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 2010 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 某大型企业的数据中心为了集中管理、控制用户对数据的访问并支持大量的连接需求，欲构建数据管理中间件，其主要功能如下：

(1）数据管理员可通过中间件进行用户管理、操作管理和权限管理。用户管理维护用户信息，用户信息(用户名、密码)存储在用户表中：操作管理维护数据实体的标准操作及其所属的后端数据库信息，标准操作和后端数据库信息存放在操作表中：权限管理维护权限表，该表存储用户可执行的操作信息。

(2）中间件验证前端应用提供的用户信息。若验证不通过，返回非法用户信息；若验证通过，中间件将等待前端应用提交操作请求。

(3）前端应用提交操作请求后，中间件先对请求进行格式检查。如果格式不正确，返回格式错误信息；如果格式正确，则进行权限验证(验证用户是否有权执行请求的操作)，若用户无权执行该操作，则返回权限不足信息，否则进行连接管理。

(4）连接管理连接相应的后台数据库并提交操作，连接管理先检查是否存在空闲的数据库连接，如果不存在，新建连接；如果存在，则重用连接。

(5）后端数据库执行操作并将结果传给中间件，中间件对收到的操作结果进行处理后，将其返回给前端应用。

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

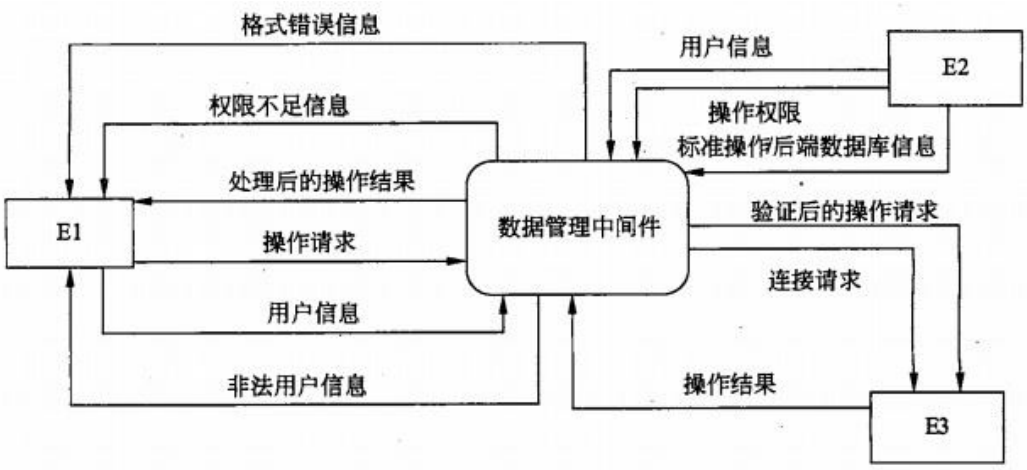


图 1-1　顶层数据流图

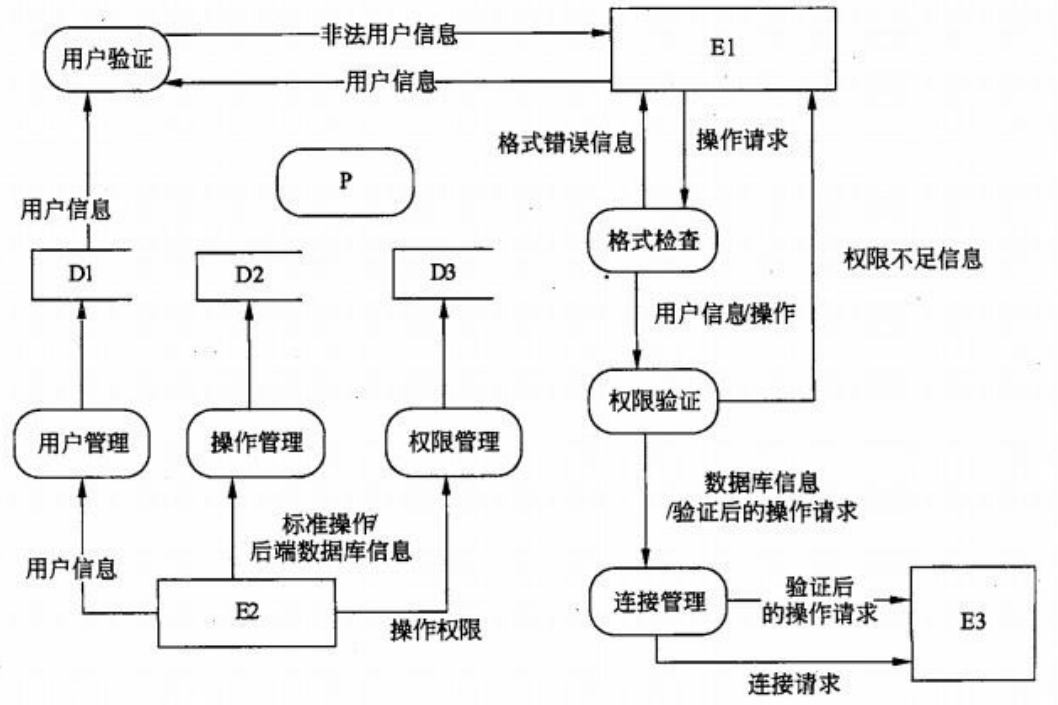


图 1-2　0 层数据流图

问题：1.1

使用说明中的词语，给出图 1-1 中的实体 E1　E3 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　D3 的名称。

问题：1.3

给出图 1-2 中加工 P 的名称及其输入、输出流。

除加工 P 的输入与输出流外，图 1-2 还缺失了两条数据流，请给出这两条数据流的起点和终点。

	名 称	起 点	终 点
输入流			P
输出流		P	

起 点	终 点

注：名称使用说明中的词汇，起点和终点均使用图 1-2 中的符号或词汇。

问题： 1.4

在绘制数据流图时，需要注意加工的绘制。请给出三种在绘制加工的输入、输出时可能出现的错误。

试题二 某学校拟发一套实验管理系统，对各课程的实验安排情况进行管理。

【需求分析】

一个实验室可进行多种类型不同的实验。由于实验室和实验员资源有限，需根据学生人数分批次安排实验室和实验员。一门课程可以为多个班级开设，每个班级每学期可以开设多门课程。一门课程的一种实验可以根据人数、实验室的可容纳人数和实验类型， 分批次开设在多个实验室的不同时间段。一个实验室的一次实验可以分配多个实验员负责辅导实验，实验员给出学生的每次实验成绩。

(1）课程信息包括：课程编号、课程名称、实验学时、授课学期和开课的班级等信息；实验信息记录该课程的实验进度信息，包括：实验名、实验类型、学时、安排周次等信息，如表 2-1 所示。

