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 2015 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 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】

某大学为进一步推进无纸化考试，欲开发一考试系统。系统管理员能够创建包括专业方向、课程编号、任课教师等相关考试基础信息，教师和学生进行考试相关的工作。系统与考试有关的主要功能如下。

(1）考试设置。教师制定试题(题目和答案)，制定考试说明、考试时间和提醒时间等考试信息，录入参加考试的学生信息，并分别进行存储。

(2）显示并接收解答。根据教师设定的考试信息，在考试有效时间内向学生显示考试说明和题目，根据设定的考试提醒时间进行提醒，并接收学生的解答。

(3）处理解答。根据答案对接收到的解答数据进行处理，然后将解答结果进行存储。

(4）生成成绩报告。根据解答结果生成学生个人成绩报告，供学生查看。

(5）生成成绩单。对解答结果进行核算后生成课程成绩单供教师查看。

(6）发送通知。根据成绩报告数据，创建通知数据并将通知发送给学生；根据成绩单数据，创建通知数据并将通知发送给教师。

现采用结构化方法对考试系统进行分析与设计，获得如图 1-1 所示的上下文数据流图和图 1-2 所示的 0 层数据流图。

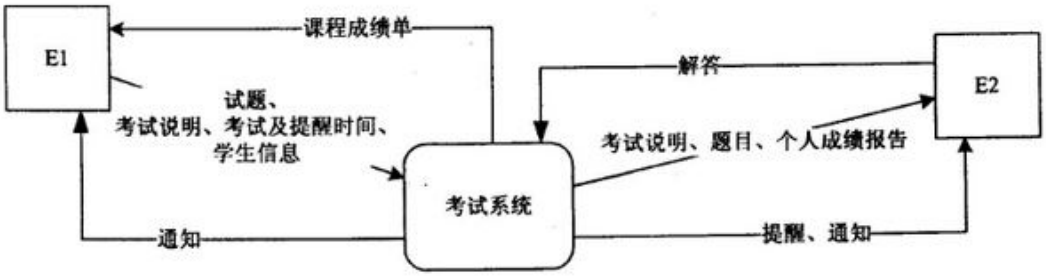

E1
课程成绩单
试题、
考试说明、考试及提醒时间、
学生信息
解答
E2
考试说明、题目、个人成绩报告
考试系统
通知
提醒、通知

图 1-1 上下文数据流图

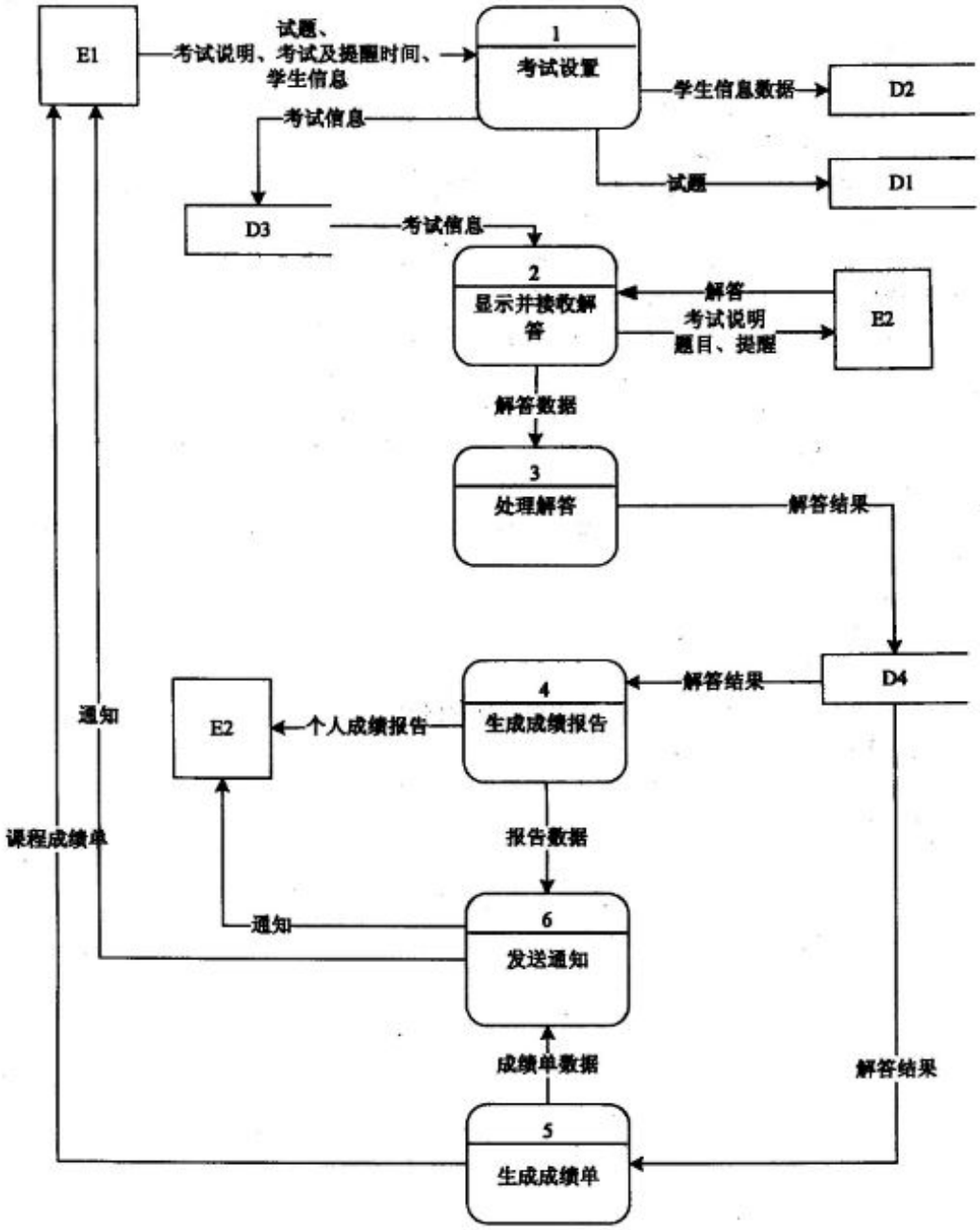


图 1-2　0 层数据流图

问题：1.1

(2 分)

