

# 全国计算机技术与软件专业技术资格（水平）考试

## 2007 年下半年 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，试题五至试题七选答 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

2007 年下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“11 月 3 日”，故在答题纸的对应栏内写上“11”和“3”（参看下表）。

例题	解答栏
(1)	11
(2)	3

### 试题一（共 15 分）

阅读以下说明和图，回答问题1至问题4，将解答填入答题纸的对应栏内。

#### 【说明】

某高校欲开发一个成绩管理系统，记录并管理所有选修课程的学生的平时成绩和考试成绩，其主要功能描述如下：

1. 每门课程都有 3 到 6 个单元构成，每个单元结束后会进行一次测试，其成绩作为这门课程的平时成绩。课程结束后进行期末考试，其成绩作为这门课程的考试成绩。
2. 学生的平时成绩和考试成绩均由每门课程的主讲教师上传给成绩管理系统。
3. 在记录学生成绩之前，系统需要验证这些成绩是否有效。首先，根据学生信息文件来确认该学生是否选修这门课程，若没有，那么这些成绩是无效的；如果他的确选修了这门课程，再根据课程信息文件和课程单元信息文件来验证平时成绩是否与这门课程所包含的单元相对应，如果是，那么这些成绩是有效的，否则无效。
4. 对于有效成绩，系统将其保存在课程成绩文件中。对于无效成绩，系统会单独将其保存在无效成绩文件中，并将详细情况提交给教务处。在教务处没有给出具体处理意见之前，系统不会处理这些成绩。
5. 若一门课程的所有有效的平时成绩和考试成绩都已经被系统记录，系统会发送课程完成通知给教务处，告知该门课程的成绩已经齐全。教务处根据需要，请求系统生成相应的成绩列表，用来提交考试委员会审查。
6. 在生成成绩列表之前，系统会生成一份成绩报告给主讲教师，以便核对是否存在错误。主讲教师须将核对之后的成绩报告返还系统。
7. 根据主讲教师核对后的成绩报告，系统生成相应的成绩列表，递交考试委员会进行审查。考试委员会在审查之后，上交一份成绩审查结果给系统。对于所有通过审查的成绩，系统将会生成最终的成绩单，并通知每个选课学生。