(2）以课程为单位制定实验安排计划信息，包括：实验地点，实验时间、实验员等信息，实验计划如表 2-2 所示。

(3）由实验员给出每个学生每次实验的成绩，包括：实验名、学号、姓名、班级、实验成绩等信息，实验成绩如表 2-3 所示。

(4）学生的实验课程总成绩根据每次实验的成绩以及每次实验的难度来计算。

【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(不完整)如图 2-1 所示。

根据概念模型设计阶段完成的实体联系图，得出如下关系模式(不完整)：

表 2-1　课程及实验信息

课程编号	15054037	课程名称	数字电视原理		实验学时	12
班级	电 0501,信 0501,计 0501	授课院系	机械与电气工程		授课学期	第三学期
序号	实验名		实验类	难度	学时	安排周次
1505403701	音视频 AD-DA 实验		验证性	1	2	3
1505403702	音频编码实验		验证性	2	2	5
1505403703	视频编码实验		演示性	0.5	1	9

表 2-2　实验安排计划

课程编号	15054037	课程名称	数字电视原理	安排学期	2009 年秋	总人数	220
实验编号	实验名		实验员	实验时间	地点	批次号	人数
1505403701	音视频 AD-DA 实验		盛×，陈×	第 3 周周四晚上	实验三楼 310	1	60
1505403701	音视频 AD-DA 实验		盛×，陈×	第 3 周周四晚上	实验三楼 310	2	60
1505403701	音视频 AD-DA 实验		吴×，刘×	第 3 周周五晚上	实验三楼 311	3	60
1505403701	音视频 AD-DA 实验		吴×	第 3 周周五晚上	实验三楼 311	4	40
1505403702	音频编码实验		盛×，刘×	第 5 周周一下午	实验四楼 410	1	70

表 2-3　实验成绩

实验员：盛×

实验名	音视频 AD-DA 实验	课程名	数字电视原理
学号	姓名	班级	实验成绩
030501001	陈民	信 0501	87
030501002	刘志	信 0501	78
040501001	张勤	计 0501	86

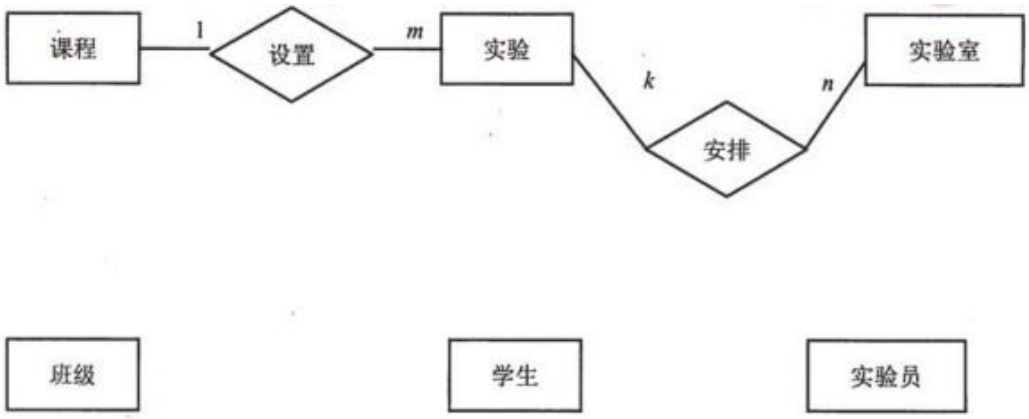


图 2-1　实体联系图

课程（课程编号，课程名称，授课院系，实验学时）
班级（班级号，专业，所属系）
开课情况（　　(1)　　，授课学期）
实验（　　(2)　　，实验类型，难度，学时，安排周次）
实验计划（　　(3)　　，实验时间，人数）
实验员（　　(4)　　，级别）
实验室（实验室编号，地点，开放时间，可容纳人数，实验类型）
学生（　　(5)　　，姓名，年龄，性别）
实验成绩（　　(6)　　，实验成绩，评分实验员）

问题：2.1
补充图 2-1 中的联系和联系的类型。
问题：2.2
根据图 2-1，将逻辑结构设计阶段生成的关系模式中的空(1)（ 6)补充完整并用下划线指出这六个关系模式的主键。
问题：2.3
如果需要记录课程的授课教师，新增加“授课教师”实体。请对图 2-1 进行修改，画出修改后的实体间联系和联系的类型。