使用说明中的词语，给出图 1-1 中的实体 E1 ～ E2 的名称。

问题：1.2

(4 分)

使用说明中的词语，给出图 1-2 中的数据存储 D1 ～ D4 的名称。

问题：1.3

(4 分)

根据说明和图中词语，补充图 1-2 中缺失的数据流及其起点和终点。

问题：1.4
(5 分)
图 1-2 所示的数据流图中，功能(6)发送通知包含创建通知并发送给学生或老师。请分解图 1-2 中加工(6)，将分解出的加工和数据流填入答题纸的对应栏内。(注：数据流的起点和终点须使用加工的名称描述)

试题二 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】

某省针对每年举行的足球联赛，拟开发一套信息管理系统，以方便管理球队、球员、主教练、主裁判、比赛等信息。

【需求分析】

(1) 系统需要维护球队、球员、主教练、主裁判、比赛等信息。

球队信息主要包括：球队编号、名称、成立时间、人数、主场地址、球队主教练。

球员信息主要包括：姓名、身份证号、出生日期、身高、家庭住址。

主教练信息主要包括：姓名、身份证号、出生日期、资格证书号、级别。

主裁判信息主要包括：姓名、身份证号、出生日期、资格证书号、获取证书时间、级别。

(2) 每支球队有一名主教练和若干名球员。一名主教练只能受聘于一支球队，一名

球员只能效力于一支球队。每支球队都有自己的唯一主场场地，且场地不能共用。

(3) 足球联赛采用主客场循环制，一周进行一轮比赛，一轮的所有比赛同时进行。

(4) 一场比赛有两支球队参加，一支球队作为主队身份、另一支作为客队身份参与

比赛。一场比赛只能有一名主裁判，每场比赛有唯一的比赛编码，每场比赛都记录比分和

日期。

【概念结构设计】

根据需求分析阶段的信息，设计的实体联系图(不完整)如图 2-1 所示。

【逻辑结构设计】

根据概念结构设计阶段完成的实体联系图，得出如下关系模式(不完整)：

球队(球队编号，名称，成立时间，人数，主场地址)

球员(姓名，身份证号，出生日期，身高，家庭住址， (1))

主教练(姓名，身份证号，出生日期，资格证书号，级别， (2))

主裁判(姓名，身份证号，出生日期，资格证书号，获取证书时间，级别)

比赛(比赛编码，主队编号，客队编号，主裁判身份证号，比分，日期)

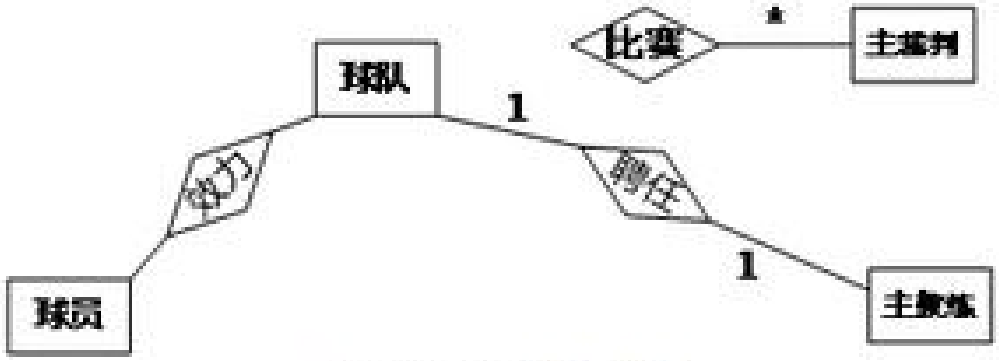


图2-1 实体联系图

问题： 2.1
(6 分)
补充图 2-1 中的联系和联系的类型。
图 2-1 中的联系“比赛”应具有的属性是哪些？
问题： 2.2
(4 分)
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)～(2)补充完整。

问题： 2.3
(5 分)
现在系统要增加赞助商信息，赞助商信息主要包括赞助商名称和赞助商编号。
赞助商可以赞助某支球队，一支球队只能有一个赞助商，但赞助商可以赞助多支球队。赞助商也可以单独赞助某些球员，一名球员可以为多个赞助商代言。请根据该要求，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 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】

某物品拍卖网站为参与者提供物品拍卖平台，组织拍卖过程，提供在线或线下交易服务。网站主要功能描述如下：

(1）拍卖参与者分为个人参与者和团体参与者两种。不同的团体也可以组成新的团体参与拍卖活动。网站记录每个参与者的名称。

(2）一次拍卖中，参与者或者是买方，或者是卖方。

(3）一次拍卖只拍出来自一个卖方的一件拍卖品；多个买方可以出价：卖方接受其中一个出价作为成交价，拍卖过程结束。

(4）在拍卖结算阶段，买卖双方可以选择两种成交方式：线下成交，买卖双方在事

先约定好的成交地点，当面完成物价款的支付和拍卖品的交付；在线成交，买方通过网上支付平台支付物价款，拍卖品由卖方通过快递邮寄给买方。

一次拍卖过程的基本事件流描述如下：

(1）卖方在网站上发起一次拍卖，并设置本次拍卖的起拍价。

(2）确定拍卖标的以及拍卖标的保留价(若在拍卖时间结束时，所有出价均低于拍卖标的保留价，则本次拍卖失败)。

(3）在网站上发布本次拍卖品的介绍。

(4）买方参与拍卖，给出竞拍价。

(5）卖方选择接受一个竞拍价作为成交价，结束拍卖。

(6）系统记录拍卖成交价，进入拍卖结算阶段。

(7）卖方和买方协商拍卖品成交方式，并完成成交。

现采用面向对象方法对系统进行分析与设计，得到如表 3-1 所示的类列表以及如图 3-1 所示的类图，类中关键属性与方法如表 3-2 所示。

表3-1 物品拍卖网站类列表

序号	类名	说明
C1	SellerRole	一次拍卖中的卖方
C2	Item	拍卖品
C3	Auction	拍卖活动
C4	Sale	拍卖结算
C5	AuctionParticipant	拍卖参与者
C6	Interchange	成交方式
C7	OneParticipant	个人参与者
C8	OfflinePay	线下成交
C9	CompositeParticipant	团体参与者
C10	OnlinePay	在线成交
C11	Bid	拍卖标的
C12	BuyerRole	一次拍卖中的买方

