全国计算机技术与软件专业技术资格（水平）考试

2011 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 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.仿照下面例题，将解答写在答题纸的对应栏内。

例题

2011 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 4 日”，故在答题纸的对应栏内写上“11”和“4”（参看下表）。

例题	解答栏
（1）	11
（2）	4

试题一至试题四是必答题

试题一

某公司欲开发招聘系统以提高招聘效率，其主要功能如下：

(1)接受申请

验证应聘者所提供的自身信息是否完整，是否说明了应聘职位，受理验证合格的申请，给应聘者发送致谢信息。

(2)评估应聘者

根据部门经理设置的职位要求，审查已经受理的申请；对未被录用的应聘者进行谢绝处理，将未被录用的应聘者信息存入未录用的应聘者表，并给其发送谢绝决策；对录用的应聘者进行职位安排评价，将评价结果存入评价结果表，并给其发送录用决策，发送录用职位和录用者信息给工资系统。

现采用结构化方法对招聘系统进行分析与设计，获得如图 1-1 所示的顶层数据流图、图 1-2 所示 0 层数据流图和图 1-3 所示 1 层数据流图。

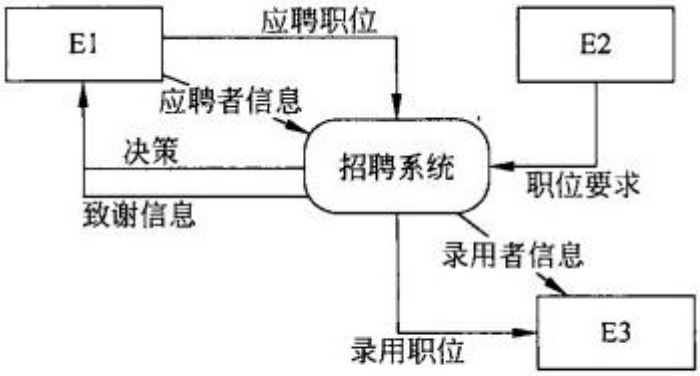

图 1-1 顶层数据流图

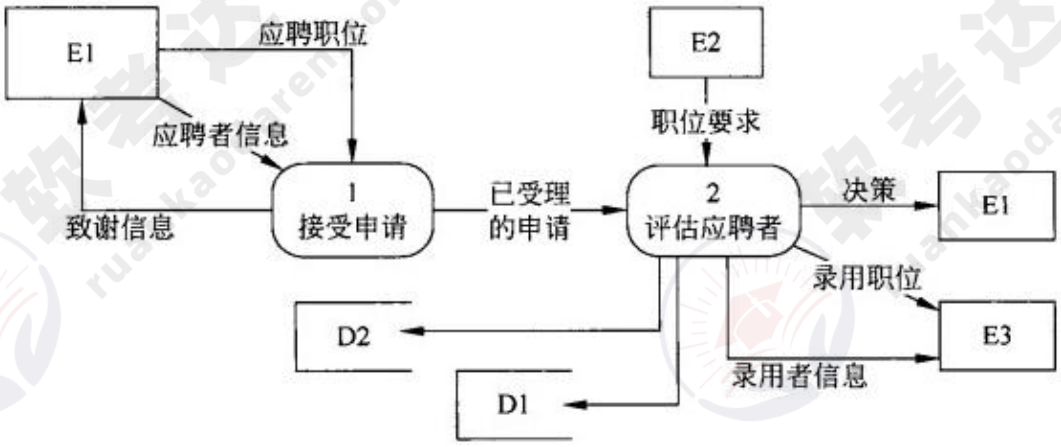

图 1-2 0 层数据流图

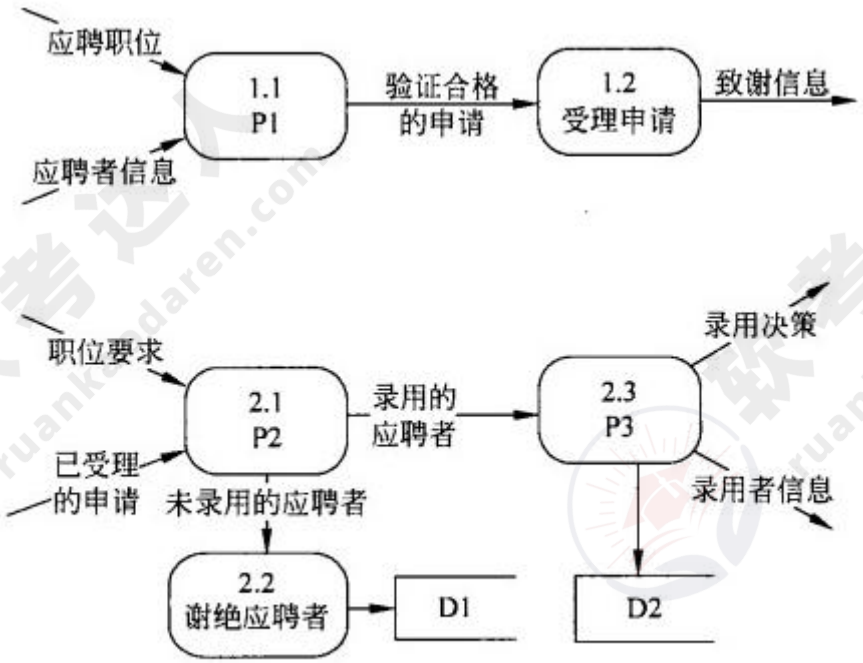

图 1-3 1 层数据流图

【问题 1】

使用说明中的术语，给出图中 E1～E3 所对应的实体名称。

【问题 2】

使用说明中的术语，给出图中 D1～D2 所对应的数据存储名称。

【问题 3】

使用说明和图中的术语，给出图 1-3 中加工 P1～P3 的名称。

【问题 4】

解释说明图 1-2 和图 1-3 是否保持平衡，若不平衡请按如下格式补充图 1-3 中数据流的名称以及数据流的起点或终点，使其平衡（使用说明中的术语或图中符号）。

