**全国计算机技术与软件专业技术资格（水平）考试**

# 2019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

2019 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9日”，故在答题纸的对应栏内写上“11”和“9”（参看下表）。

|  |  |
| --- | --- |
| 例题 | 解答栏 |
| （1） | 11 |
| （2） | 9 |

**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】

某公司欲开发一款二手车物流系统，以有效提升物流成交效率。该系统的主要功能是：

（1）订单管理：系统抓取线索，将车辆交易系统的交易信息抓取为线索。帮买顾问看到有买车线索后，会打电话询问买家是否需要物流，若需要，帮买顾问就将这个线索发起为订单并在系统中存储，然后系统帮助买家寻找物流商进行承运。

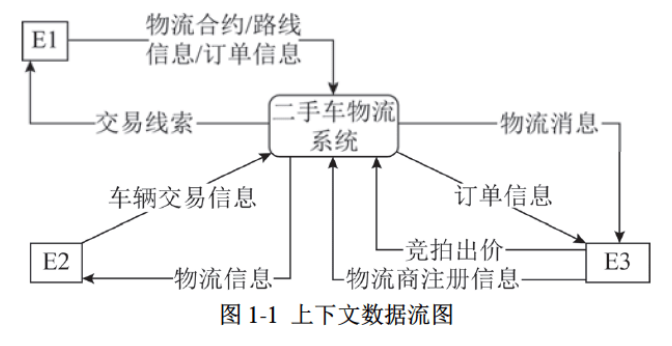
（2）路线管理：帮买顾问对物流商的路线进行管理，存储的路线信息包括路线类型、物流商、起止地点。路线分为三种，即固定路线、包车路线、竞拍体系，其中固定路线和包车路线是合约制。包车路线的发车时间由公司自行管理，是订单的首选途径。

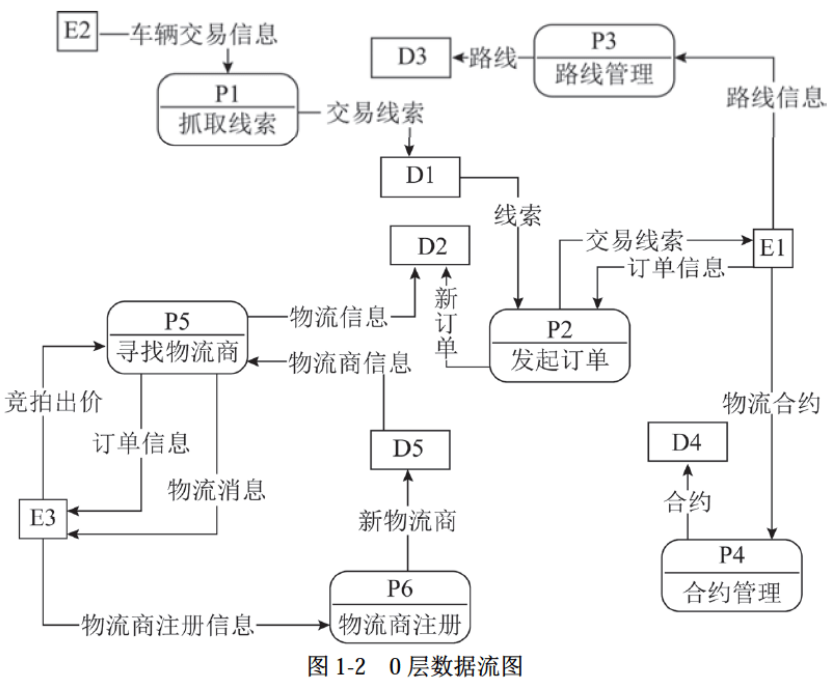
（3）合约管理：帮买顾问根据公司与物流商确定的合约，对合约内容进行设置，合约信息包括物流商信息、路线起止城市、价格、有效期等。