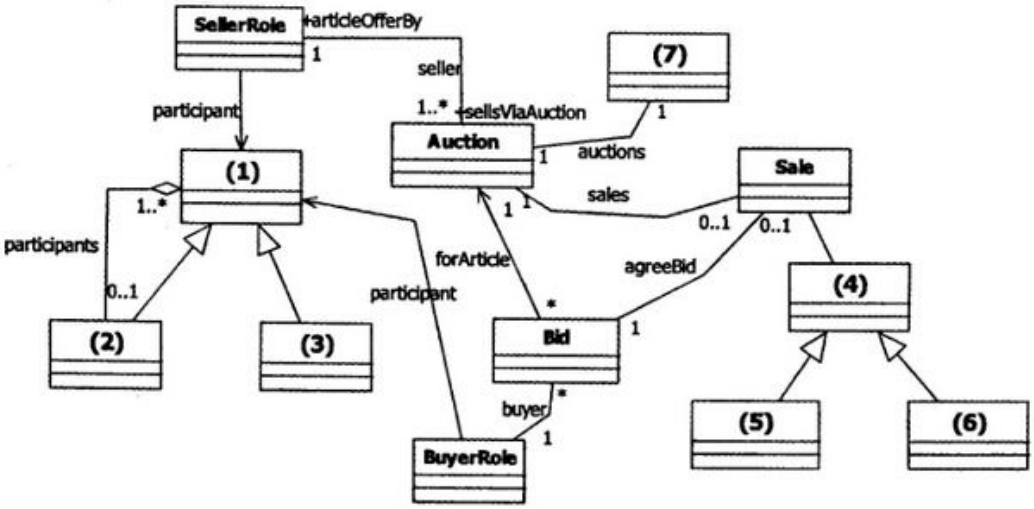


图 3-1 类图

表3-2 关键属性与方法列表

序号	名称	说明
M1	name	属性名，用户名称
M2	description	属性名，拍卖品描述
M3	minBidPrice	属性名，拍卖的起拍价
M4	agreePrice	属性名，拍卖成交价
M5	bidPrice	属性名，拍卖标的保留价
M6	address	属性名，线下成交地点
M7	sellerAccount	属性名，卖方网上支付账户名
M8	buyerAddress	属性名，买方邮寄地址
M9	placeBidForAuction	方法名，为拍卖品出竞拍价
M10	sellNewArticle	方法名，发起一次拍卖

问题： 3.1

(7 分)

根据说明中的描述，给出图 3-1 中(1)～(7)所对应的类名(类名使用表 3-1 中给出的序号)。

问题：3.2
(5 分)
根据说明中的描述，确定表 3-2 中的属性 / 方法分别属于哪个类(类名、方法 / 属性名使用表 3-1 、 3-2 中给出的序号)。
问题：3.3
(3 分)
在图 3-1 采用了何种设计模式？以 100 字以内文字说明采用这种设计模式的原因。

试题四 阅读下列说明和 C 代码，回答问题 1 至问题 3 ，将解答写在答题纸的对应栏内。

【说明】

n-皇后问题是在 n 行 n 列的棋盘上放置 n 个皇后，使得皇后彼此之间不受攻击，其规则是任意两个皇后不在同一行、同一列和相同的对角线上。

拟采用以下思路解决 n-皇后问题：第 i 个皇后放在第 i 行。从第一个皇后开始，对每个皇后，从其对应行(第 i 个皇后对应第 i 行)的第一列开始尝试放置，若可以放置，确定该位置，考虑下一个皇后；若与之前的皇后冲突，则考虑下一列；若超出最后一列，则重新确定上一个皇后的位置。重复该过程，直到找到所有的放置方案。

【 C 代码】

下面是算法的 C 语言实现。

(1) 常量和变量说明

pos ： 一维数组， pos[i]表示第 i 个皇后放置在第 i 行的具体位置

count ： 统计放置方案数

i ， j ， k ： 变量

N ： 皇后数

(2) C程序

```
#include
#include

#define N4
/*判断第k个皇后目前放置位置是否与前面的皇后冲突*/
in isplace(int pos[], int k) {
      int i;
      for (i=1; i
      if( (1) || fabs(i-k) ═══ fabs(pos[i] - pos[k])) {
            return();
      }
}
return 1;
}
int main() {
int i,j,count=1;
int pos[N+1];
//初始化位置
for (i=1; i<=N; i++) {
      pos[i]=0;
}
(2) ;
while(j>=1) {
      pos[j]= pos[j]+1;
      /*尝试摆放第i个皇后*/
      while(pos[j]<=N&& (3) _) {
            pos[j]= pos[j]+1;
      }
      /*得到一个摆放方案*/
      if(pos[j]<=N&&j═══ N) {
            printf(″方案%d: ",count++);
            for (i=1; i<=N; i++) {
                  printf("%d ",pos[i]);
            }
            printf("/n");
      }
      /*考虑下一个皇后*/
      if(pos[j]<=N&& (4) ) {
            j=j+1;
```

```
        } else {
                //返回考虑上一个皇后
                pos[j]=0;
                (5) ;
        }
}
return 1;
}
```

问题：4.1
(10 分)
根据以上说明和 C 代码，填充 C 代码中的空(1)～(5)。
问题：4.2
(2 分)
根据以上说明和 C 代码，算法采用了（6）设计策略。
问题：4.3
(3 分)
上述 C 代码的输出为：(7) 。

试题五 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】

某图书管理系统中管理着两种类型的文献：图书和论文。现在要求统计所有馆藏文献的总页码(假设图书馆中有一本 540 页的图书和两篇各 25 页的论文，那么馆藏文献的总页码就是 590 页)。采用 Visitor(访问者)模式实现该要求，得到如图 5-1 所示的类图。

