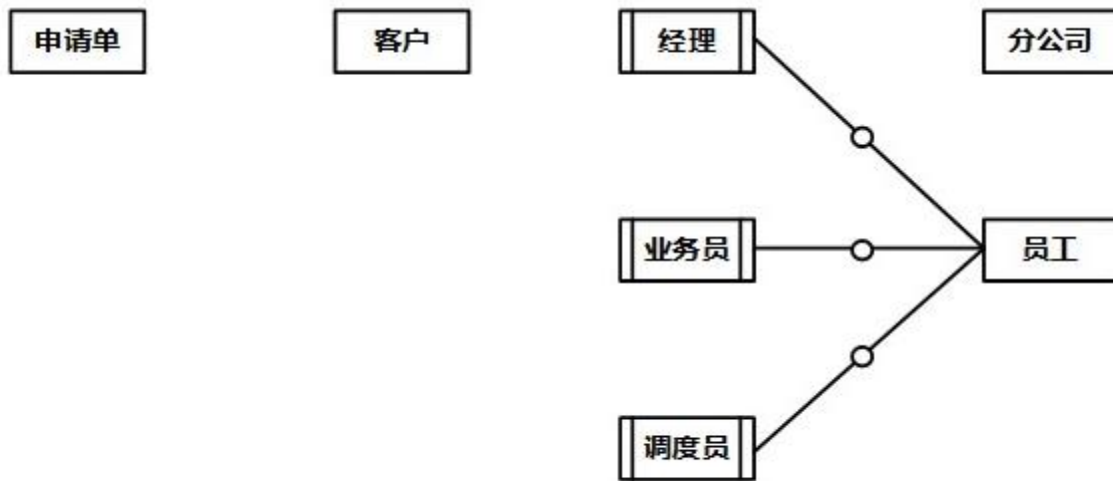


图 2-1 实体联系图

【关系模型设计】

- 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 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】

- (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 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)。

● 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) 不能积累里程数。

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 25000 英里但不足 50000 英里, 则自动升级为银卡会员; 若累积的里程数在 50000 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 50000 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 25000 英里, 则自动降级为普卡会员; 若累积的里程数达到 25000 英里, 但是不足 50000 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 25000 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。