（4）寻找物流商：系统根据订单的类型（保卖车、全国购和普通二手车）、起止城市、需要的服务模式（买家接、送到买家等）进行自动派发或以竞拍体系方式选择合适的物流商。即:有新订单时，若为保卖车或全国购，则直接分配到竞拍体系中；否则，若符合固定路线和/或包车路线，系统自动分配给合约物流商，若不符合固定路线和包车路线，系统将订单信息分配到竞拍体系中。竞拍体系接收到订单后，将订单信息推送给有相关路线的物流商，物流商对订单进行竞拍出价，最优报价的物流商中标。最后，给承运的物流商发送物流消息，更新订单的物流信息，给车辆交易系统发送物流信息。

（5）物流商注册：物流商账号的注册开通。

现采用结构化方法对二手车物流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】（3分）

使用说明中的词语，给出图1-1中的实体E1〜E3的名称。

【问题2】（5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】（4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）

根据说明，采用结构化语言对“P5：寻找物流商”的加工逻辑进行描述。

**试题二（共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】

某公司拟开发一套新入职员工的技能培训管理系统，以便使新员工快速胜任新岗位。该系统的部分功能及初步需求分析的结果如下所述：

【需求分析结果】

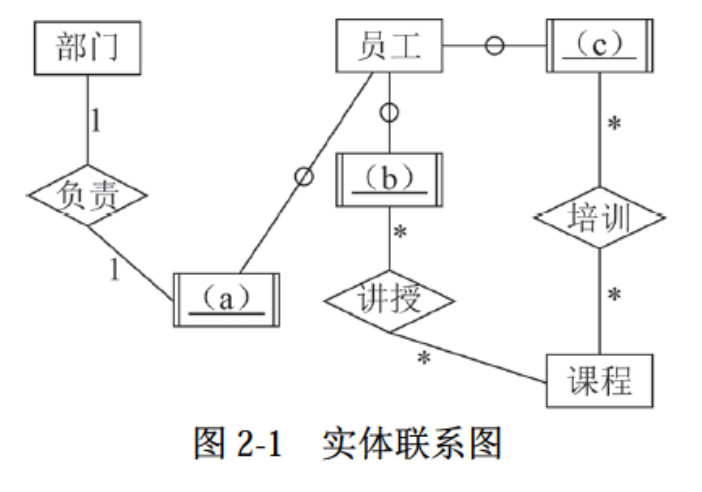
（1）部门信息包括部门号、名称、部门负责人、电话等，部门号唯一标识部门关系中的每一个元组，一个部门有多个员工，但一名员工只属于一个部门；每个部门只有一名负责人，负责部门工作。

（2）员工信息包括员工号、姓名、部门号、岗位、基本工资、电话、家庭住址等，其中员工号是唯一标示员工关系中的每一个元组；岗位有新入职员工、培训师、部门负责人等；不同的岗位设置不同的基本工资。新入职员工要选择多门课程进行培训，并通过考试取得课程的成绩。一名培训师可以讲授多门课程，一门课程可以由多名培训师讲授。

（3）课程信息包括课程号、课程名称、学时等，其中课程号唯一标识课程关系的每一个元组。

【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）：

部门（部门号，部门名，部门负责人，电话）

员工（员工号，姓名，部门号， （d） ，电话，家庭住址）

课程（ （e） ，课程名称，学时）

讲授（课程号，培训师，培训地点）

培训（课程号， （f） ）

【问题1】（5分）

（1）补充图2-1中的空（a）〜（c）。

（2）图2-1中是否存在缺失联系，若存在，则说明所缺失的联系和联系类型。

【问题2】（3分）

根据题意，将关系模式中的空（d）〜（f）补充完整。

【问题3】（5分）

（1）员工关系模式的主键为 （g） ，外键为 （h） ；

（2）讲授关系模式的主键为 （i） ，外键为 （j） 。

【问题4】（2分）

员工关系是否存在传递依赖？用100字以内的文字说明理由。

**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】

某牙科诊所拟开发一套信息系统，用于管理病人的基本信息和就诊信息。诊所工作人员包括：医护人员（DentalStaff）、接待员（Receptionist）和办公人员（OfficeStaff）等。系统主要功能需求描述如下：