【C++代码】

```
class LibraryVisitor;
class LibraryItemInterface {
public:
      (1) ;
};
class Article : public LibraryItemInterface {
private:
      string      m_title;        /* 论文名 */
      string      m_author;       /* 论文作者 */
```

```
      int   m_start_page;
      int   m_end_page;
public:
      Article( string p_author, string p_title, int p_start_page, int
p_end_page );
      int getNumber() fPages();


      void accept( Library Visitor* visitor );
};
class Book : public LibraryItemInterface {
private:
      string      m_title;        /* 书名 */
      string      m_author;       /* 作者 */
      int   m_pages;        /* 页数 */
public:
      Book( string p_author, string p_title, int p_pages );
      int getNumber() fPages();


      void accept( LibraryVisitor* visitor );
};
class LibraryVisitor {
public:
      (2);
      (3);
      virtual void printSum() = 0;
};
class LibrarySumPrintVisitor : public LibraryVisitor { /* 打印总页数
*/
private:
      int sum;
public:
      LibrarySumPrintVisitor();
      void visit( Book* p_book );


      void visit( Article* p_article );


      void printSum();
};
/* visitor.cpp */
int Article : : getNumber() fPages()
```

```
{
      retum m_end_page - m_start_page;
}
void Article::accept( LibraryVisitor* visitor )
{
      (4);
}


Book: : Book( string p_author, string p_title, int p_pages )
{
      m_title           = p_title;
      m_author    = p_author;
      m_pages           = p_pages;
}
int Book::getNumberOfPages()
{
      return(m_pages);
}


void Book::accept( LibraryVisitor* visitor )
{
      (5);
}


/* 其余代码省略 */
```

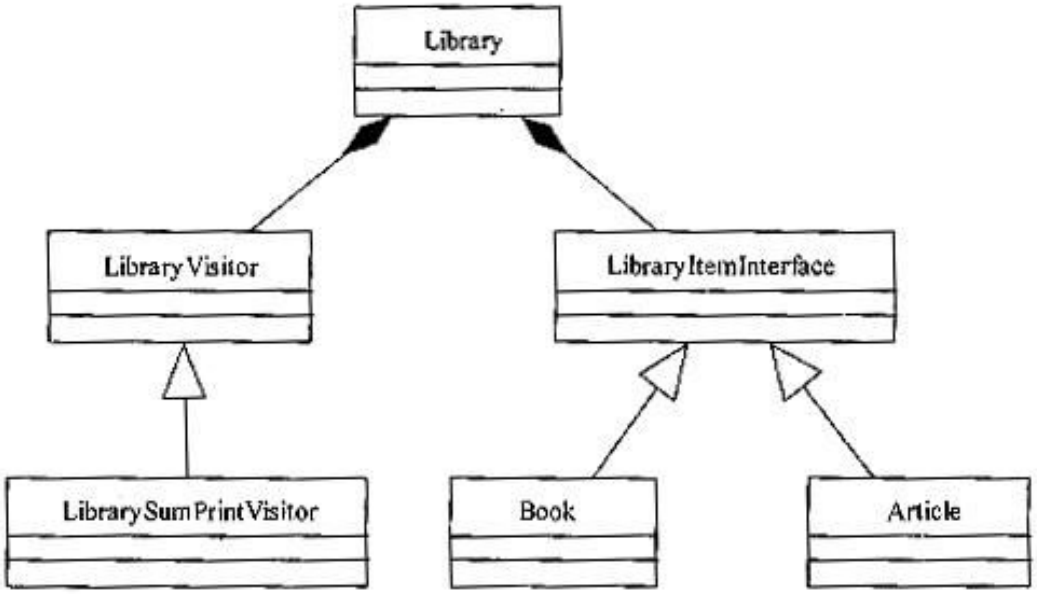


图 5-1　Visitor 模式类图

问题：5.1
(15 分)
阅读上述说明和 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　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】

某图书管理系统中管理着两种类型的文献：图书和论文。现在要求统计所有馆藏文献的总页码(假设图书馆中有一本 540 页的图书和两篇各 25 页的论文，那么馆藏文献的总页码就是 590 页)。采用 Visitor(访问者)模式实现该要求，得到如图 6-1 所示的类图。

【Java 代码】

```
import java.util.*;
interface LibraryVisitor {
  (1)    :
  (2)      :
void printSum();
```

```
}
class LibrarySumPrintVisitor implements LibraryVisitor { //打印总页数
private int sum = 0;
public void visit(Book p_book) {
sum = sum + p_book.getNumberOfPages();
public void visit(Article p_article) {
sum = sum + p_article.getNumberOfPages0;
}
public void printSum(){
System.out.println("SUM = " + sum);
}
}
interface LibraryItemInterface {
(3) ;
}
class Article implements LibraryItemInterface{
private String m_title; //价仑文名 。
private String m_author; //仑文作者
private int m_start_page;
private int m_end_page;
public Article(String p_author, String p_title,int p_start_page,int
p_end_page){
m_title=p_title;
m_author= p_author;
m_end_page=p_end_page;
}
public int getNumbelOfPages(){
rctum m_end_page - m_start_page;
}
public void acccpt(LibraryVisitor Visitor){
(4) :
}
}
class Book implements LibraryItemInterface{
private String m_title; //书名
private String m_author; //书作者
private int m_pages; //页教
public Book(String p_author, String p_title,int p_ pages){
m_title= p_title;
m_author= p_author;
m_pages= p_pages;
}
public int getNumberOfPages(){
```

```
return m_pages;
}
public void accept(LibraryVisitor visitor){
(5)___;
}
}
```