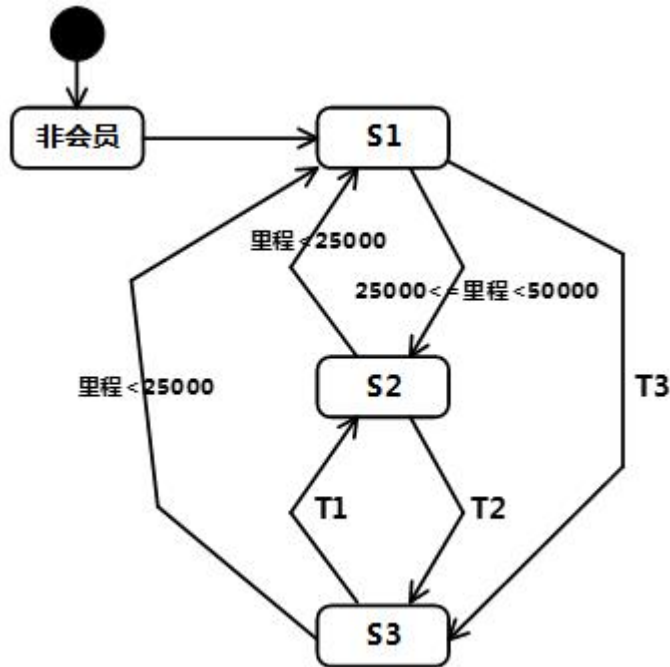


图 3-1 状态

图

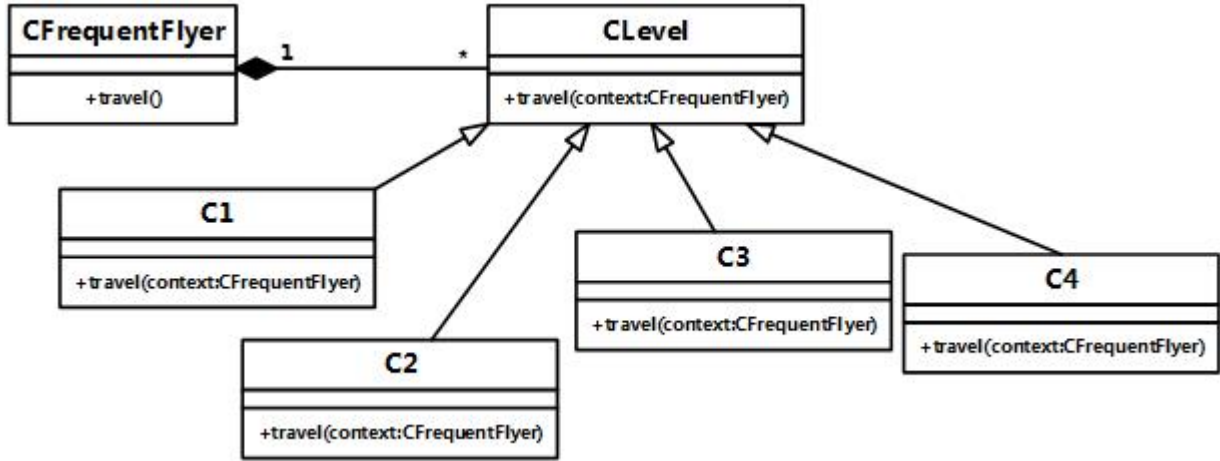


图 3-2 的类图

【问题 1】 (6分)

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 2】 (4分)

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】 (5分)

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?

• 阅读下列说明和 C 代码, 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 $A_{10 \times 100}$, $A_{200 \times 5}$, $A_{5 \times 50}$ 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 $\langle A_1, A_2, \dots, A_n \rangle$, 矩阵 A_i 的维数为 $p_{i-1} \times p_i$, 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,

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

其中, $\text{cost}[i][j]$ 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 $\text{cost}[0][n-1]$ 。

【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 $n \times n$, 其中元素 $cost[i][j]$ 表示 $A_{i+1} \times A_{i+2} \times \dots \times A_{j+1}$ 的最优计算的计算代价

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

(2) 函数 cmm

```
#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 1】 (8 分)

根据以上说明和 C 代码, 填充 C 代码中的空 (1) ~ (4)。

【问题 2】 (4 分)

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 (6)。

(用 O 符号表示)

【问题 3】 (3 分)

考虑实例 $n=6$, 各个矩阵的维数: A_1 为 5×10 , A_2 为 10×3 , A_3 为 3×12 , A_4 为 12×5 , A_5 为 5×50 , A_6 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)。

•

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】

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。

表 5-1 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)

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。

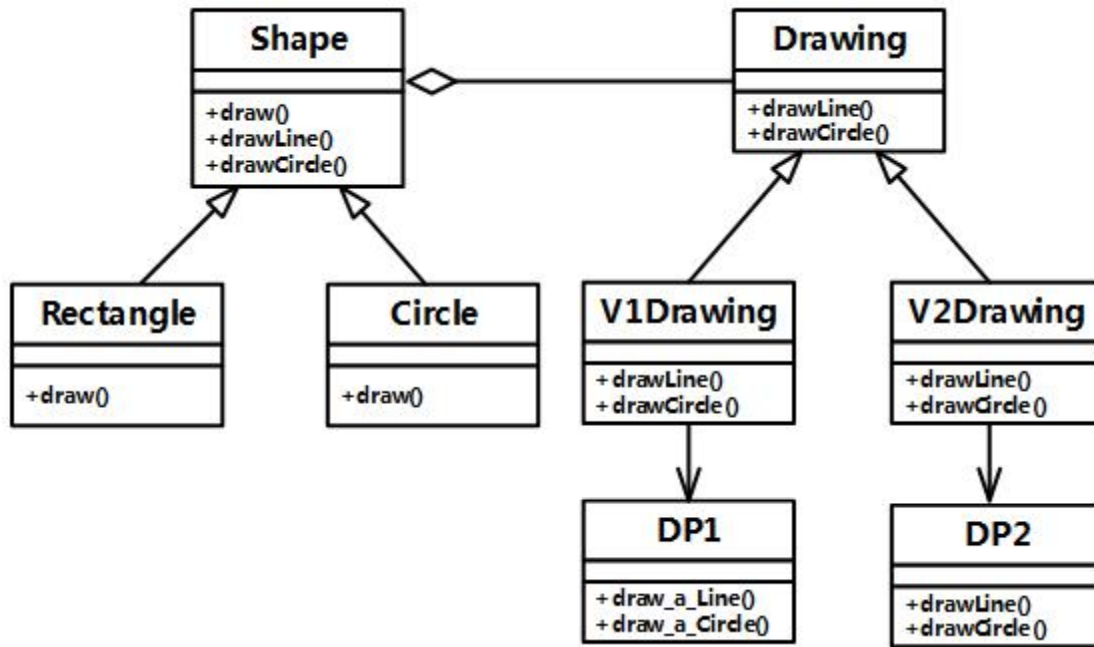


图 5-1 类图 【C++代码】