1．记录病人基本信息（Maintain patient info）。初次就诊的病人，由接待员将病人基本信息录入系统。病人基本信息包括病人姓名、身份证号、出生日期、性别、首次就诊时间和最后一次就诊时间等。每位病人与其医保信息（MedicalInsurance）关联。

2．记录就诊信息（Record office visit info）。病人在诊所的每一次就诊，由接待员将就诊信息（Office Visit）录入系统。就诊信息包括就诊时间、就诊费用、支付代码、病人支付费用和医保支付费用等。

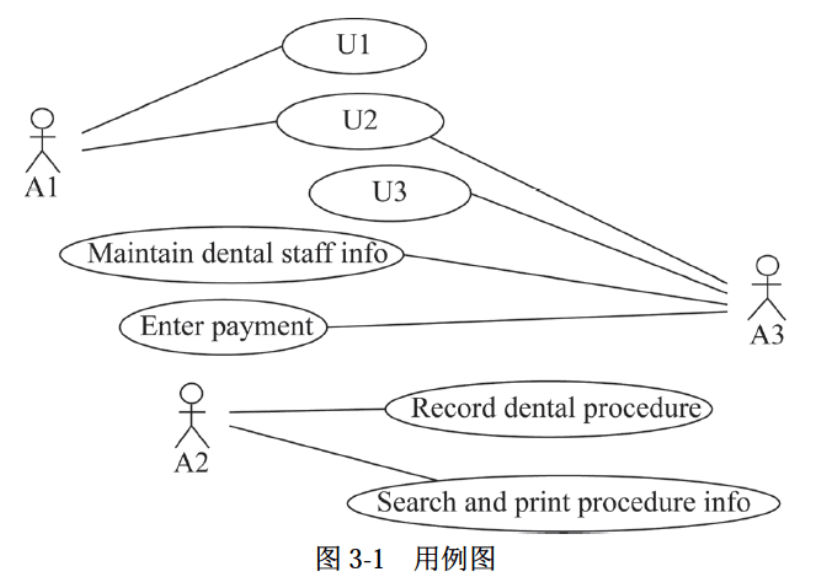
3．记录治疗信息（Record dental procedure）。病人在就诊时，可能需要接受多项治疗，每项治疗（Procedure）可能由多位医护人员为其服务。治疗信息包括：治疗项目名称、治疗项目描述、治疗的牙齿和费用等。治疗信息由每位参与治疗的医护人员分别向系统中录入。

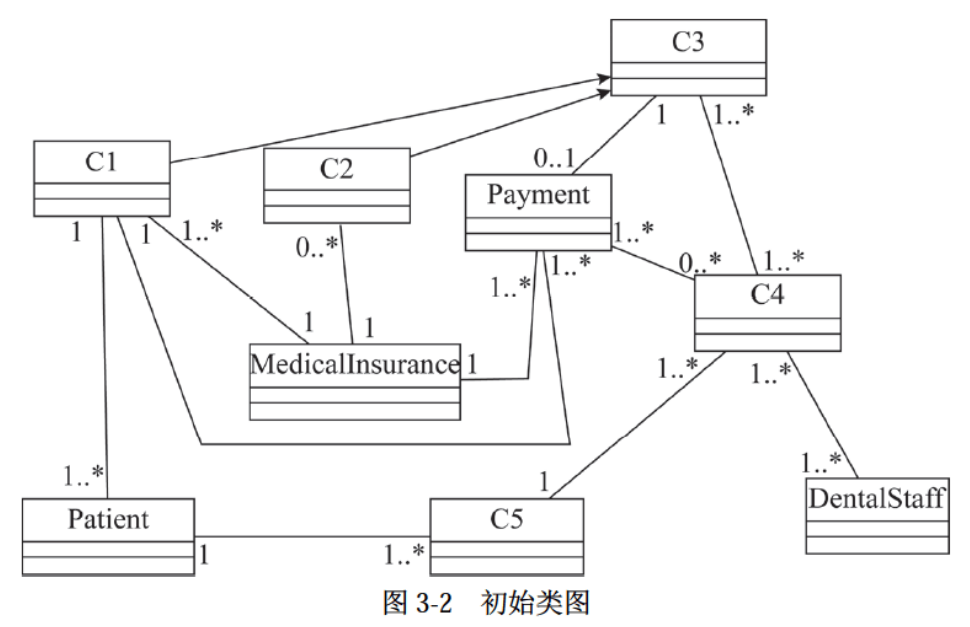
4．打印发票（Print invoices）。发票（Invoice）由办公人员打印。发票分为两种：给医保机构的发票（InsuranceInvoice）和给病人的发票（PatientInvoice）。两种发票内容相同，只是支付的费用不同。当收到治疗费用后，办公人员在系统中更新支付状态（Enterpayment）。