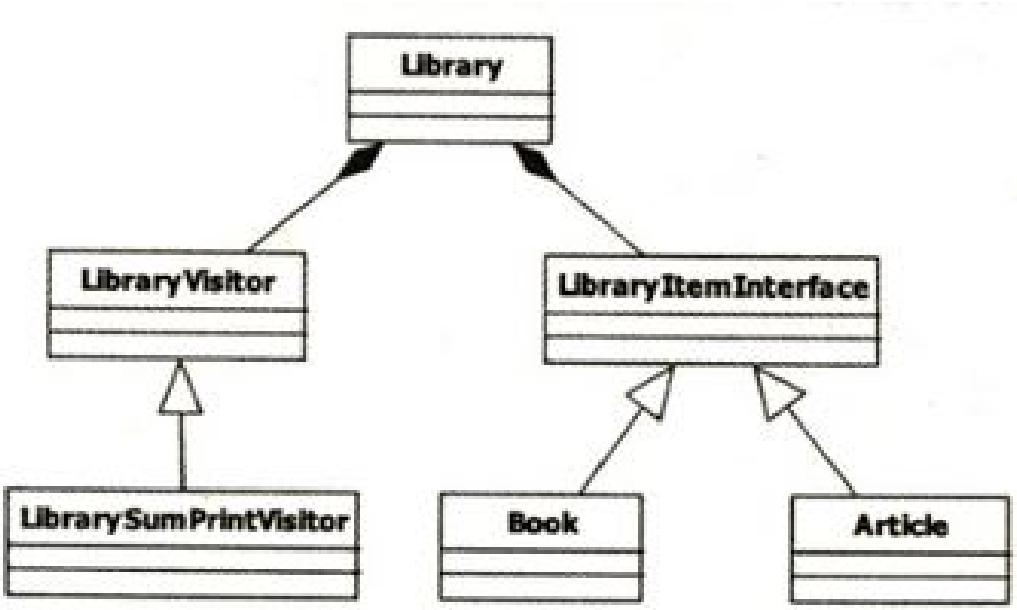


图 6-1 Visitor 模式类图

问题： 6.1
(15 分)
阅读上述说明和 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：教师；

E2：学生。

解析：本题考查采用结构化方法进行系统分析与设计，主要考查数据流图(DFD)的应用，是比较传统的题目，与往年考点类似，要求考生细心分析题目中所描述的内容。

DFD 是一种便于用户理解、分析系统数据流程的图形化建模工具，是系统逻辑模型的重要组成部分/上下文 DFD（顶层 DFD)通常用来确定系统边界，将待开发系统本身看作一个大的加工(处理)，然后根据谁为系统提供数据流，谁使用系统提供的数据流， 来确定外部实

体。建模出的上下文 DFD 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,。在上下文确定的系统外部实体以及与外部实体的输入输出数据流的基础上，建模 0 层 DFD，将上下文 DFD 中的加工进一步分解成多个加工， 识别这些加工的输入输出数据流，使得所有上下文 DFD 中的输入数据流，经过这些加工之后变换成上下文 DFD 的输出数据流。根据 0 层 DFD 中加工的复杂程度进一步建模加工的内容。

在建模分层 DFD 时，根据需求情况可以将数据存储建模在不同层次的 DFD 中，注意要在绘制下层数据流图时要保持父图与子图平衡。父图中某加工的输入输出数据流必须与它的子图的输入输出数据流在数量和名字上相同，或者父图中的一个输入(或输出) 数据流对应于子图中几个输入(或输出)数据流，而子图中组成这些数据流的数据项全体正好是父图中的这一个数据流。

本问题考查上下文 DFD,要求确定外部实体。考察系统的主要功能，不难发现，针对系统与考试有关的主要功能，涉及到教师和学生，系统管理员不在与考试有关的主要功能中涉及，另外没有提到其他与系统交互的外部实体。根据描述(1) 中“教师制定试 题等考试信息”等信息，描述(2) 中“根据教师设定的考试信息，在考试有效时间内向学生显示考试说明和题目”，从而即可确定 E1 为“教师”实体， E2 为“学生”实体。

D1 ：试题(表)或题目和答案(表)

D2 ：学生信息(表)

D3 ：考试信息(秦)

D4 ：解答结果(表)

解析：本问题要求确定 0 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，说明中(1) 中“教师制定试题(题目和答案)，制定考试说明、考试时间和提醒时间等考试信息，录入参加考试的学生信息，并分别进行存储”，可知 D1 、 D2 和 D3 为试题、 学生信息和考试信息，再从图 1-2 中流入 D2 的数据流名称“学生信息数据”，确定 D2 是学生信息，流入 D1 的数据流名称为“试题”，确定 D1 为试题，流入 D3 的数据流名称为考试信息，确定 D3 为考试信息。说明中(3) 根据答案对接收到的解答数据进行处理，然后将解答结果进行存储，确定 D4 是解答结果。参考其他描述中对数据存储的使用更多说明，进一步确定 D1 D4 满足上述分析。

解析：本问题要求补充缺失的数据流及其起点和终点。通过不同层的 DFD 以及说明中描述和图之间的对应#系加以确定。首先对照图 1-1 和图 1-2 的输入、输出数据流，发现数据流的数量和名称均相同，所以，需进一步考查说明中的功能描述和图 1-1 中的数据流的对应关系，以确定缺失的是加工之间还是加工与数据存储之间的数据流。

说明(2) 显示并接收解答，需要“根据教师设定的考试信息，在考试有效时间内向学生显示考试说明和题目”，对照图 1-2 可以看出，加工 2 缺少所要显示的题目的输入源，即缺

失输入流“题目”，题目存储于数据存储“试题”中，因此，缺少的数据流为从题目(D1)到加工 2 显示并接收解答的题目。说明(3) 处理解答，需要“根据答案对 接收到的解答数据进行处理”，对照图 1-2 可以看出，加工 3“处理解答”缺少输入流“答案”，而从说明(1) 中可以看出“答案”存储在试题(题目和答案)数据存储中(D1)， 因此确定缺失的一条数据流“答案”，从 D1 或试题到加工 3 或处理解答。

分解为加工：发送通知和加工：创建通知

解析：

本问题考查建模分层 DFD 时的分解粒度。在说明(6) 发送通知中，“根据成绩报告数据，创建通知数据并将通知发送给学生;根据成绩单数据，创建通知数据并将通知发送给教师。”说明功能(6) 发送通知包含创建通知并发送给学生或老师。在图 1-2 中建模为一个加工，完成的功能是依据不同的输入数据流创建通知，然后发送给相应的外部实体老师或学生,因此为了进一步清晰每个加工的职责，需对图 1-2 中原有加工 6 进 行分解，分解为“创建通知”和“发送通知”。创建通知针对输入数据流“报告数据”和 “成绩单数据”，这两条数据流保持原有的起点，终点即为创建通知。创建通知产生出“通知数据’“通知数据”作为加工“发送通知”的输入流，进一步根据通知数据是针对哪个外部实体而发送“通知”给相应的学生或者教师。至此，对图 1-2 中原有加工 6 的分解完成。