试题三　某运输公司决定为新的售票机开发车票销售的控制软件。图 3-1 给出了售票机的面板示意图以及相关的控制部件。

售票机相关部件的作用如下所述：

(1）目的地键盘用来输入行程目的地的代码(例如， 200 表示总站)。

(2) 乘客可以通过车票键盘选择车票种类(单程票、多次往返票和座席种类)。

(3) 继续/取消键盘上的取消按钮用于取消购票过程，继续按钮允许乘客连续购买多张票。

(4) 显示屏显示所有的系统输出和用户提示信息。

(5) 插卡口接受 MCard（现金卡)，硬币口和纸币槽接受现金。

(6) 打印机用于输出车票。

假设乘客总是支付恰好需要的金额而无需找零，售票机的维护工作(取回现金、放入空白车票等)由服务技术人员完成。

系统采用面向对象方法开发/使用 UML 进行建模。系统的顶层用例图和类图分别如图 3-2 和图 3-3 所示。

系统采用面向对象方法开发/使用 UML 进行建模。系统的顶层用例图和类图分别 如图 3-2 和图 3-3 所示。

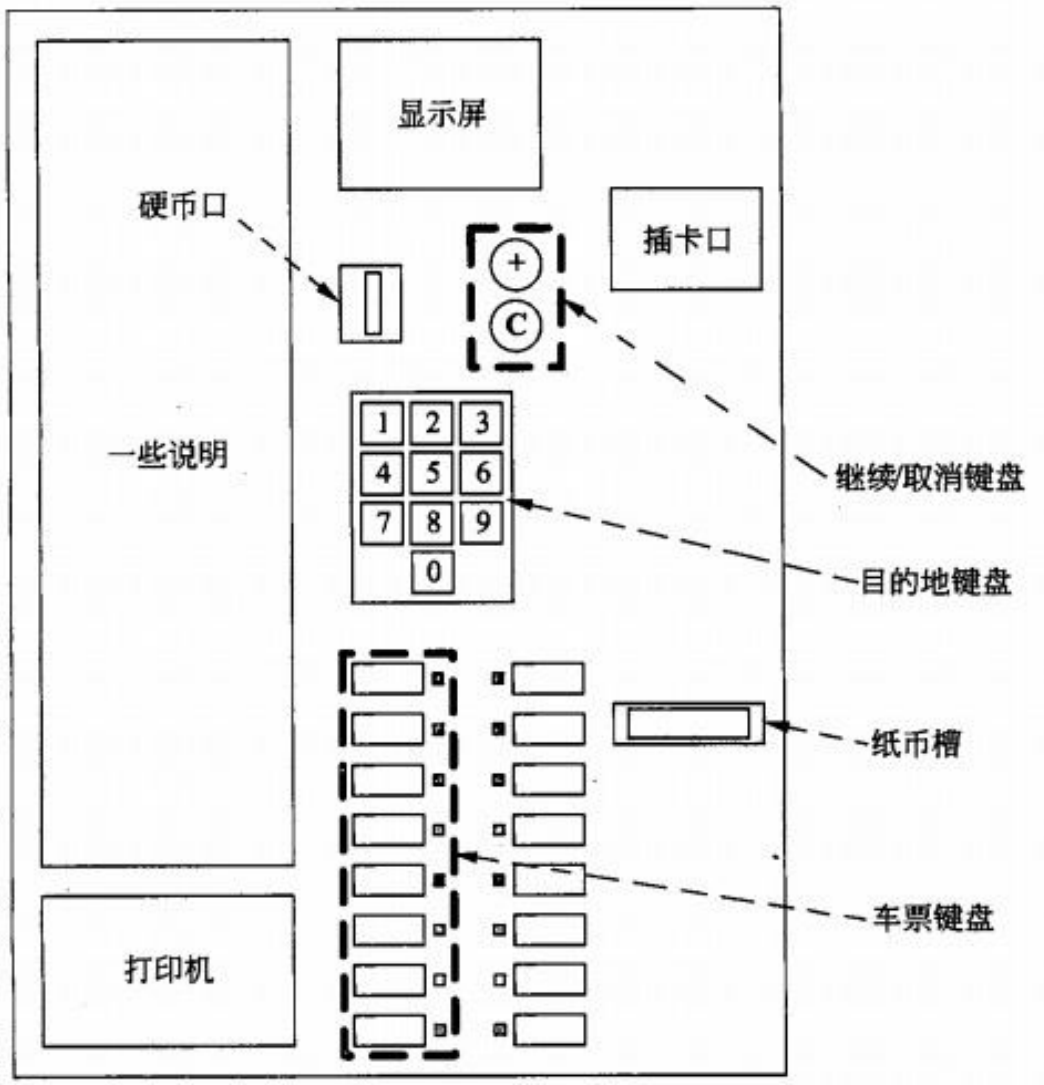


图 3-1　售票机面板示意图

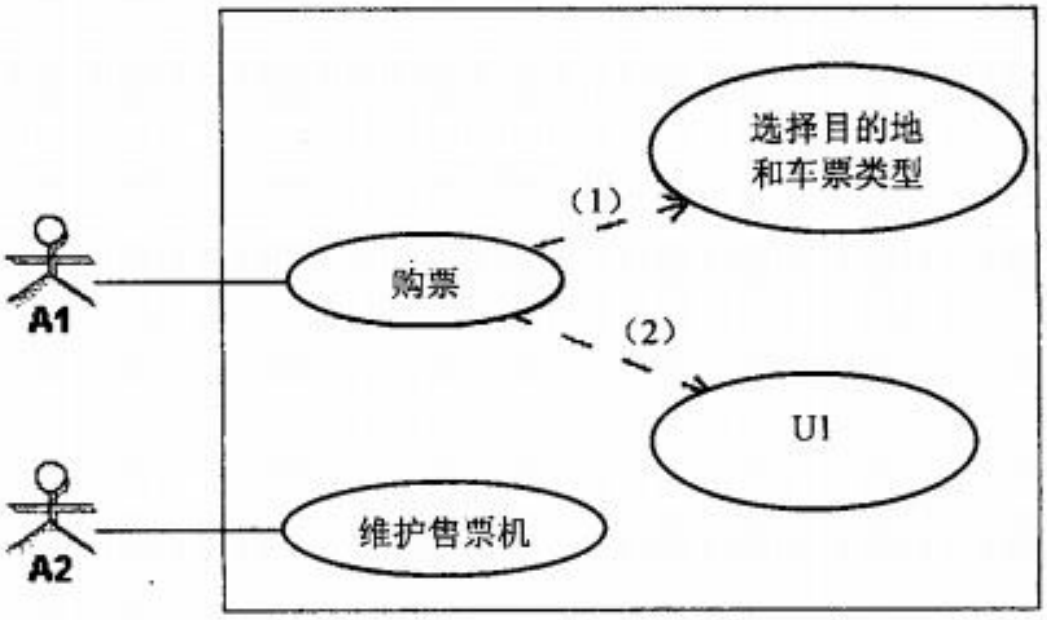


图 3-2　顶层用例图

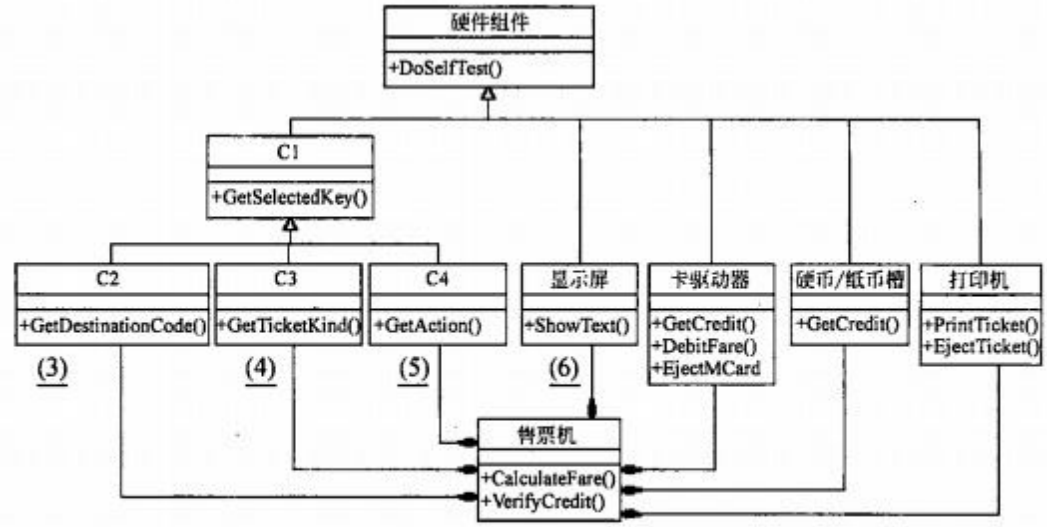


图 3-3　类图

问题：3.1

根据说明中的描述，给出图 3-2 中 A1 和 A2 所对应的参与者， U1 所对应的用例，以及(1)、(2)处所对应的关系。

问题：3.2

根据说明中的描述，给出图 3-3 中缺少的 C1　C4 所对应的类名以及(3)（ 6）处所对应的多重度。

问题：3.3

图 3-3 中的类图设计采用了中介者(Mediator)设计模式，请说明该模式的内涵。

试题四　对有向图进行拓扑排序的方法是：

(1）初始时拓扑序列为空；

(2）任意选择一个入度为 0 的顶点，将其放入拓扑序列中，同时从图中删除该顶点 以及从该顶点出发的弧：

(3）重复(2），直到不存在入度为 0 的顶点为止(若所有顶点都进入拓扑序列则完 成拓扑排序，否则由于有向图中存在回路无法完成拓扑排序)。

函数 int* TopSort(LinkedDigraphG)的功能是对有向图 G 中的顶点进行拓扑排序，返回拓扑序列中的顶点编号序列，若不能完成拓扑排序，则返回空指针。其中，图 G 中的顶点从 1 开始依次编号，顶点序列为 vl ， v2, …， vn, 图 G 采用邻接表表示，其数据类型定义如下：