数据流名称	起点	终点

试题二

某物流公司为了整合上游供应商与下游客户，缩短物流过程，降低产品库存，需要构建一个信息系统以方便管理其业务运作活动。

【需求分析结果】

（1）物流公司包含若干部门，部门信息包括部门号、部门名称、经理、电话和邮箱。一个部门可以有多名员工处理部门的日常事务，每名员工只能在一个部门工作。每个部门有一名经理，只需负责管理本部门的事务和人员。

（2）员工信息包括员工号、姓名、职位、电话号码和工资；其中，职位包括：经理、业务员等。业务员根据托运申请负责安排承运货物事宜，例如：装货时间、到达时间等。一个业务员可以安排多个托运申请，但一个托运申请只由一个业务员处理。

（3）客户信息包括客户号、单位名称、通信地址、所属省份、联系人、联系电话、银行账号，其中，客户号唯一标识客户信息的每一个元组。每当客户要进行货物托运时，先要提出货物托运申请。托运申请信息包括申请号、客户号、货物名称、数量、运费、出发地、目的地。其中，一个申请号对应唯一的一个托运申请；一个客户可以有多个货物托运申请，但一个托运申请对应唯一的一个客户号。

【概念模型设计】

根据需求阶段收集的信息，设计的实体联系图和关系模式（不完整）如图 2-1 所示。

图 2-1 实体联系图

【关系模式设计】

部门（部门号，部门名称，经理，电话，邮箱）

员工（员工号，姓名，职位，电话号码，工资，（A））

客户（（B）单位名称，通信地址，所属省份，联系人，联系电话，银行账号）

托运申请（（C），货物名称，数量，运费，出发地，目的地）

安排承运（（D），装货时间，到达时间，业务员）

【问题 1】

根据问题描述，补充四个联系、联系的类型，以及实体与子实体的联系，完善图 2-1 所示的实体联系图。

【问题 2】

根据实体联系图，将关系模式中的空（A)～（D)补充完整。分别指出部门、员工和安排承运关系模式的主键和外键。

【问题 3】

若系统新增需求描述如下：

为了数据库信息的安全性，公司要求对数据库操作设置权限管理功能，当员工登录系统时，系统需要检查员工的权限。权限的设置人是部门经理。为满足上述需要，应如何修改（或补充）图 2-1 所示的实体联系图，请给出修改后的实体联系图和关系模式。