数 据 流	起 点	终 点
答案	D1 或试题（表）或题目和答案（表）	3 或处理解答
题目	D1 或试题（表）或题目和答案（表）	2 或显示并接收解答

数 据 流	起 点	终 点
报告数据	生成成绩报告	创建通知
成绩单数据	生成成绩单	创建通知
通知数据	创建通知	发送通知

试题二 答案： 解析：

比赛联系应具有的属性包括：比赛编码，比分，日期。

解析：本题考查数据库概念结构设计及向逻辑结构转换的掌握。

此类题目要求考生认真阅读题目，根据题目的需求描述，给出实体间的联系。

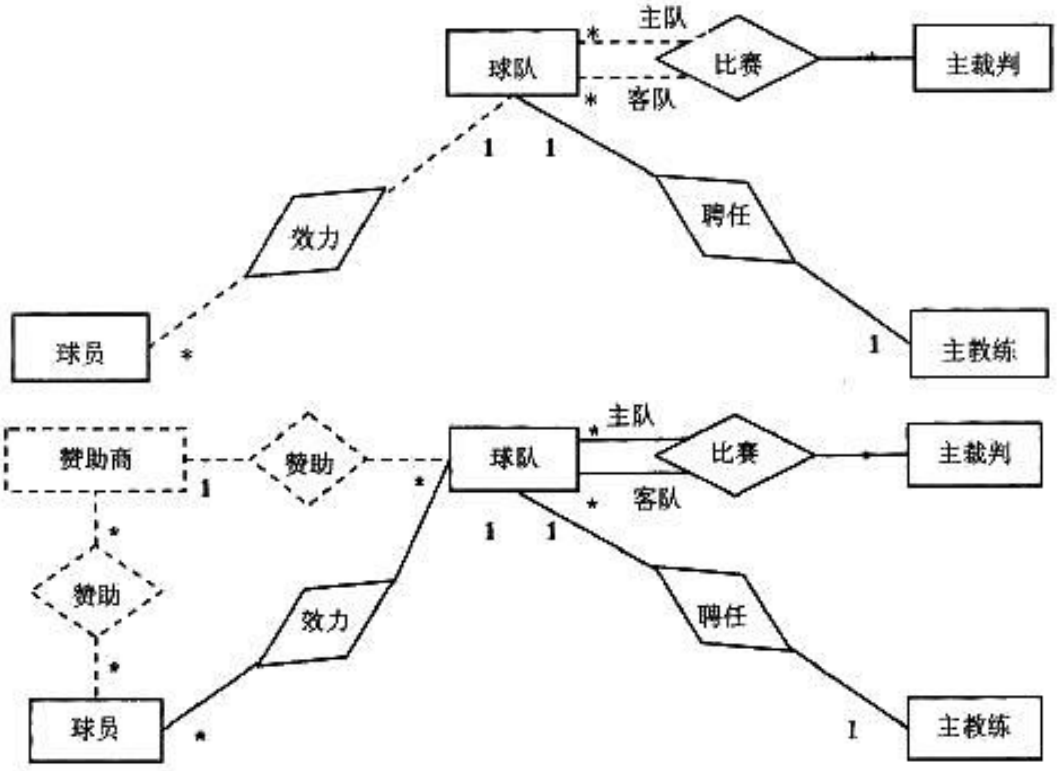
根据题意由“一名球员只能效力于一支球队”可知球队和球员之间为 1:*联系。由“一场比赛有两支球队参加，一支球队作为主队身份、另一支作为客队身份参与比赛”可知球队分别按照“主队”和“客队”两种角色参与“比赛”的联系。“比赛”应具有的属性：比赛编码，出分和日期。

(1）球队编号；

(2）球队编号。

解析：根据问题 1 分析可知球队和球员之间为 1:*联系，所以在球员关系里应该包括球队的主键，即“球队编号”。根据“每支球队有一名主教练，一名主教练只能受聘于一支球队”可知球队和教练之间为 1:1 联系，而球队关系已经给定，所以需要在主教练关系中包含球队的主键，即“球队编号”。

解析：根据题意由“赞助商可以赞助某支球队，一支球队只能有一个赞助商，但赞助商可以赞助多支球队”可知赞助商和球队之间为 1:*联系。由“赞助商也可以单独赞助某些球员，一名球员可以为多个赞助商代言”可知赞助商和球员之间为联系。



试题三 答案： 解析：

解析：本题属于经典的考题，主要考查面向对象分析方法与设计的基本概念。在建模方面， 本题中只涉及到了 UML 类图。类图上的考点也是比较常规的对类的识别以及类中属性及方法的确定，题目难度不大。

图 3-1 共需要确定 7 个类，可以先从图中几个特殊关系处入手，即(1） （ 3）和(4） （ 6)。

先来分析(1) （ 3）,这是一个继承+聚集的结构，而且联系的名称“participants” 是一个比较明显的提示，说明这个层次结构是与【说明】中的功能描述(1) 相对应的。 参考表 3-1，与之相关的类是C5 (AuctionParticipant)、 C7 (OneParticipant)和C9 (CompositeParticipant)。 C7 、 C9 是特殊的参与者，所以(1) 处应该为C5; (2) 处应该为C9,这个聚集关系针对着【说明】中的“不同的团体也可以组成新的团体参与拍卖活动”需求；(3) 处为C7 。

结合【说明】和表 3-1,另外一组具有“一般-特殊”关系的类只有C6 (Interchange)，C8 (OfflinePay)和C10 (OnlinePay)。显而易见， C8 和C10 是C6 的两种具体方式，所以(4) 处应该为C6，(5) 、(6) 处分别为C8 和C10 。