5．记录医护人员信息（Maintain dental staff info）。办公人员将医护人员信息录入系统。医护人员信息包括姓名、职位、身份证号、家庭住址和联系电话等。

6.医护人员可以查询并打印其参与的治疗项目相关信息（Search and print procedure info）。

现采用面向对象方法开发该系统，得到如图3-1所示的用例图和3-2所示的初始类图。





【问题1】（6分）

根据说明中的描述，给出图3-1中A1〜A3所对应的参与者名称和U1〜U3所对应的用例名称。

【问题2】（5分）

根据说明中的描述，给出图3-2中C1〜C5所对应的类名。

【问题3】（4分）

根据说明中的描述，给出图3-2中类C4、C5、Patient和DentalStaff的必要属性。

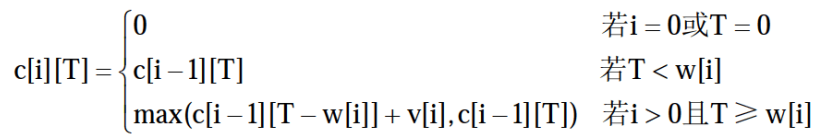
**试题四（共15分）**

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】

0-1背包问题定义为：给定i个物品的价值v[1…i]、小重量w[1…i]和背包容量T，每个物品装到背包里或者不装到背包里。求最优的装包方案，使得所得到的价值最大。

0-1背包问题具有最优子结构性质。定义c[i][T]为最优装包方案所获得的最大价值，则可得到如下所示的递归式。



【C代码】

下面是算法的C语言实现。

（1）常量和变量说明

T：背包容量

v[]：价值数组

w[]：重量数组

c[][]：c[i][j]表示前i个物品在背包容量为j的情况下最优装包方案所能获得的最大价值

（2）C程序

#include <stdio.h>

#include <math.h>

#define N 6

#define maxT 1000

int c[N][maxT] = {0};

int Memoized\_Knapsack(int v[N], int w[N], int T) {

int i;

int j;

for (i = 0; i < N; i ++ ) {

for (j = 0; j <= T; j ++ ) {

c[i][j] = -1;

}

}

return Calculate\_Max\_Value(v, w, N - 1, T);

}

int Calculate\_Max\_Value(int v[N], int w[N], int i, int j) {

int temp = 0;

if (c[i][j] != -1) {

return (1) ;

}

if (i == 0 || j == 0) {

c[i][j] == 0;

} else {

c[i][j] = Calculate\_Max\_Value(v, w, i - 1, j);

if ( (2) ) {

temp = (3) ;

if (c[i][j] < temp) {

(4) ;

}

}

}

return c[i][j];

}

【问题1】（8分）

根据说明和C代码，填充C代码中的空（1）〜（4）。

【问题2】（4分）

根据说明和C代码，算法采用了 （5） 设计策略。在求解过程中，采用了 （6）（自底向上或者自顶向下）的方式。

【问题3】（3分）

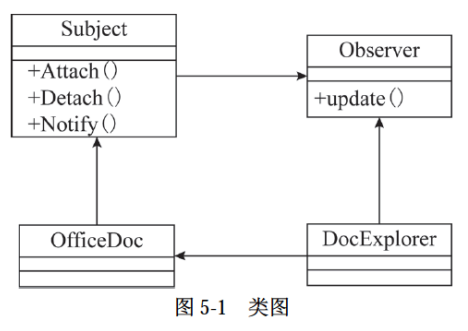
若5项物品的价值数组和重量数组分别为v[] = {0, 1, 6, 18, 22, 28}和w[] = {0, 1, 2, 5, 6, 7}背包容量为T = 11，则获得的最大价值为 （7） 。