例如，某有向图 G 如图 4-1 所示，其邻接表如图 4-2 所示。

函数 TopSort 中用到了队列结构(Queue 的定义省略)，实现队列基本操作的函数原型如下表所示：

```
#define MAXVNUM 50                  /* 最大顶点数 */
typedef struct ArcNode{             /* 表结点类型 */
      int adjvex;                   /* 邻接顶点编号 */
      struct ArcNode *nextarc;      /* 指示下一个邻接顶点 */
}ArcNode;
typedef struct AdjList{             /* 头结点类型 */
      char vdata;                   /* 顶点的数据信息 */
      ArcNode *firstarc;            /* 指向邻接表的第一个表结点 */
}AdjList;
typedef struct LinkedDigraph{       /* 图的类型 */
      int n;                        /* 图中顶点个数 */
      AdjList Vhead[MAXVNUM];       /* 所有顶点的头结点数组 */
}LinkedDigraph;
```

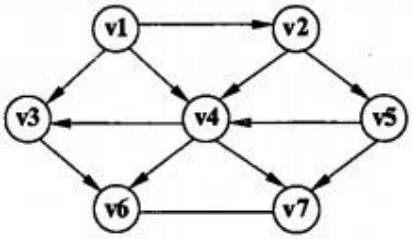


图 4-1　有向图 G

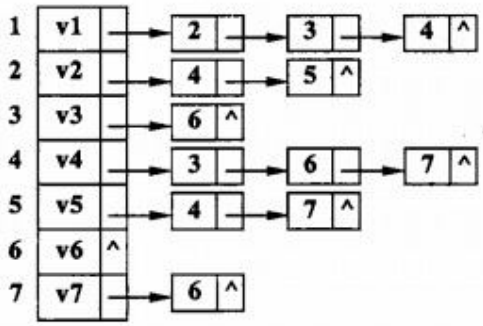


图 4-2　有向图 G 的邻接表示意图

函 数 原 型	说　明
void InitQueue(Queue *Q)	初始化队列（构造一个空队列）
bool IsEmpty(Queue Q)	判断队列是否为空，若是则返回 true，否则返回 false
void EnQueue(Queue *Q, int e)	元素入队列
void DeQueue(Queue *Q,int *p)	元素出队列

【C 代码】

```
int *TopSort(LinkedDigraph G) {
    ArcNode *p;                          /* 临时指针，指示表结点 */
Queue Q;                                 /* 临时队列，保存入度为 0 的顶点编号 */
int k = 0;                               /* 临时变量，用作数组元素的下标 */
int j = 0, w = 0;                        /* 临时变量，用作顶点编号 */
int  *topOrder, *inDegree;
topOrder = (int *)malloc((G.n+1) * sizeof(int));
                                         /* 存储拓扑序列中的顶点编号 */
inDegree = (int *)malloc((G.n+1) * sizeof(int));
                                         /* 存储图 G 中各顶点的入度 */
if (!inDegree || !topOrder) return NULL;

   (1)   ;                               /* 构造一个空队列 */

for ( j = 1; j <= G.n; j++ ) {  /* 初始化 */
     topOrder[j] = 0;    inDegree[j] = 0;
}

  for (j = 1; j <= G.n; j++)  /* 求图 G 中各顶点的入度 */
  for( p = G.Vhead[j].firstarc; p; p = p->nextarc )
          inDegree[p-> adjvex] += 1;

for (j = 1; j <= G.n; j++)          /* 将图 G 中入度为 0 的顶点保存在队列中 */
if ( 0 == inDegree[j] )     EnQueue(&Q,j);

while (!IsEmpty(Q)) {
        (2)   ;                      /* 队头顶点出队列并用 w 保存该顶点的编号 */
topOrder[k++] = w;
/* 将顶点 w 的所有邻接顶点的入度减 1（模拟删除顶点 w 及从该顶点出发的弧的操作） */
        for(p = G.Vhead[w].firstarc; p; p = p->nextarc) {
            (3)  -= 1;
            if (0 ==   (4)   )   EnQueue(&Q, p->adjvex);
    }/* for */
    }/* while */

    free(inDegree);

    if (   (5)   )
        return NULL;
return topOrder;
} /*TopSort*/
```

问题： 4.1

根据以上说明和 C 代码，填充 C 代码中的空(1) (5)。

问题： 4.2

对于图 4-1 所示的有向图 G, 写出函数 TopSort 执行后得到的拓扑序列。若将函数 TopSort 中的队列改为栈，写出函数 TopSort 执行后得到的拓扑序列。

问题： 4.3

设某有向无环图的顶点个数为 n 、弧数为 e ，那么用邻接表存储该图时，实现上述拓扑排序算法的函数 TopSort 的时间复杂度是(6)。 若有向图采用邻接矩阵表示(例如，图 4-1 所示有向图的邻接矩阵如图 4-3 所示)，且将函数 TopSort 中有关邻接表的操作修改为针对邻接矩阵的操作，那么对于有 n 个顶点、 e 条弧的有向无环图，实现上述拓扑排序算法的时间复杂度是(7)。

	v1	v2	v3	v4	v5	v6	v7
v1	0	1	1	1	0	0	0
v2	0	0	0	1	1	0	0
v3	0	0	0	0	0	1	0
v4	0	0	1	0	0	1	1
v5	0	0	0	1	0	0	1
v6	0	0	0	0	0	0	0
v7	0	0	0	0	0	1	0

图 4-3 有向图 G 的邻接矩阵

试题五 某软件公司现欲开发一款飞机飞行模拟系统，该系统主要模拟不同种类飞机的飞行特征与起飞特征。需要模拟的飞机种类及其特征如表 5-1 所示。

为支持将来模拟更多种类的飞机，采用策略设计模式(Strategy)设计的类图如图 5-1 所

示。 ..

