**全国计算机技术与软件专业技术资格（水平）考试**

# 2015年上半年 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卡** |

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6道题，试题一至试题四是必答题，试题五至试题六选答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. 仿照下面的例题，将解答写在答题纸的对应栏内。

例题

2015 年上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“5月23日”，故在答题纸的对应栏内写上“5”和“23”（参看下表）。

|  |  |
| --- | --- |
| 例题 | 解答栏 |
| （1） | 5 |
| （2） | 23 |

**试题一（共15分）**

阅读下列说明和数据流图，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。

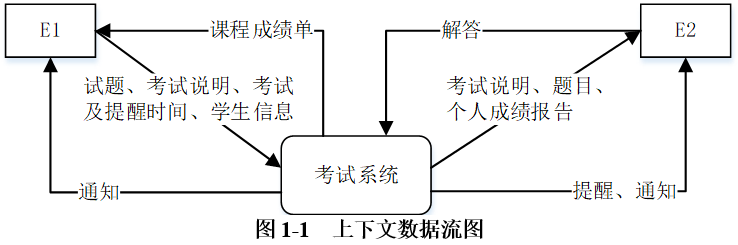
（3）处理解答。根据答案对接收到的解答数据进行处理，然后将解答结果进行存储。

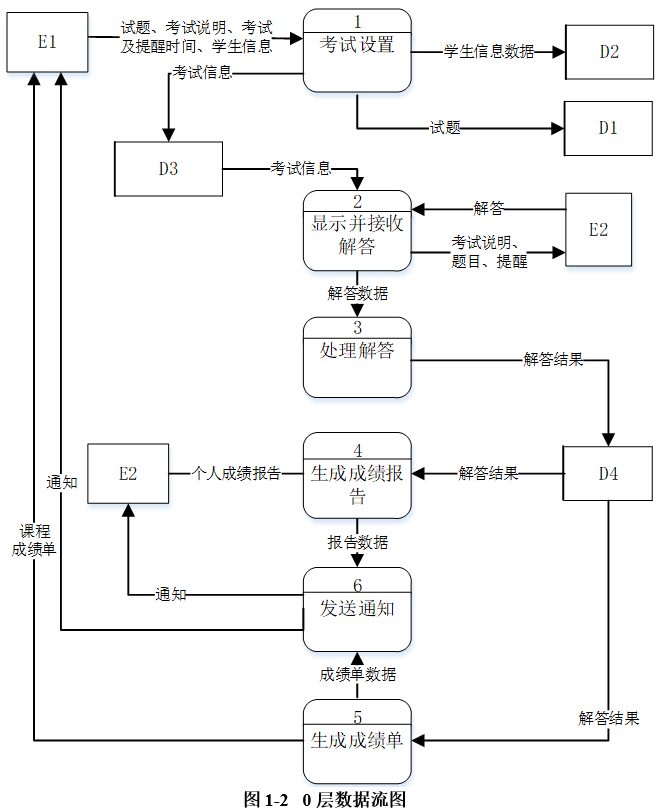
（4）生成成绩报告。根据解答结果生成学生个人成绩报告，供学生查看。

（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。





【问题1】（2分）

使用说明中的词语，绘出图1-1中的实体E1〜E2的名称。

【问题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4】（5分）

在图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（共15分）**

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。

【需求分析结果】

（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。

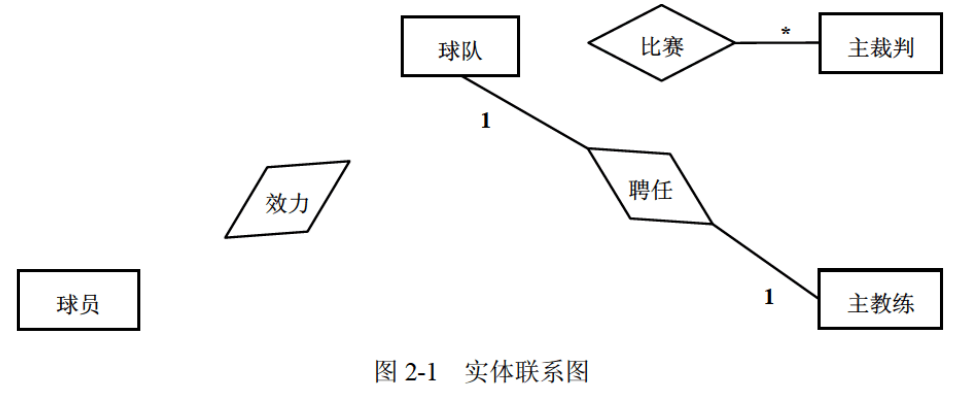
（2）每支球队有一名主教练和若干名球员。一名主教练只能受聘于一支球队，一名球员只能效力于一支球队。每支球队都有自己的唯一主场场地，且场地不能共用。

（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。



【逻辑结构设计】

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）：

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）

【问题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】（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（共15分）**