**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】

某文件管理系统中定义了类OfficeDoc和DocExplorer。当类OfficeDoc发生变化时，类DocExplorer的所有对象都要更新其自身的状态。现采用观察者（Observer）设计模式来实现该需求，所设计的类图如图5-1所示。



【C++代码】

#include <iostream>

#include <vector>

#include <string>

using namespace std;

class Observer {

public :

(1) ;

};

class Subject {

protected :

vector< (2) > myObs;

public :

virtual void Attach(Observer \*obs) { myObs.push\_back(obs); }

virtual void Detach(Observer \*obs) {

for (vector<Observer\*>::iterator iter = myObs.begin();

iter != myObs.end(); iter ++ ) {

if (\*iter == obs) { myObs.erase(iter); return; }

}

}

virtual void Notify() {

for (vector<Observer\*>::iterator iter = myObs.begin();

iter != myObs.end(); iter ++ ) {

(3) ;

}

}

virtual int getStatus() = 0;

virtual void setStatus(int status) = 0;

};

class OfficeDoc : public Subject {

private :

string mySubjectName;

int m\_status;

public :

OfficeDoc(string name) : mySubjectName(name), m\_status(0) {}

void setStatus(int status) { m\_status = status; }

int getStatus() { return m\_status; }

};

class DocExplorer : public Observer {

private :

string myObsName;

public :

DocExplorer(string name, (4) sub) : myObsName(name) {

sub-> (5) ;

}

void update() { cout << "update observer:" << myObsName << endl; }

};

int main() {

Subject \*subjectA = new OfficeDoc("subject A");

Observer \*observerA = new DocExplorer("observerA", subjectA);

subjectA->setStatus(1); subjectA->Notify();

return 0;

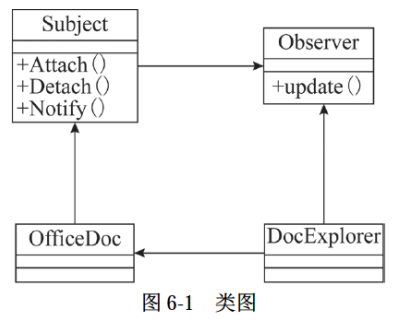
}

**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】

某文件管理系统中定义了类OfficeDoc和DocExplorer。当类OfficeDoc发生变化时，类DocExplorer的所有对象都要更新其自身的状态。现采用观察者（Observer）设计模式来实现该需求，所设计的类图如图6-1所示。



【Java代码】

import java.util.\*;

interface Observer {

public (1) ;

}

interface Subject {

public void Attach(Observer obs);

public void Detach(Observer obs);

public void Notify();

public void setStatus(int status);

public int getStatus();

}

class OfficeDoc implements Subject {

private List< (2) > myObs;

private String mySubjectName;

private int m\_status;

public OfficeDoc(String name) {

mySubjectName = name;

this.myObs = new ArrayList<Observer>();

m\_status = 0;

}

public void Attach(Observer obs) { this.myObs.add(obs); }

public void Detach(Observer obs) { this.myObs.remove(obs); }

public void Notify() {

for (Observer obs : this.myObs) { (3) ; }

}

public void setStatus(int status) {

m\_status = status;

System.out.println("SetStatus subject[" + mySubjectName + "]status:" + status);

}

public int getStatus() { return m\_status; }

}

class DocExplorer implements Observer {

private String myObsName;

public DocExplorer(String name, (4) sub) {

myObsName = name;

sub. (5) ;

}

public void update() {

System.out.println("update observer[" + myObsName + "]");

}

}

class ObserverTest {

public static void main(String[] args) {

Subject subjectA = new OfficeDoc("subject A");

Observer observerA = new DocExplorer("observer A", subjectA);

subjectA.setStatus(1);

subjectA.Notify();

}

}