现采用结构化方法对这个系统进行分析与设计，得到如图 1-1 所示的顶层数据流图和图 1-2 所示的 0 层数据流图。

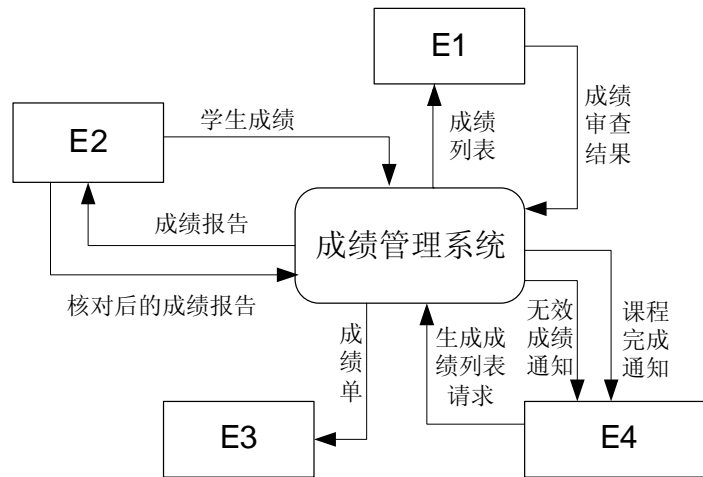


图 1-1 顶层数据流图

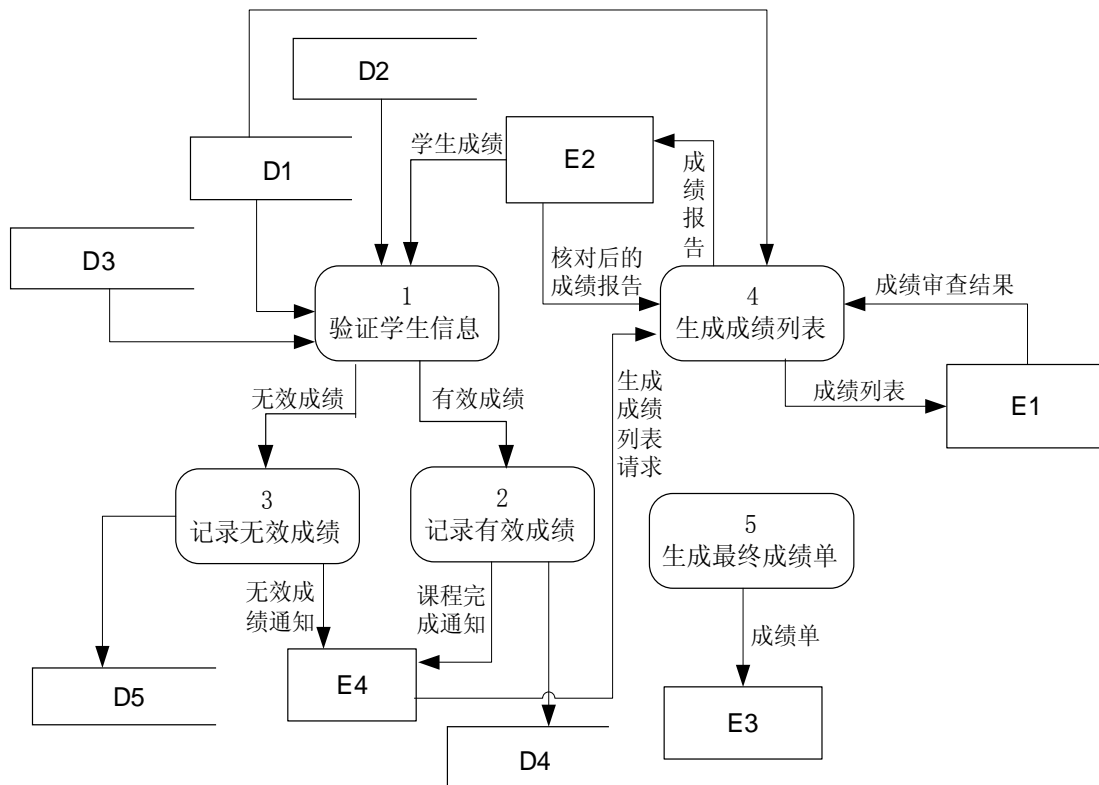


图 1-2 0 层数据流图

**【问题 1】(4 分)**

使用说明中的词语，给出图 1-1 中的外部实体 E1~E4 的名称。

**【问题 2】(3 分)**

使用说明中的词语，给出图 1-2 中的数据存储 D1~D5 的名称。

**【问题 3】(6 分)**

数据流图 1-2 缺少了三条数据流，根据说明及数据流图 1-1 提供的信息，分别指出这三条数据流的起点和终点。

起点	终点

**【问题 4】(2 分)**

数据流图是在系统分析与总体设计阶段宏观地描述系统功能需求的重要图形化工具，程序流程图也是软件开发过程中比较常用的图形化工具。简要说明程序流程图的适用场合与作用。

## 试题二（15分）

阅读下列说明，回答问题1至问题4，将解答填入答题纸的对应栏内。

### 【说明】

某汽车维修站拟开发一套小型汽车维修管理系统，对车辆的维修情况进行管理。

1. 对于新客户及车辆，汽车维修管理系统首先登记客户信息，包括：客户编号、客户名称、客户性质（个人、单位）、折扣率、联系人、联系电话等信息；还要记录客户的车辆信息，包括：车牌号、车型、颜色等信息。一个客户至少有一台车。客户及车辆信息如表2-1所示。

表 2-1 客户及车辆信息

客户编号	GS0051	客户名称	××公司	客户性质	单位
折扣率	95%	联系人	杨浩东	联系电话	82638779
车牌号	**0761	颜色	白色	车型	帕萨特
				车辆类型	微型车

2. 记录维修车辆的故障信息。包括：维修类型（普通、加急）、作业分类（大、中、小修）、结算方式（自付、三包、索赔）等信息。维修厂的员工分为：维修员和业务员。车辆维修首先委托给业务员。业务员对车辆进行检查和故障分析后，与客户磋商，确定故障现象，生成维修委托书。如表2-2所示。

表 2-2 维修委托书

No.20070702003

登记日期：2007-07-02

车牌号	**0765	客户编号	GS0051	维修类型	普通
作业分类	中修	结算方式	自付	进厂时间	20070702 11:09
业务员	张小江	业务员编号	012	预计完工时间	
故障描述					
车头损坏，水箱漏水					

3. 维修车间根据维修委托书和车辆的故障现象，在已有的维修项目中选择并确定一个或多个具体维修项目，安排相关的维修工及工时，生成维修派工单。维修派工单如表2-3所示。