图 5-1 中， AirCraft 为抽象类，描述了抽象的飞机，而类 Helicopter 、 AirPlane 、 Fighter 和 Harrier 分别描述具体的飞机种类，方法 flyO 和 takeOff()分别表示不同飞机都具有飞行特征和起飞特征；类 FlyBehavior 与 TakeOfffIehavior 为抽象类，分别用于表示抽象的飞行行为与起飞行为；类 SubSonicFly 与 SuperSonicFly 分别描述亚音速飞行和超音速飞行 的行为；类 VerticalTakeOff 与 LongDistanceTakeOff 分别描述垂直起飞与长距离起飞的行为。

表 5-1

飞 机 种 类	起 飞 特 征	飞 行 特 征
直升机（Helicopter）	垂直起飞（VerticalTakeOff）	亚音速飞行（SubSonicFly）
客机（AirPlane）	长距离起飞（LongDistanceTakeOff）	亚音速飞行（SubSonicFly）
歼击机（Fighter）	长距离起飞（LongDistanceTakeOff）	超音速飞行（SuperSonicFly）
鹞式战斗机（Harrier）	垂直起飞（VerticalTakeOff）	超音速飞行（SuperSonicFly）

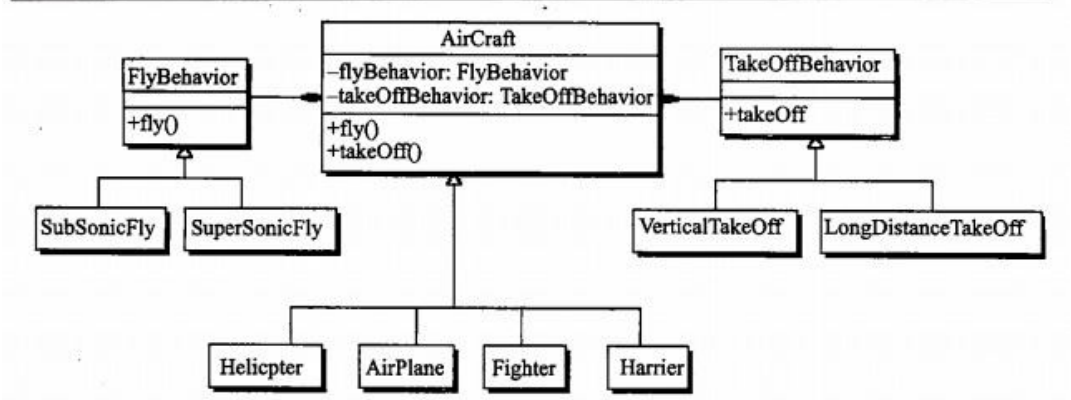


图 5-1 类图

问题： 5.1

【C++代码】

```
#include<iostream>
using namespace std;
class FlyBehavior {
public : virtual void fly() = 0;
};
class SubSonicFly:public FlyBehavior{
public: void fly(){ cout << "亚音速飞行 ! " << endl; }
};
class SuperSonicFly:public FlyBehavior{
public: void fly(){ cout << "超音速飞行 ! " << endl; }
};

class TakeOffBehavior {
public: virtual void takeOff() = 0;
};
class VerticalTakeOff:public TakeOffBehavior{
public: void takeOff(){ cout << "垂直起飞 ! " << endl;    }
};
class LongDistanceTakeOff:public TakeOffBehavior {
public: void takeOff (){ cout << "长距离起飞 ! " << endl; }
};

class AirCraft{
protected:
   (1)   ;
   (2)   ;
public:
    void fly(){  (3)  ;  }
    void takeOff() {  (4)  ; };
};
class Helicopter: public AirCraft {
public:
    Helicopter (){
        flyBehavior = new  (5)  ;
        takeOffBehavior = new  (6)  ;
    }
     (7)  {
        if(!flyBehavior) delete flyBehavior;
        if(!takeOffBehavior) delete takeOffBehavior;
    }
};
//其他代码省略
```

试题六 某软件公司现欲开发一款飞机飞行模拟系统，该系统主要模拟不同种类飞机的飞行特征与起飞特征。需要模拟的飞机种类及其特征如表 6-1 所示。

为支持将来模拟更多种类的飞机，采用策略设计模式(Strategy)设计的类图如图 6-1 所示。

图 6-1 中， AirCraft 为抽象类，描述了抽象的飞机，而类 Helicopter 、 AirPlane 、 Fighter 和 Harrier 分别描述具体的飞机种类，方法 fly()和 takeOffQ 分别表示不同飞机都具有飞行特征和起飞特征；类 FlyBehavior 与 TakeOffBehavior 为抽象类，分别用于表示抽象的飞行行为与起飞行为；类 SubSonicFly 与 SuperSonicFly 分别描述亚音速飞行和超音速飞行 的行为；类 VerticalTakeOff 与 LongDistanceTakeOff 分别描述垂直起飞与长距离起飞的 行为。

表 6-1

飞 机 种 类	起 飞 特 征	飞 行 特 征
直升机（Helicopter）	垂直起飞（VerticalTakeOff）	亚音速飞行（SubSonicFly）
客机（AirPlane）	长距离起飞（LongDistanceTakeOff）	亚音速飞行（SubSonicFly）
歼击机（Fighter）	长距离起飞（LongDistanceTakeOff）	超音速飞行（SuperSonicFly）
鹞式战斗机（Harrier）	垂直起飞（VerticalTakeOff）	超音速飞行（SuperSonicFly）

问题： 6.1

【Java 代码】

```
interface FlyBehavior {
    public void fly();
};
class SubSonicFly implements FlyBehavior{
    public void fly(){ System.out.println("亚音速飞行 ！"); }
};
class SuperSonicFly implements FlyBehavior{
    public void fly(){ System.out.println("超音速飞行 ！" ); }
};
```

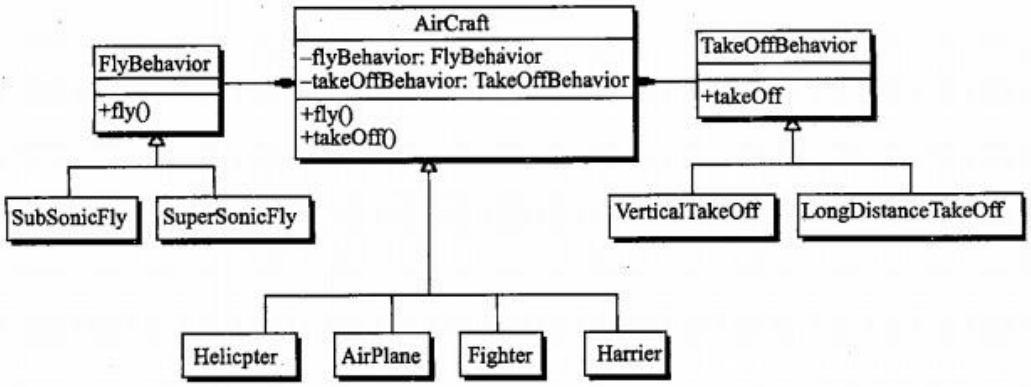


图 6-1 类图

```
interface TakeOffBehavior {
    public void takeOff();
};

class VerticalTakeOff implements TakeOffBehavior {
    public void takeOff (){ System.out.println("垂直起飞！" );  }
};
class LongDistanceTakeOff implements TakeOffBehavior {
    public void takeOff(){ System.out.println("长距离起飞！");  }
};
abstract class AirCraft {
    protected    (1)    ;
    protected    (2)    ;
    public void fly(){   (3)   ;  }
    public void takeOff() {  (4)   ; };
};
class Helicopter   (5)   AirCraft{
    public Helicopter (){
        flyBehavior = new   (6)   ;
        takeOffBehavior = new   (7)   ;
    }
};
//其他代码省略
```