阅读下列说明和UML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。

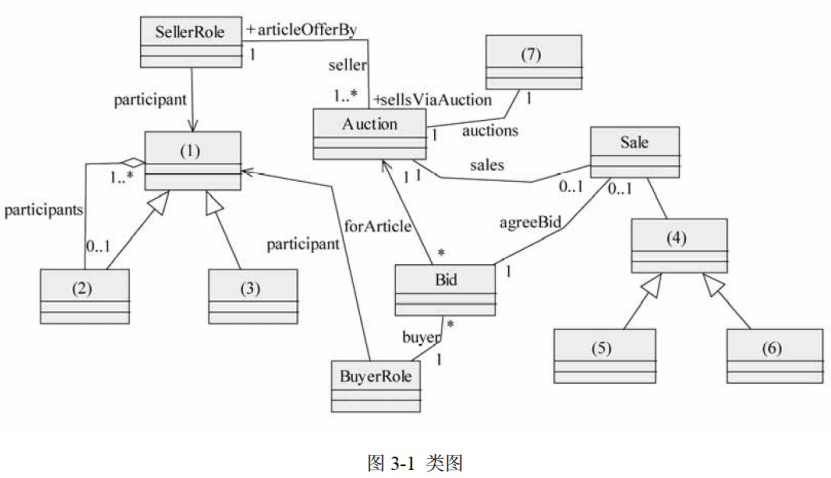
（5）卖方选择接受一个竞拍价作为成交价，结束拍卖。

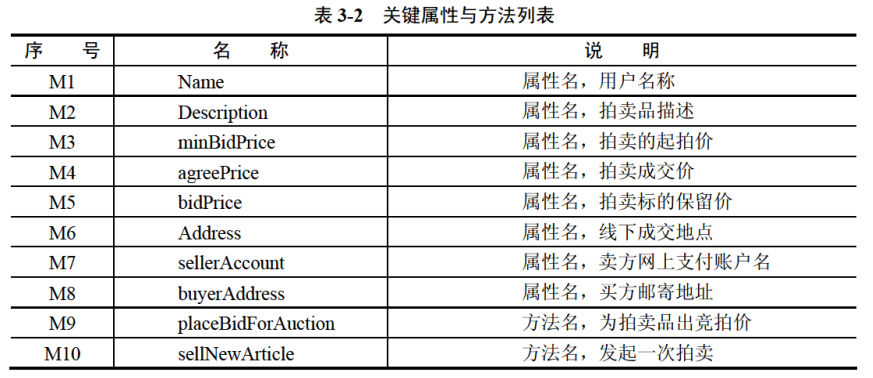
（6）系统记录拍卖成交价，进入拍卖结算阶段。

（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。







【问题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（共15分）**

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】

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 <stdio.h>

#include <math.h>

#define N 4

/\* 判断第k个皇后目前放置位置是否与前面的皇后冲突 \*/

int isplace(int pos[], int k) {

int i;

for(i = 1; i < k; i ++ )

if ( (1) || fabs(i - k) == fabs(pos[i] - pos[k])

return 0;

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3】（3分）

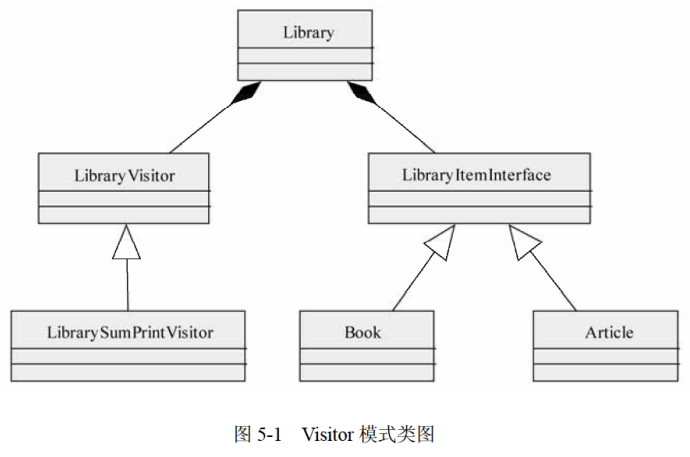
上述C代码的输出为： （7）

**试题五（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】

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 getNumberOfPages();

void accept(LibraryVisitor\* 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 getNumberOfPages();

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::getNumberOfPages() {

return 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) ; }

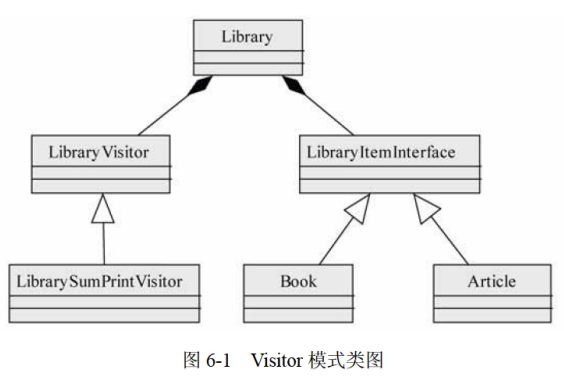
// 其余代码省略

**试题六（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】

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.getNumberOfPages();

}

public void printSum() {

System.out.println("SUM = " + sum);

}

}

interface LibraryItemInterface {

(3) ;

}

class Article implements LibraryItemInterface {

private String m\_title; // 论文名

private String m\_author; // 论文作者

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\_start\_page = p\_start\_page;

m\_end\_page = p\_end\_page;

}

public int getNumberOfPages() {

return m\_end\_page - m\_start\_page;

}

public void accept(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) ;

}

}