表 2-3 维修派工单

No.20070702003

维修项目编号	维修项目	工时	维修工编号	维修工工种
12	维修车头	5.00	02	机修
12	维修车头	2.00	03	漆工
15	水箱焊接补漏	1.00	06	焊工
17	更换车灯	1.00	02	机修

4. 客户车辆在车间修理完毕后，根据维修项目单价和维修派工单中的工时计算车辆此

次维修的总费用，记录在委托书中。

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）如下所示。图 2-1 中业务员和维修工是员工的子实体。

**【概念结构设计】**

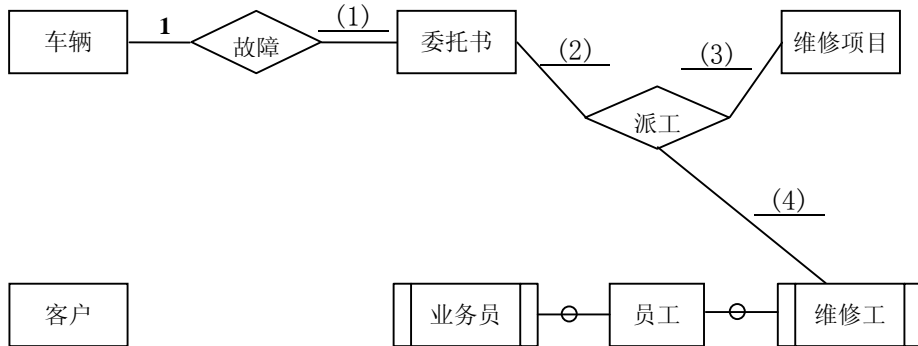


图 2-1 实体联系图

**【逻辑结构设计】**

客户(\_\_\_\_\_(5)\_\_\_\_\_, 折扣率, 联系人, 联系电话)

车辆(车牌号, 客户编号, 车型, 颜色, 车辆类别)

委托书(\_\_\_\_\_(6)\_\_\_\_\_, 维修类型, 作业分类, 结算方式, 进厂时间, 预计完工时间, 登记日期, 故障描述, 总费用)

维修项目(维修项目编号, 维修项目, 单价)

派工单(\_\_\_\_\_(7)\_\_\_\_\_, 工时)

员工(\_\_\_\_\_(8)\_\_\_\_\_, 工种, 员工类型, 级别)

**【问题 1】(4 分)**

根据问题描述，填写图 2-1 中(1)~(4)处联系的类型。联系类型分为一对一、一对多和多对多三种，分别使用 1:1, 1:n 或 1:\*, m:n 或 \*: \*表示。

**【问题 2】(4 分)**

补充图 2-1 中的联系并指明其联系类型。联系名可为：联系 1, 联系 2, …。

**【问题 3】(4 分)**

根据图 2-1 和说明，将逻辑结构设计阶段生成的关系模式中的空(5)~(8)补充完整。

**【问题 4】(3 分)**

根据问题描述，写出客户、委托书和派工单这三个关系的主键。

**试题三（共 15 分）**

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】**

已知某唱片播放器不仅可以播放唱片，而且可以连接电脑并把电脑中的歌曲刻录到唱片上（同步歌曲）。连接电脑的过程中还可自动完成充电。

关于唱片，还有以下描述信息：

1. 每首歌曲的描述信息包括：歌曲的名字、谱写这首歌曲的艺术家以及演奏这首歌曲的艺术家。只有两首歌曲的这三部分信息完全相同时，才认为它们是同一首歌曲。艺术家可能是一名歌手或一支由 2 名或 2 名以上的歌手所组成的乐队。一名歌手可以不属于任何乐队，也可以属于一个或多个乐队。

2. 每张唱片由多条音轨构成；一条音轨中只包含一首歌曲或为空白，一首歌曲可分布在多条音轨上；同一首歌曲在一张唱片中最多只能出现一次。

3. 每条音轨都有一个开始位置和持续时间。一张唱片上音轨的次序是非常重要的，因此对于任意一条音轨，播放器需要准确地知道，它的下一条音轨和上一条音轨是什么（如果存在的话）。

根据上述描述，采用面向对象方法对其进行分析与设计，得到了如表 3-1 所示的类列表、如图 3-1 所示的初始类图以及如图 3-2 所示的描述播放器行为的 UML 状态图。