试题一 答案：　解析：　本问题考查顶层 DFD 。顶层 DFD 一般用来确定系统边界，将待开发系统看作一个加工，因此图中只有唯一的一个加工和一些外部实体，以及这两者之间

的输入输出数据流。题目要求根据描述确定图中的外部实体。分析题目中的描述，并结合已经在顶层数据流图中给出的数据流进行分析。题目中有信息描述：数据管理员可通过中间件进行用户管理、操作管理和权限管理；前端应用提交操作请求；连接管理连接相应的后台数据库并提交操作。由此可知该中间件系统有数据管理员、前端应用和后端数据库三个外部实体。从图 1-1 中数据流和实体的对应关系可知， E1 为前端应用， E2 为数据管理员， E3 为后端数据库。

本问题考查 0 层 DFD 中数据存储的确定。说明中描述：用户信息(用户名、密码) 存储在用户表中；标准操作和后端数据库信息存放在操作表中；权限管理维护信息存放在权限表中。因此数据存储为用户表、操作表以及权限表。再根据图 1-2 可知 D1 的输入数据流从用户管理来， D2 的输入数据流从操作管理来， D3 的输入数据流从权限管理来，所以 D1 为用户表， D2 为操作表， D3 为权限表。

本问题考查 0 层 DFD 中缺失的加工和数据流。比较图 1-1 和图 1-2, 可知顶层 DFD 中的操作结果和处理后的操作结果没有在 0 层 DFD 中体现。再根据描述“后端数据库执行操作并将结果传给中间件，中间件对收到的操作结果进行处理后，将其返回给前端应用”可知，需要有操作结果处理，因此 P 为操作结果处理，其输入流为从后端数据库 E3 来的操作结果，输出结果为处理后的操作结果，并返回给前端应用 E1 。

考查完 P 及其输入输出流之后，对图 1-2 的内部数据流进行考查，以找出缺失的另外 2 条数据流。从图中可以看出 D2 和 D3 只有输入流没有输出流，这是常见 DFD 设计时的错误，所以首先考查 D2 和 D3 的输出流。描述中有“权限验证是验证用户是否有权执行请求的操作，若用户有权执行该操作，进行连接管理；连接管理连接相应的后台数 据库并提交操作;权限表存储用户可执行的操作信息”。因此，权限验证有从权限表 D3 来的输入数据流。而要连接后端数据库，需要数据库信息，从权限验证的输出流中包含有数据库信息可知，权限验证需要获取到数据库信息，所以还需从操作表 D2 来的输入流。

在绘制数据流图的加工时，可能出现的输入、输出错误：

只有输入而无输出 或者 黑洞

只有输出而无输入 或者 奇迹

输入的数据流无法通过加工产生输出流 或者 灰洞

输入的数据流与输出的数据流名称相同

P 的名称：操作结果处理

	名　称	起　点	终　点
输入流	操作结果	E3	P
输出流	处理后的操作结果	P	E1

缺少的数据流：

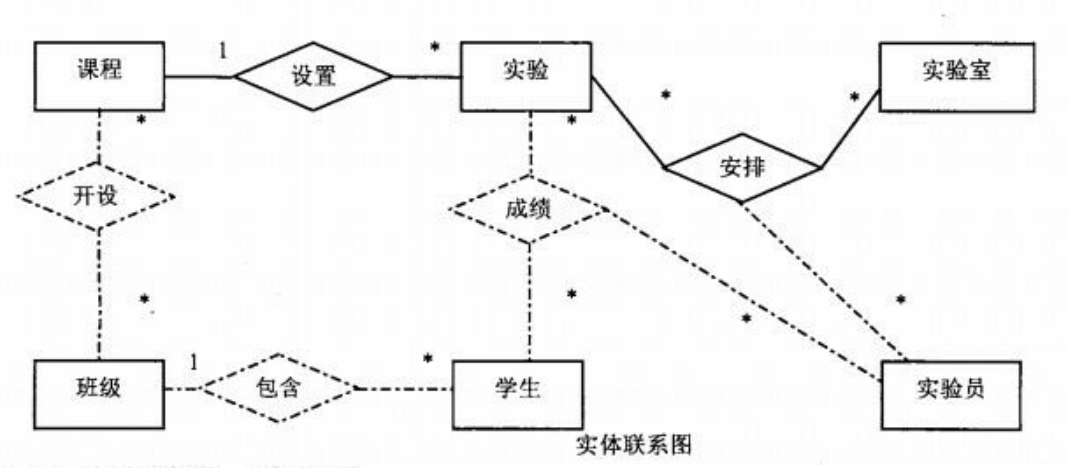
起　点	终　点
D2	权限验证
D3	权限验证

试题二 答案：　解析：

根据题意，由“一门含实验的课程可以开设给多个班级，每个班级每学期可以开设多门含实验的课程”可知课程和班级之间的开设关系为 m:n 联系。由“一个实验室的一次实验可以分配多个实验员负责辅导实验”可知实验、实验室与实验员之间的安排关系为 k:n:m 联系。由“实验员给出学生的每次实验成绩”可知实验、学生与实验员之间的成绩关系为联系。班级和学生之间的包含关系为 1:n 联系。

实验室(实验室编号,地点，开放时间，可容纳人数，实验课类型)根据题意可知课程编号是课程的主键，班级号是班级的主键。从表 2-1 可知，开课情况是体现课程与班级间的 m:n 联系，因此开课情况关系模式应该包含课程编号和班级号，并共同作为主键。一门课程包含多次实验，实验与课程之间是 m:1 关系，因此，根据表 2-1，实验关系模式应包含实验编号和课程编号，并且以实验编号为主键，以课程编号为外键。在制定试验计划时，每个班的每次实验可能按实验室被分成多个批次，每个批次的实验会有若干名实验员来辅导学生实验并打分。实验员关系模式应该记录实验员编号和实验员姓名，并以实验员编号为主键。实验室编号是实验室的主键。从表 2-2 可见，实验计划关系模式应记录实验编号、批次号和授课学期，并且共同作为主键。从表 2-3 可见，实验成绩关系模式记录每个学生的每次实验成绩，应包含学号和实验编号，并共同作为主键。