试题三

PAY&DRIVE 系统（开多少付多少）能够根据驾驶里程自动计算应付的费用。

系统中存储了特定区域的道路交通网的信息。道路交通网由若干个路段（ROAD SEGMENT）构成，每个路段由两个地理坐标点（NODE）标定，其里程数（DISTANCE）是已知的。在某些地理坐标点上安装了访问控制（ACCESSCONTROL）设备，可以自动扫描行驶卡（CARD）。行程（TRAJECTORY)由一组连续的路段构成。行程的起点（ENTRY）和终点（EXIT）都装有访问控制设备。

系统提供了 3 种行驶卡。常规卡（REGULAR CARD）有效期（VALID PERIOD）为一年，可以在整个道路交通网内使用。季卡（SEASONCARD)有效期为三个月，可以在整个道路交通网内使用。单次卡（MINITRIP CARD)在指定的行程内使用，且只能使用一次。其中，季卡和单次卡都是预付卡（PREPAIDCARD)，需要客户（CUSTOMER）预存一定的费用。

系统的主要功能有：客户注册、申请行驶卡、使用行驶卡行驶等。

使用常规卡行驶，在进入行程起点时，系统记录行程起点、进入时间（DATE OF ENTRY）等信息。在到达行程终点时，系统根据行驶的里程数和所持卡的里程单价（UNIT PRICE）计算应付费用，并打印费用单（INVOICE）。

季卡的使用流程与常规卡类似，但是不需要打印费用单，系统自动从卡中扣除应付费用。单次卡的使用流程与季卡类似，但还需要在行程的起点和终点上检查行驶路线是否符合该卡所规定的行驶路线。