表 3-1 类列表

类名	说明
Artist	艺术家
Song	歌曲
Band	乐队
Musician	歌手
Track	音轨
Album	唱片

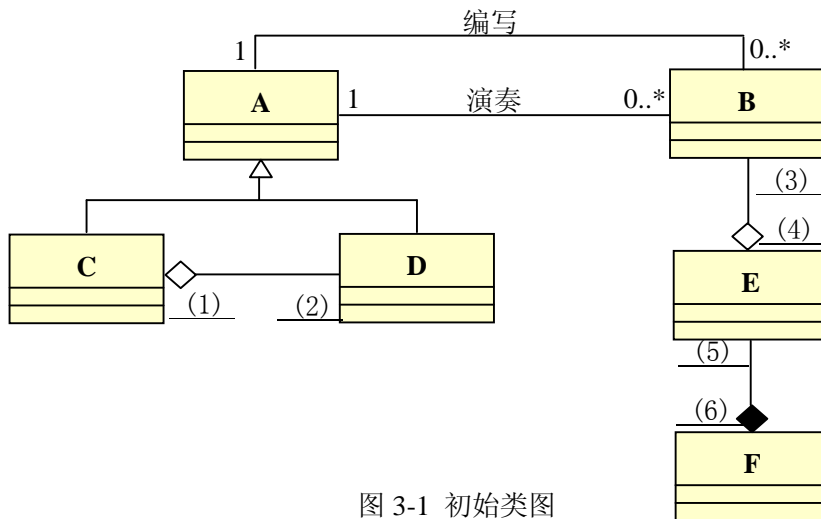


图 3-1 初始类图

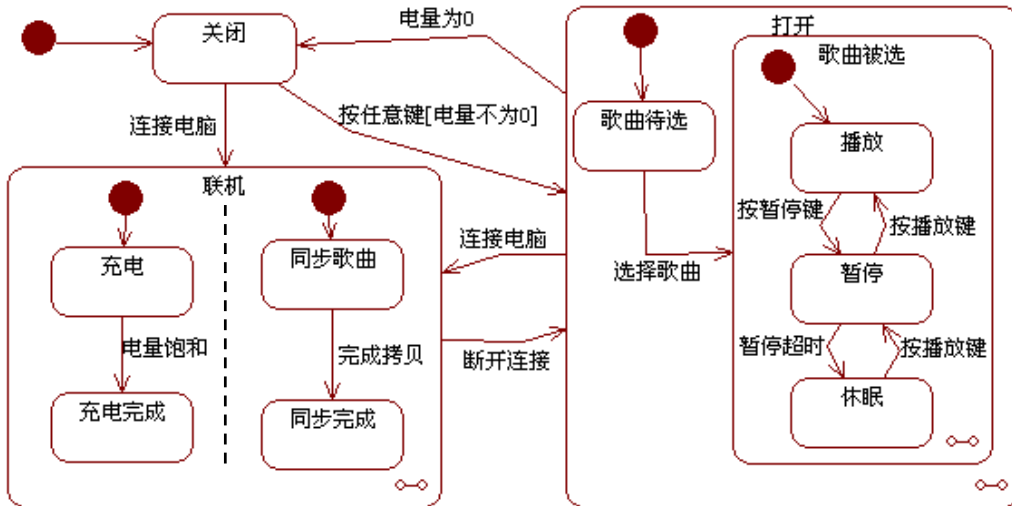


图 3-2 播放器行为 UML 状态图

**【问题 1】(3 分)**

根据说明中的描述，使用表 3-1 给出的类的名称，给出图 3-1 中的 A~F 所对应的类。

**【问题 2】(6 分)**

根据说明中的描述，给出图 3-1 中 (1) ~ (6) 处的多重度。

**【问题 3】(4 分)**

图 3-1 中缺少了一条关联，请指出这条关联两端所对应的类以及每一端的多重度。

类	多重度

**【问题 4】(2 分)**

根据图 3-2 所示的播放器行为 UML 状态图，给出从“关闭”状态到“播放”状态所经过的最短事件序列（假设电池一开始就是有电的）。

#### 试题四(共 15 分)

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】

某机器上需要处理  $n$  个作业  $job_1, job_2, \dots, job_n$ ，其中：

- (1) 每个作业  $job_i (1 \leq i \leq n)$  的编号为  $i$ ， $job_i$  有一个收益值  $p[i]$  和最后期限值  $d[i]$ ；
- (2) 机器在一个时刻只能处理一个作业，而且每个作业需要一个单位时间进行处理，一旦作业开始就不可中断，每个作业的最后期限值为单位时间的正整数倍；
- (3)  $job_1 \sim job_n$  的收益值呈非递增顺序排列，即  $p[1] \geq p[2] \geq \dots \geq p[n]$ ；
- (4) 如果作业  $job_i$  在其期限之内完成，则获得收益  $p[i]$ ；如果在其期限之后完成，则没有收益。