由于授课教师负责给若干个班级开设若干门课程，因此，课程、班级和授课教师之间的开设关系是联系。



实体联系图

（1）课程编号，班级号

（2）实验编号，课程编号

（3）实验编号，批次号，安排学期，实验室编号，实验员编号

（4）实验员编号，实验员姓名

（5）学号，班级号

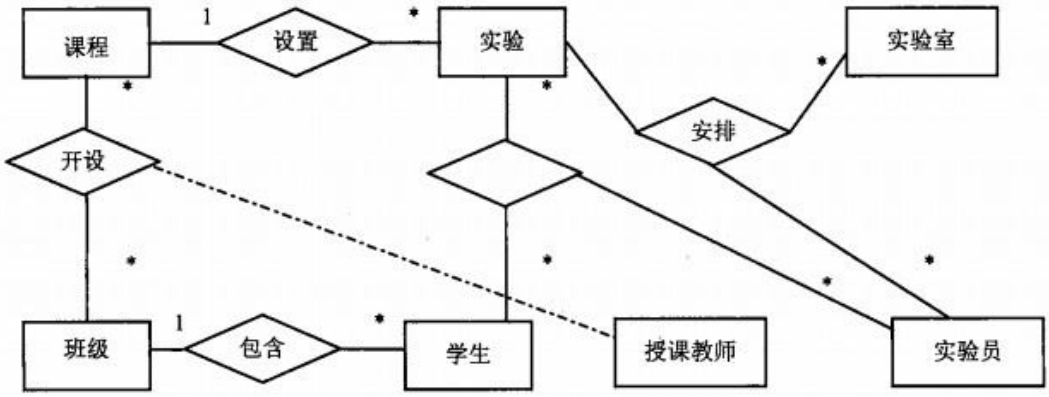
（6）实验编号，学号

其他关系模式主键：

课程（课程编号，课程名称，授课院系，实验学时）

班级（班级号，专业，所属系）

实验室（实验室编号，地点，开放时间，可容纳人数，实验课类型）



试题三 答案：　解析：　A1：乘客 A2:服务技术人员

U1：支付（1）«include»（2）«include»

本问题考查用例图。用例图用于确定系统边界，识别与系统交互的参与者，通过判断参与者发起的用例，建立和参与者之间的关联，然后再确认用例之间的关系。

本题中对售票机的描述为“乘客可以通过车票键盘选择车票种类(单程票、多次往返票和座

席种类)；售票机的维护工作(取回现金、放入空白车票等)由服务技术人员完成”。由此可知，图 3-1 中 A1 为乘客， A2 为服务技术人员。

对购票用例，要选择目的地和车票类型、通过插卡口进行支付才可完成购票。因此 U2 为支付。 ’

在考查用例之间的关系时，购票过程可以取消，也允许乘客连续购买多张票，因此，购票时可以包含多次选择目的地和车票类型、支付，即购票用例包含(关系《 include 》) 选择目的地和车票类型以及支付。

C1 ： 键盘 C2:目的地键盘 C3:车票键盘 C4:继续/取消键盘

(3) （ 6) ： 1

本问题考查类图。类图设计的重点是类的抽象和继承关系以及多重度。售票机的面板由多个控制部件组成。根据说明这些控制部件有目的地键盘、车票键盘和继续/取消键盘、显示屏、卡驱动器、硬币/纸币槽、打印机。图 3-3 中只有前 3 个部件在图中没有给出，而要填如 4 个类。从图中已经抽象出的硬件组件，给出了抽象的思路，从而可以把键盘抽象出来。由 C1 与 C2 、 C3 、 C4 的继承关系中 C1 为基类，可知 C1 为键盘。由 C2 、 C3 和 C4 给出的方法名称可知， C2 为目的地键盘获取目的地代码， C3 为车票键盘选择 产品类型， C4 为继续/和取消动作。

本题中的重复度比较简单. 从图 3-1 售票机的图示中可以看出，一个售票机只包含一个目的地键盘、一个车票键盘和一个继续/取消键盘，因此(3) （ 6）均为 1 。

本问题考查设计模式。设计模式题目虽然比较难，但是本题题目中已经给出了所采用的设计模式为 Mediator 模式，只需说明设计模式的内涵即可，也比较容易。使用 Mediator 模式，可以使各个对象间的耦合松散，只需关心和 Mediator 的关系，使多对多的关系变成了一对多的关系，可以降低系统的复杂性，提高可修改扩展性。

试题四 答案： 解析： (1) InitQueue(&Q)

(2) DeQueue(&Q ， &w)

(3) inDegree[p-> adjvex]或其等价形式

(4) inDegree[p->adjvex］或其等价形式

(5) k

队列方式： v1v2v5v4v3v7v6 或者 1254376

栈方式： v1v2v5v4v7v3v6 或者 1254736

使用栈和队列的差别在于拓扑序列中顶点的排列次序可能不同。对于本题中的有向图，在使用队列的方式下：

(1）开始时仅顶点 v1 的入度为 0 ，因此顶点 v1 入队；

(2）队头顶点 v1 出队，并进入拓扑序列，然后删除从顶点 v1 出发的弧后，仅使顶点 v2 的入度为 0 ，因此顶点 v2 入队；

(3）队头顶点 v2 出队，并进入拓扑序列，然后删除从顶点 v2 出发的弧后，仅使顶点 v5 的入度为 0 ，因此顶点 v5 入队；

(4）队头顶点 v5 出队，并进入拓扑序列，然后删除从顶点 v5 出发的弧后，仅使顶点 v4 的入度为 0 ，因此顶点 v4 入队；

(5）队头顶点 v4 出队，并进入拓扑序列，然后删除从顶点 v4 出发的弧后，仅使顶点 v3 和 v7 的入度为 0 ，因此顶点 v3 和 v7 依次入队；

(6）队头顶点 v3 出队，并进入拓扑序列，然后删除从顶点 v3 出发的弧后，没有产生新的入度为 0 的顶点；

(7）队头顶点 v7 出队，并进入拓扑序列，然后删除从顶点 v7 出发的弧后，使顶点 v6 的入度为 0 ，因此顶点 v6 入队；

(8）队头顶点 v6 出队，并进入拓扑序列，然后删除从顶点 v6 出发的弧后，没有产生新的入度为 0 的顶点，队列已空，因此结束拓扑排序过程，得到的拓扑序列为 v1v2v5v4v3v7v6 。