这样(7) 处对应的类只能是 Item 了。结合【说明】和表 3-1 可知，(7) 处对应的类表达的应该是拍卖中的拍卖品，所以(7) 处应该是C2 。

在图 3-1 中使用了 Composite 模式。

以树形结构表示个人参与者和团体参与者之间的“部分-整体”关系，使得对单个对象和组合对象的使用具有一致性。

解析：在【说明】部分有一个很明显的提示：“拍卖参与者分为个人参与者和团体参与者两种。不同的团体也可以组成新的团体参与拍卖活动”。这里很清晰地表达了一种“部分-整体”的层次关系，这种关系非常适合于采用Composite (组合)设计模式来表达。

Composite 设计模式将对象组合成树形结构以表示“部分-整体”的层次结构。 Composite 使得用户对单个对象和组合对象的使用具有一致性。

	类		类
(1)	C5 (AuctionParticipant)	(5)	C8 (OfflinePay)
(2)	C9 (CompositeParticipant)	(6)	C10 (OnlinePay)
(3)	C7 (OneParticipant)	(7)	C2 (Item)
(4)	C6 (Interchange)		

注：(5) 和 (6) 的类名可互换。

属性/方法序号	所属类的序号	属性/方法序号	所属类的序号
M1	C5	M6	C8
M2	C2	M7	C10
M3	C3	M8	C10
M4	C4	M9	C12
M5	C11	M10	C1

试题四 答案： **解析：** (1) pos[i]==pos[k］或其等价形式

(2) j=1

(3) ！ isplace(pos,j）或其等价形式

(4)j(5) j=j-1 或其等价形式

解析：本题考查算法设计和 C 程序设计语言的相关知识。

此类题目要求考生认真阅读题目，理解算法思想，并思考将算法思想转化为具体的程序设计语言的代码。

根据题干描述。空(1) 所在的代码行判断皇后合法放置的约束条件，即不在同一行，这通过把第 i 个皇后放在第 i 行实现，条件“fabs(i-k)==fabs(pos[i]-pos[k])”判断的是当前摆放的皇后是否与之前摆放的皇后在同一对角线上。因此，空(1) 判断的是当 前摆放的皇后是否和之前摆放的皇后在同一列上，即应填入“pos[i]==pos[k]”。

根据算法思想和主函数上下文，空(2) 处应该考虑第 1 个皇后，即初始化 j 为 1， 空(2) 填写“j=1”。空(3) 所在的行是判断放置第 j 个皇后的位置是否合适，“pos[j] <=N”表示在该行的合法列上，但还需要进一步判断是否与前面的皇后有冲突，根据满足条件后的语句，尝试放入下一列，因此空(3) 处填入“!isplace(pos， j)”。根据前面的注释，空(4)所在的行是考虑下一个皇后，其条件是，当前皇后找到了合适的位置，而且还存在下一个皇后，因此空(4)处应填入“j

(6) 回溯法

解析：从上述题干的叙述和 C 代码很容易看出，从第一个皇后开始，对每个皇后总是从第个位置开始尝试，找到可以放置的合法位置；若某个皇后在对应的行上没有合法位置， 则回溯到上一个皇后，尝试将上一个皇后放置另外的位置。这是典型的深度优先的系统搜索方式，即回溯法的思想。

(7)

方案 1： 2413；

方案 2： 3142 。

试题五 答案： 解析： (1) virtual void accept(LibraryVisitor*visitor)=0

(2) virtual void visit(Book* p_book)=0

(3) virtual void visit(Article* p_article)=0

(4) visitor->visit(this)

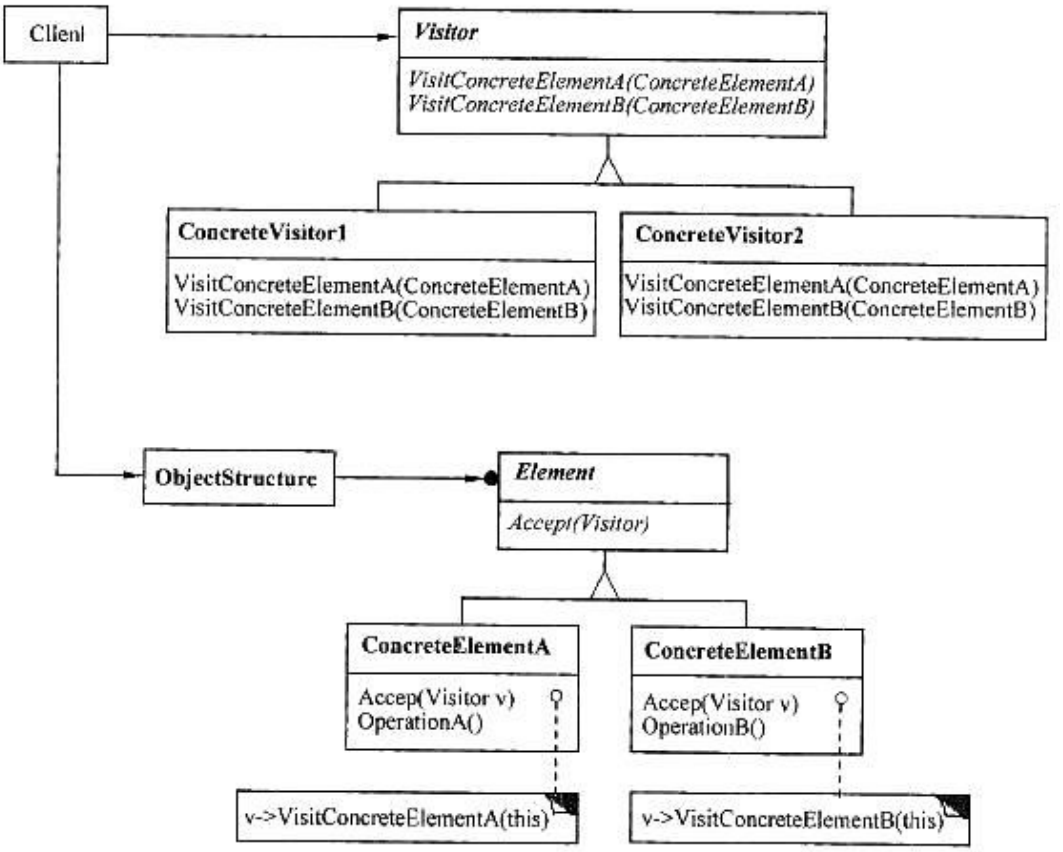
(5) visitor->visit(this)