为获得较高的收益，采用贪心策略求解在期限之内完成的作业序列。图 4-1 是基于贪心策略求解该问题的流程图。

(1) 整型数组  $J[]$  有  $n$  个存储单元，变量  $k$  表示在期限之内完成的作业数， $J[1..k]$  存储所有能够在期限内完成的作业编号，数组  $J[1..k]$  里的作业按其最后期限非递减排序，即  $d[J[1]] \leq \dots \leq d[J[k]]$ 。

(2) 为了方便于在数组  $J$  中加入作业，增加一个虚拟作业  $job_0$ ，并令  $d[0] = 0$ ， $J[0] = 0$ 。

(3) 算法大致思想：先将作业  $job_1$  的编号 1 放入  $J[1]$ ，然后，依次对每个作业  $job_i (2 \leq i \leq n)$  进行判定，看其能否插入到数组  $J$  中，若能，则将其编号插入到数组  $J$  的适当位置，并保证  $J$  中作业按其最后期限非递减排列，否则不插入。

$job_i$  能插入数组  $J$  的充要条件是： $job_i$  和数组  $J$  中已有作业均能在其期限之内完成。

(4) 流程图中的主要变量说明如下：

- $i$ ：循环控制变量，表示作业的编号；
- $k$ ：表示在期限内完成的作业数；
- $r$ ：若  $job_i$  能插入数组  $J$ ，则其在数组  $J$  中的位置为  $r+1$ ；
- $q$ ：循环控制变量，用于移动数组  $J$  中的元素。

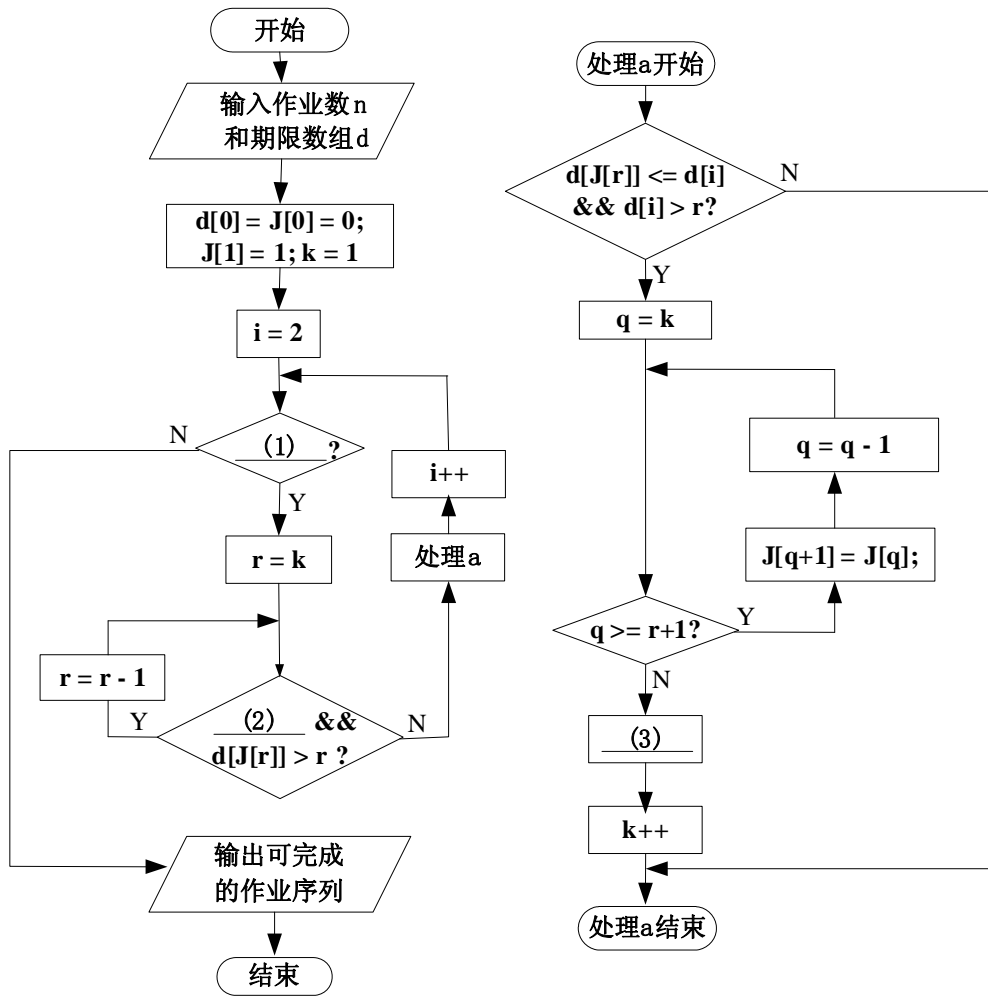


图 4-1 贪心策略流程图

**【问题 1】(9 分)**

请填充图 4-1 中的空缺(1)、(2)和(3)处。

**【问题 2】(4 分)**

假设有 6 个作业  $job_1, job_2, \dots, job_6$ ;