```

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){__(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__(17)__/*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__(18)__ { drawCircle(_x, _y, _r);}
};

```

- 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】

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。

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)

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。

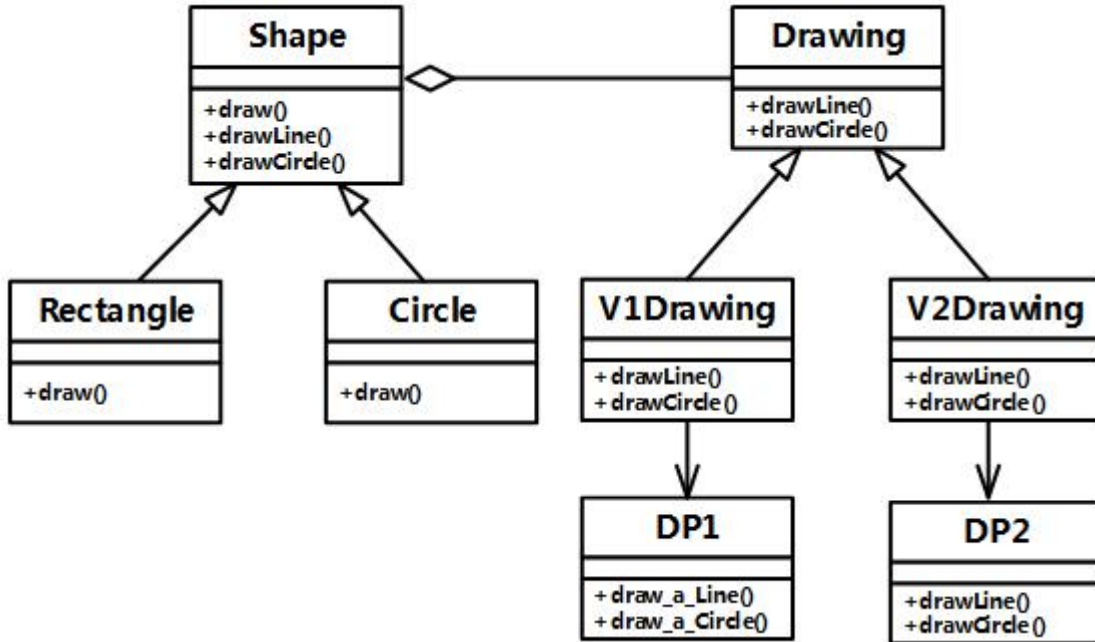


图 6-1 类图

【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__(21)__ { /*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__(22)__ {drawCircle(_x,_y,_r);}
}
```

希赛网在线题库