解析：本题考查 Visitor（访问者)模式的基本概念和应用。

访问者模式是行为设计模式中的一种。行为模式不仅描述对象或类的模式，还描述它们之间的通信模式。这些模式刻画了在运行时难以跟踪的复杂的控制流。

访问者模式表示一个作用于某对象结构中的各元素的操作。它使在不改变各元素的类的前提下可以定义作用于这些元素的新操作。此模式的结构图如下图所示。

Visitor（访问者)为该对象结构中 ConcreteElement 的每一个类声明一个 Visit 操作。该操作的名字和特征标识了发送 Visit 请求给该访问者的哪个类。这使得访问者可以确定正被访问元素的具体的类。这样访问者就可以通过该元素的特定接口直接访问它。
ConcreteVisitor（具体访问者)实现每个有 Visitor 声明的操作，每个操作实现本算法的一部分，而该算法片段乃是对应于结构中对象的类。 ConcreteVisitor 为该算法提供了上下文并存储它的局部状态。这一状态常常在遍历该结构的过程中累积结果。
Element（元素)定义以一个访问者为参数的 Accept 操作。
ConcreteElement（具体元素)实现以一个访问者为参数的 Accept 操作。
ObjectStructure（对象结构)能枚举它的元素；可以提供一个高层的接口以允许该访问者访问它的元素；可以是一个组合或者一个集合，如一个列表或一个无序集合。
本题中类 Library 对应着上图中的 Client ， LibraryVisitor 对应着 Visitor ，LibrarySumPrintVisitor 对应着 ConcreteVisitor 。 Libraryltemlnterface 对应着上图中的元素部分。下面可以结合程序代码来完成程序填空了。
Libraryltemlnterface 在本题中充当着 Element 的作用,其中应定义以一个访问者为参数的 Accept 操作。对照其两个子类 Article 和 Book 的代码，可以得知该操作的原型是 void accept(LibraryVisitor* visitor)。由于该操作的具体实现在子类 Article 和 Book 中，所以这里适用于采用 C++中的纯虚拟函数来实现。由此可以得知，(1) 处应填写“virtual voidaccept(LibraryVisitor* visitor) = 0”。
(2) 和(3) 空与类 LibraryVisitor 有关。由前文分析已知， LibraryVisitor 对应着访问者模式中的 Visitor,其作用是为类 LibrarySumPrintVisitor 声明 Visit 操作。类 LibrarySumPrintVisitor 需要访问两种不同的元素，每种元素应该对应不同的 visit 操作。 再结合类 LibrarySumPrintVisitor 的定义部分，可以得知(2) 和(3) 处应给出分别以 Book 和 Article 为参数的 visit 方法，同样采用纯虚拟函数机制。因此(2) 和(3) 处分别为“virtualvoidvisit(Book* p_book) = 0”、“virtualvoidvisit(Article* p_article) = 0”。
(4) 和(5) 处考查的是 accept 接口的实现。由访问者模式的结构图可以看出，在 Book 和 Article 中 accept 方法的实现均为 visitor->visit(this)。



试题六 答案：　解析： (1) void visit(Book p_book)

(2) void visit(Article p_article)

(3) void accept(LibraryVisitor visitor)

(4) visitor.visit(this)

(5) visitor.visit(this)

解析：本题考查 Visitor（访问者)模式的基本概念和应用。

访问者模式是行为设计模式中的一种。行为模式不仅描述对象或类的模式，还描述它们之间的通信模式。这些模式刻画了在运行时难以跟踪的复杂的控制流。

访问者模式表示一个作用于某对象结构中的各元素的操作。它使在不改变各元素的 类的前提下可以定义作用于这些元素的新操作。此模式的结构图如下图所示。

Visitor（访问者)为该对象结构中 ConcreteElement 的每一个类声明一个 Visit 操作。该操作的名字和特征标识了发送 Visit 请求给该访问者的哪个类。这使得访问者可以确定正被访问元素的具体的类。这样访问者就可以通过该元素的特定接口直接访问它。

ConcreteVisitor（具体访问者)实现每个有 Visitor 声明的操作，每个操作实现本算法的

一部分，而该算法片段乃是对应于结构中对象的类。 ComcreteVisitor 为该算法提供了上下文并存储它的局部状态。这一状态常常在遍历该结构的过程中累积结果。

Element （元素)定义以一个访问者为参数的 Accept 操作。

ConcreteElement （具体元素)实现以一个访问者为参数的 Accept 操作。

ObjectS-tructure （对象结构)能枚举它的元素；可以提供一个高层的接口以允许该访问者访问它的元素；可以是一个组合或者一个集合，如一个列表或一个无序集合。

本题中类 Library 对应着上图中的 Client ， LibraryVisitor 对应着 Visitor ，LibrarySumPrintVisitor 对应着 ConcreteVisitor 。 Libraryltemlnterface 对应着上图中的元素部分。下面可以结合程序代码来完成程序填空了。

(1）和(2）空与类 LibraryVisitor 有关。由前文分析已知， LibraryVisitor 对应着访问者模式中的 Visitor ，其作用是为类 LibrarySumPrintVisitor 声明 Visit 操作。类 LibrarySumPrintVisitor 需要访问 2 种不同的元素，每种元素应该对应不同的 visit 操作。 再结合类 LibrarySumPrintVisitor 的定义部分，可以得知(2）和(3）处应给出分别以 Book 和 Article 为参数的 visit 方法。因此(1）和(2）处分别为“voidvisit(Bookp_book)”、“voidvisit(Articlep_article)”。

Libraryltemlnterface 在本题中充当着 Element 的作用，其中应定义以一个访问者为参数的 Accept 操作。对照实现该接口的两个类 Article 和 Book 的代码，可以得知该操作的原型是 voidaccept(LibraryVisitorvisitor)。由此可以得知，(3)处应填写“ voidaccept (LibraryVisitorvisitor）”。

(4）和(5）处考查的是 accept 接口的实现。由访问者模式的结构图可以看出，在 Book 和 Article 中 accept 方法的实现均为 visitor.visit(this)。