完成作业的收益数组  $p=(p[1], p[2], p[3], p[4], p[5], p[6]) = (90, 80, 50, 30, 20, 10)$ ;

每个作业的处理期限数组  $d=(d[1], d[2], d[3], d[4], d[5], d[6]) = (1, 2, 1, 3, 4, 3)$ 。

请应用试题中描述的贪心策略算法，给出在期限之内处理的作业编号序列\_\_\_\_(4)\_\_\_\_  
(按作业处理的顺序给出)，得到的总收益为\_\_\_\_(5)\_\_\_\_。

**【问题 3】(2 分)**

对于本题的作业处理问题，用图 4-1 的贪心算法策略，能否求得最高收益？\_\_\_\_(6)\_\_\_\_。

用贪心算法求解任意给定问题时，是否一定能得到最优解？\_\_\_\_(7)\_\_\_\_。

从下列 3 道试题（试题五至试题七）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（共 15 分）

阅读以下说明和 C 代码，将应填入     (n)     处的字句写在答题纸的对应栏内。

#### 【说明】

在一个简化的绘图程序中，支持的图形种类有点(point)和圆(circle)，在设计过程中采用面向对象思想，认为所有的点和圆都是一种图形(shape)，并定义了类型 shape\_t、point\_t 和 circle\_t 分别表示基本图形、点和圆，并且点和圆具有基本图形的所有特征。

#### [C 代码]

```
typedef enum { point,circle } shape_type; /* 程序中的两种图形：点和圆 */
typedef struct {
    shape_type type; /* 图形种类标识：点或者圆 */
    void (*destroy)(); /* 销毁图形操作的函数指针 */
    void (*draw)(); /* 绘制图形操作的函数指针 */
} shape_t;
typedef struct { shape_t common; int x; int y; } point_t; /* 定义点类型，x、y 为点坐标 */
void destroyPoint(point_t* this) { free(this); printf("Point destroyed!\n"); } /* 销毁点对象 */
void drawPoint(point_t* this) { printf("P(%d,%d)", this->x, this->y); } /* 绘制点对象 */
shape_t* createPoint(va_list* ap) { /* 创建点对象，并设置其属性 */
    point_t* p_point;
    if( (p_point = (point_t*)malloc(sizeof(point_t))) == NULL ) return NULL;
    p_point->common.type = point; p_point->common.destroy = destroyPoint;
    p_point->common.draw = drawPoint;
    p_point->x = va_arg(*ap, int); /* 设置点的横坐标 */
    p_point->y = va_arg(*ap, int); /* 设置点的纵坐标 */
    return (shape_t*)p_point; /* 返回点对象指针 */
}
typedef struct { /* 定义圆类型 */
    shape_t common;
    point_t *center; /* 圆心点 */
    int radius; /* 圆半径 */
} circle_t;
void destroyCircle(circle_t* this){
    free(__ (1) ); free(this); printf("Circle destroyed!\n");
}
void drawCircle(circle_t* this) {
```

```

    printf("C");
    (2) .draw( this->center );      /* 绘制圆心 */
    printf("%d", this->radius);
}
shape_t* createCircle(va_list* ap) { /* 创建一个圆，并设置其属性 */
    circle_t* p_circle;
    if( (p_circle = (circle_t*)malloc(sizeof(circle_t))) == NULL ) return NULL;
    p_circle->common.type = circle;    p_circle->common.destroy = destroyCircle;
    p_circle->common.draw = drawCircle;
    (3) = createPoint(ap);          /* 设置圆心 */
    p_circle->radius = va_arg(*ap, int); /* 设置圆半径 */
    return p_circle;
}
shape_t* createShape(shape_type st, ...) { /* 创建某一种具体的图形 */
    va_list ap;                          /* 可变参数列表 */
    shape_t* p_shape = NULL;
    (4) (ap, st);
    if( st == point ) p_shape = createPoint( &ap); /* 创建点对象 */
    if( st == circle ) p_shape = createCircle(&ap); /* 创建圆对象 */
    va_end(ap);
    return p_shape;
}
int main() {
    int i;                                /* 循环控制变量，用于循环计数 */
    shape_t* shapes[2]; /* 图形指针数组，存储图形的地址 */
    shapes[0] = createShape( point, 2, 3); /* 横坐标为 2，纵坐标为 3 */
    shapes[1] = createShape( circle, 20, 40, 10); /* 圆心坐标(20,40)，半径为 10 */
    for(i=0; i<2; i++) { shapes[i]->draw(shapes[i]); printf("\n"); } /* 绘制数组中图形 */
    for( i = 1; i >= 0; i-- ) shapes[i]->destroy(shapes[i]); /* 销毁数组中图形 */
    return 0;
}