现采用面向对象方法开发该系统，使用 UML 进行建模。构建出的用例图和类图分别如图 3-1 和图 3-2 所示。

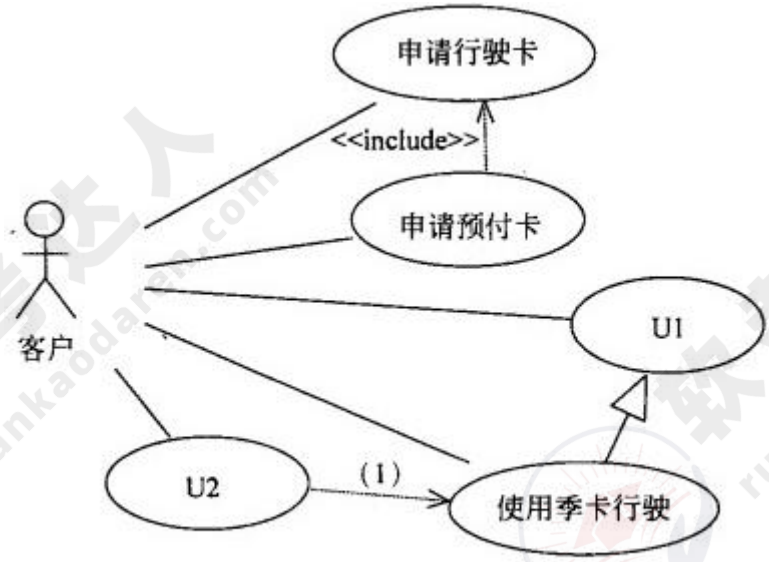

图 3-1 用例图

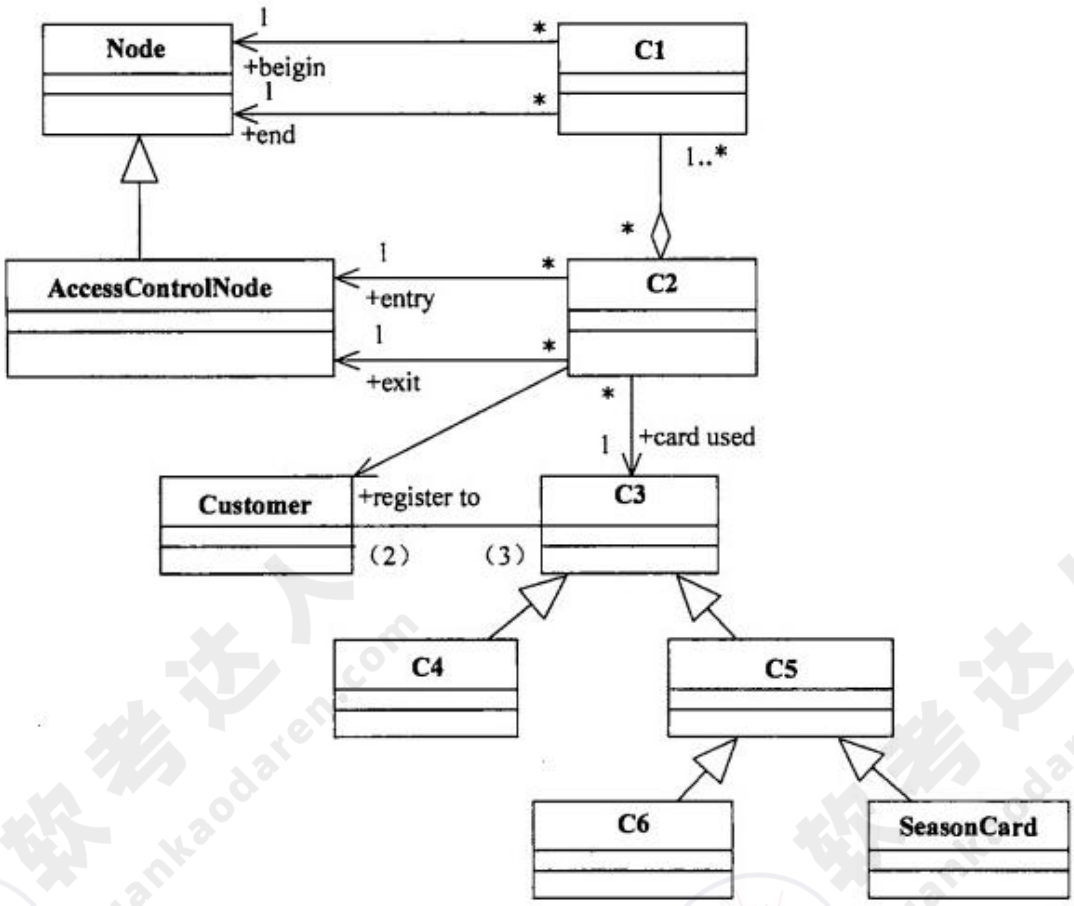

图 3-2 类图

【问题 1】

根据说明中的描述，给出图 3-1 中 U1 和 U2 所对应的用例，以及（1）所对应的关系。

【问题 2】

根据说明中的描述，给出图 3-2 中缺少的 C1～C6 所对应的类名以及（2）～（3） 处所对应的多重度（类名使用说明中给出的英文词汇）。

【问题 3】

根据说明中的描述，给出 ROAD SEGMENT、TRAJECTORY 和 CARD 所对应的类的关键属性（属性名使用说明中给出的英文词汇）。

试题四

设某一机器由 N 个部件组成，每一个部件都可以从 M 个不同的供应商处购得。供应商 J 供应的部件 I 具有重量 WIJ 和价格 CIJ。设计一个算法，求解总价格不超过上限 CC 的最小重量的机器组成。

采用回溯法来求解该问题：

首先定义解空间。解空间由长度为 N 的向量组成，其中每个分量取值来自集合{1,2，…，M} 将解空间用树形结构表示。

接着从根结点开始，以深度优先的方式搜索整个解空间。从根结点开始，根结点成为活结点，同时也成为当前的扩展结点。向纵深方向考虑第一个部件从第一个供应商处购买，得到一个新结点。判断当前的机器价格(C11)是否超过上限(CC)，重量(W11) 是否比当前已知的解（最小重量）大，若是，应回溯至最近的一个活结点；若否，则该新结点成为活结点，同时也成为当前的扩展结点，根结点不再是扩展结点。继续向纵深方向考虑第二个部件从第一个供应商处购买，得到一个新结点。同样判断当前的机器价格(C11+C21)是否超过上限(CC)，重量(W11+W21)是否比当前已知的解(最小重量)大。若是，应回溯至最近的一个活结点；若否，则该新结点成为活结点，同时也成为当前的扩展结点，原来的结点不再是扩展结点。以这种方式递归地在解空间中搜索，直到找到所要求的解或者解空间中已无活结点为止。