使用栈保存入度为 0 的顶点时，前 4 步都是一样的，因为每次仅有一个元素进栈， 因此出栈序列与入栈序列一致。到第 5 步时， v3 和 v7 依次入栈后，出栈时的次序为 v7 和 v3 ，因此得到的拓扑序列为 v1v2v5v4v7v3v6 。

(6) $O(n)$

(7) $O(n^2)$

以邻接表为存储结构时，计算各顶点入度的时间复杂度为 $O(e)$，建立零入度顶点队列的时间复杂度为 $O(n)$。在拓扑排序过程中，(图中无环情况下)每个顶点进出队列各 1 次，入度减 1 的操作在 while 循环中共执行 e 次，所以总的时间复杂度为 $O(n+e)$。

以邻接矩阵为存储结构时，计算各顶点入度时需要遍历整个矩阵，因此时间复杂度为 $O(n^2)$，建立零入度顶点队列的时间复杂度为 $O(n)$。在拓扑排序过程中，(图中无环情况下)每个顶点进出队列各 1 次，实现入度减 1 操作时需遍历每个顶点的行向量 1 遍(时间复杂度为 $O(n)$)，所以总的时间复杂度为 $O(n^2)$。

拓扑排序是将有向无环图中所有顶点排成一个线性序列的过程，并且该序列满足：若在有向图中从顶点 v_i 到 v_j 有一条路径，则在该线性序列中，顶点 v_i 必然在顶点 v_j 之前。

对 AOE 网进行拓扑排序的方法如下：

① 在 AOE 网中选择一个入度为零（没有前驱）的顶点且输出它；

② 从网中删除该顶点及其与该顶点有关的所有边；

③ 重复上述两步，直至网中不存在入度为零的顶点为止。

在拓扑排序过程中，需要将入度为 0 的顶点临时存储起来。函数中用一个队列暂存入度为 0 且没有进入拓扑序列的顶点。显然，空（1）处应填入 InitQueue(&Q)。

进行拓扑排序之前，应先求出网中每个顶点的入度并存入数组 inDegree[]中，从而将“从网中删除该顶点及其与该顶点有关的所有边”的操作转换为“相关顶点的入度减 1”，一旦发现某个顶点的入度变为 0，就将其编号压入堆栈。从而将选择入度为 0 的顶点操作转化为令队头所代表的顶点出队。

根据注释，空（2）处应填入 DeQueue(&Q,&w)，实现队头元素出队列的处理。

题中图采用邻接表存储结构，当指针 p 指向 v_i 邻接表中的结点时，p->adjvex 表示 v_i 的一个邻接顶点，删除 v_i 至顶点 p-> adjvex 的弧的操作实现为顶点 p->adjvex 的入度减 1，因此，空（3）处应填入 inDegree[p->adjvex]，当顶点 p->adjvex 的入度为 0 时，需要将其加入队列，因此空（4）处也应填入 inDegree[p->adjvex]。

空（5）处判断是否所有顶点都加入了拓扑序列，算法中变量 k 用于对加入序列的顶点计数，因此，空（5）处应填入“k < G.n”或“k !=G.n”。

试题五 答案： 解析：

本题目考查了设计模式中的策略设计模式，实际上与 2007 年上半年考核内容相同。

从本题的叙述中可以看出，存在 4 种不同的飞机类型，但每种飞机类型的起飞特征和飞行特征并不完全相同，这就使得我们很难采用比较直接的方法来实现重用。例如，定义一个抽象的飞机类，实现飞机的起飞特征，然后 4 种飞机直接重用该特征。但是我们可以观察到，尽管飞机的起飞特征和飞行特征有所不同，有一点可以肯定的是，每一种飞机都具备了飞行特征和起飞特征。因此，可以抽象出一个飞机类，其中含有飞行特征与起飞特征，但关于两个特征的实现要单独抽取出来，所以又形成了 FlyBehavior 类和 TakeOffBehavior 类分别表示抽象的飞行和起飞特征，而这两个类的子类则分别实现不同的起飞和飞行特征，最终转化为，在创建一个具体的飞机时，给其配上不同的起飞 特征和飞行特征即可。

本题中的空(1) 和空(2) 应该填写成员变量，根据类图可以得知，此处应该表示的是飞行和起飞特征变量，在 C++中可以采用指针来表示。空(3) 和空(4) 处需要实现飞行与起飞特征，但 AirCraft 是抽象的类，所以把实现代理给指针变量。 Helicopter 类需要指定由父类继承而来的成员变量的初始值，因为 Helicopter 的特征是垂直起飞和亚音速飞行，因此生成这两个特征的对象，分别赋值给 flyBehavior 和 takeOffBehavior 变量。

（1）FlyBehavior * flyBehavior

（2）TakeOffBehavior * takeOffBehavior

（3）flyBehavior->fly()

（4）takeOffBehavior->takeOff()__

（5）SubSonicFly()

（6）VerticalTakeOff()

（7）~Helicopter()

注：空（1）与空（2）答案可互换

试题六 答案：　解析： (1) FlyBehavior flyBehavior

(2) TakeOffBehavior takeOffBehavior

(3) flyBehavior.fly()

(4) takeOffBehavior.takeOff()

(5) extends

(6) SubSonicFly()

(7) VerticalTakeOff()

本题目考查了设计模式中的策略设计模式，实际上与 2007 年上半年 Java 题目的考核内容相同。

从本题的叙述中可以看出，存在四种不同的飞机类型，但每种飞机类型的起飞特征和飞行特征并不完全相同，这就使得我们很难采用比较直接的方法来实现重用。例如’定义一个抽象的飞机类，实现飞机的起飞特征，然后四种飞机直接重用该特征。但是，我们可以观察到，尽管飞机的起飞特征和飞行特征有所不同，有一点可以肯定的是，每 一种飞机都具备了飞行特征和起飞特征。因此，可以抽象出一个飞机类，其中含有飞行特征与起飞特征，但关于两个特征的实现要单独抽取出来，所以又形成了 FlyBehavior 类和 TakeOffBehavior 类，分别表示抽象的飞行和起飞特征，而这两个类的子类则分别实现不同的起飞和飞行特征，最终转化为，在创建一个具体的飞机时，给其配上不同的起飞特征和飞行特征即可。

本题中的空(1) 和空(2) 应该填写成员变量，根据类图可以得知，此处应该表示 的是飞行和起飞特征变 M 。空(3) 和空(4) 处需要实现飞行与起飞特征，但 AiiCraft 是抽象的类，所以把实现代理给指针变量。 Helicopter 类需要指定由父类继承而来的成员 变量的初始值，因为 Helicopter 的特征是垂直起飞和亚音速飞行，因此生成这两个特征的对象，分别赋值给 flyBehavior 和 takeOffBehavior 变量。