```

#### [运行结果]

```

P(2,3)
(5)
Circle destroyed!
Point destroyed!

```

### 试题六（共 15 分）

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】

已知某企业的采购审批是分级进行的，即根据采购金额的不同由不同层次的主管人员来审批，主任可以审批 5 万元以下（不包括 5 万元）的采购单，副董事长可以审批 5 万元至 10 万元（不包括 10 万元）的采购单，董事长可以审批 10 万元至 50 万元（不包括 50 万元）的采购单，50 万元及以上的采购单就需要开会讨论决定。

采用责任链设计模式（Chain of Responsibility）对上述过程进行设计后得到的类图如图 6-1 所示。

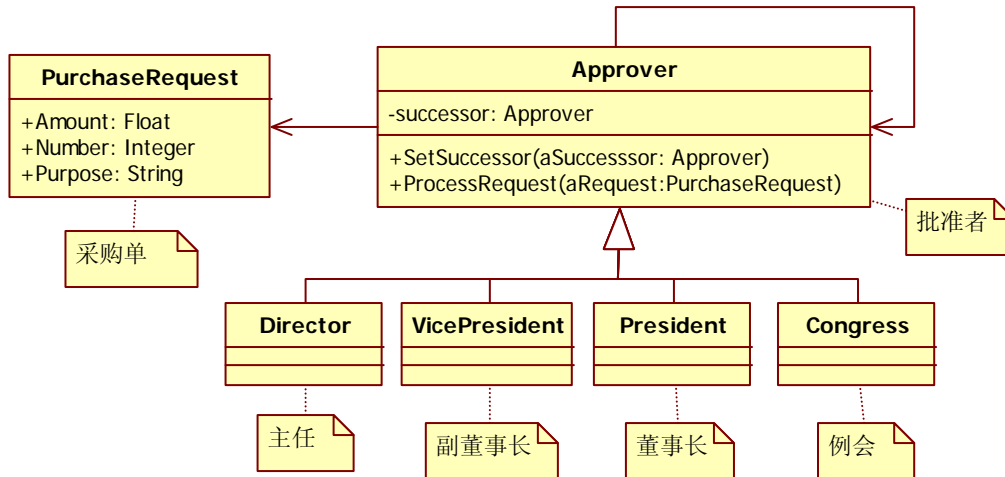


图 6-1 设计类图

#### [C++代码]

```
#include <string>
#include <iostream>
using namespace std;
class PurchaseRequest {
public:
    double Amount;           // 一个采购的金额
    int Number;              // 采购单编号
    string Purpose;          // 采购目的
};
class Approver {             // 审批者类
public:
    Approver(){ successor = NULL; }
    virtual void ProcessRequest(PurchaseRequest aRequest){
        if (successor != NULL){ successor->    (1)    ; }
    }
};
```

```

    void SetSuccessor(Approver *aSuccessor){ successor = aSuccessor; }
private:
    ____ (2) ____ successor;
};
class Congress : public Approver {
public:
    void ProcessRequest(PurchaseRequest aRequest){
        if(aRequest.Amount >= 500000){ /* 决定是否审批的代码省略 */ }
        else ____ (3) ____ ProcessRequest(aRequest);
    }
};
class Director : public Approver {
public:
    void ProcessRequest(PurchaseRequest aRequest){ /* 此处代码省略 */ }
};
class President : public Approver {
public:
    void ProcessRequest(PurchaseRequest aRequest){ /* 此处代码省略 */ }
};
class VicePresident : public Approver {
public:
    void ProcessRequest(PurchaseRequest aRequest){ /* 此处代码省略 */ }
};
void main(){
    Congress Meeting; VicePresident Sam; Director Larry ; President Tammy;
    // 构造责任链
    Meeting.SetSuccessor(NULL);          Sam.SetSuccessor(____ (4) ____);
    Tammy.SetSuccessor(____ (5) ____);    Larry.SetSuccessor(____ (6) ____);

    PurchaseRequest aRequest; // 构造一采购审批请求
    cin >> aRequest.Amount;    // 输入采购请求的金额

    ____ (7) ____ .ProcessRequest(aRequest);    // 开始审批
    return ;
}

```

### 试题七（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】

已知某企业的采购审批是分级进行的，即根据采购金额的不同由不同层次的主管人员来审批，主任可以审批 5 万元以下（不包括 5 万元）的采购单，副董事长可以审批 5 万元至 10 万元（不包括 10 万元）的采购单，董事长可以审批 10 万元至 50 万元（不包括 50 万元）的采购单，50 万元及以上的采购单就需要开会讨论决定。

采用责任链设计模式（Chain of Responsibility）对上述过程进行设计后得到的类图如图 7-1 所示。

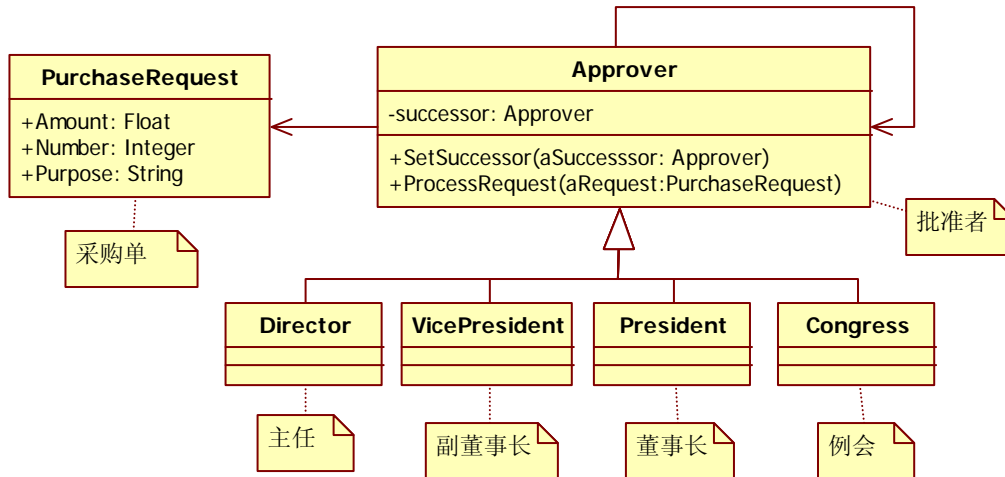


图 7-1 设计类图

#### [Java 代码]

```
class PurchaseRequest {
    public double Amount; // 一个采购的金额
    public int Number; // 采购单编号
    public String Purpose; // 采购目的
};

class Approver { // 审批者类
    public Approver(){ successor = null; }
    public void ProcessRequest(PurchaseRequest aRequest){
        if (successor != null){ successor.__(1)__; }
    }
    public void SetSuccessor(Approver aSuccessor){ successor = aSuccessor; }
    private ____(2)____ successor;
};

class Congress extends Approver {
```

```

public void ProcessRequest(PurchaseRequest aRequest){
    if(aRequest.Amount >= 500000){ /* 决定是否审批的代码省略 */ }
    else ____ (3) ____ .ProcessRequest(aRequest);
}
};
class Director extends Approver {
    public void ProcessRequest(PurchaseRequest aRequest){ /* 此处代码省略 */ }
};
class President extends Approver {
    public void ProcessRequest(PurchaseRequest aRequest){ /* 此处代码省略 */ }
};
class VicePresident extends Approver {
    public void ProcessRequest(PurchaseRequest aRequest){ /* 此处代码省略 */ }
};

public class rs {
    public static void main(String[] args) throws IOException {
        Congress Meeting = new Congress();
        VicePresident Sam = new VicePresident();
        Director Larry = new Director();
        President Tammy = new President();
        // 构造责任链
        Meeting.SetSuccessor(null);    Sam.SetSuccessor(____ (4) ____);
        Tammy.SetSuccessor(____ (5) ____);    Larry.SetSuccessor(____ (6) ____);

        // 构造一采购审批请求
        PurchaseRequest aRequest = new PurchaseRequest();
        BufferedReader br =
            new BufferedReader(new InputStreamReader(System.in));
        aRequest.Amount = Double.parseDouble(br.readLine());

        ____ (7) ____ .ProcessRequest(aRequest);    // 开始审批
        return ;
    }
}

```