【C 代码】

下面是该算法的 C 语言实现。

（1）变量说明

n：机器的部件数

m：供应商数

cc：价格上限

w[][]：二维数组，w[i][j]表示第 j 个供应商供应的第 i 个部件的重量

c[][]：二维数组，c[i][j]表示第 j 个供应商供应的第 i 个部件的价格

bestW：满足价格上限约束条件的最小机器重量

bestC：最小重量机器的价格

bestX[]：最优解，一维数组，bestX[i]表示第 i 个部件来自哪个供应商

cw：搜索过程中机器的重量

cp：搜索过程中机器的价格

x[]：搜索过程中产生的解，x[i]表示第 i 个部件来自哪个供应商

i：当前考虑的部件，从 0 到 n–1

j：循环变量

（2）函数 backtrack

```
int n = 3;
int m = 3;
int cc = 4;
int w[3][3] = {{1,2,3},{3,2,1},{2,2,2}};
int c[3][3] = {{1,2,3},{3,2,1},{2,2,2}};
int bestW = 8;
int bestC = 0;
int bestX[3] = {0,0,0};
int cw = 0;
int cp = 0;
int x[3] = {0,0,0};
int backtrack(int i){
```

```
    int j = 0;
    int found = 0;
    if(i > n - 1){  /*得到问题解*/
        bestW = cw;
        bestC = cp;
        for(j = 0; j < n; j++){
            ____(1)____;
        }
return 1;
    }
    if(cp <= cc){ /*有解*/
        found = 1;
    }
    for(j = 0; ___(2)___; j++){
        /*第 i 个部件从第 j 个供应商购买*/
____(3)____;
        cw = cw + w[i][j];
        cp = cp + c[i][j];
if(cp <= cc &&___(4)___){ /*深度搜索，扩展当前结点*/
            if(backtrack(i + 1)){   found = 1;  }
        }
        /*回溯*/
        cw = cw - w[i][j];
        ___(5)___ ;
    }
    return found;
}
```

从下列的 2 道试题（试题五至试题六）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

某大型商场内安装了多个简易的纸巾售卖机，自动出售 2 元钱一包的纸巾，且每次仅售出一包纸巾。纸巾售卖机的状态图如图 5-1 所示。

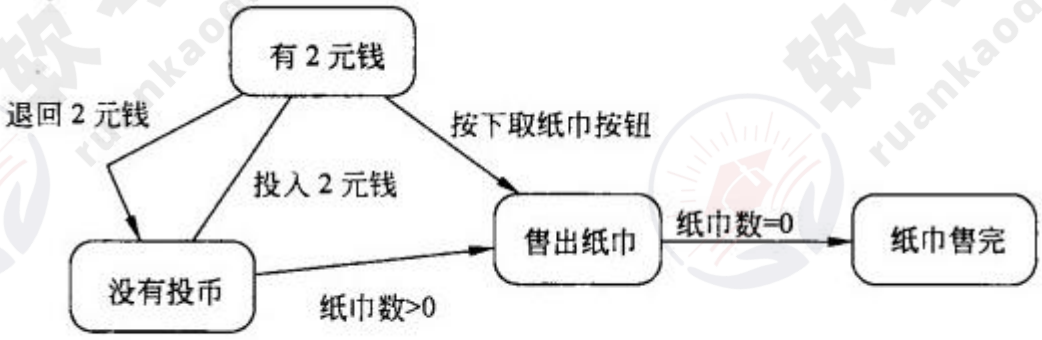

图 5-1 纸巾售卖机状态图

采用状态（STATE)模式来实现该纸巾售卖机，得到如图 5-2 所示的类图。其中类 STATE 为抽象类，定义了投币、退币、出纸巾等方法接口。类 SOLDSTATE、SOLDOUTSTATE> NOQUARTERSTATE 和 HASQUARTERSTATE 分别对应图 5-1 中纸巾售卖机的 4 种状态：售出纸巾、纸巾售完、没有投币、有 2 元钱。

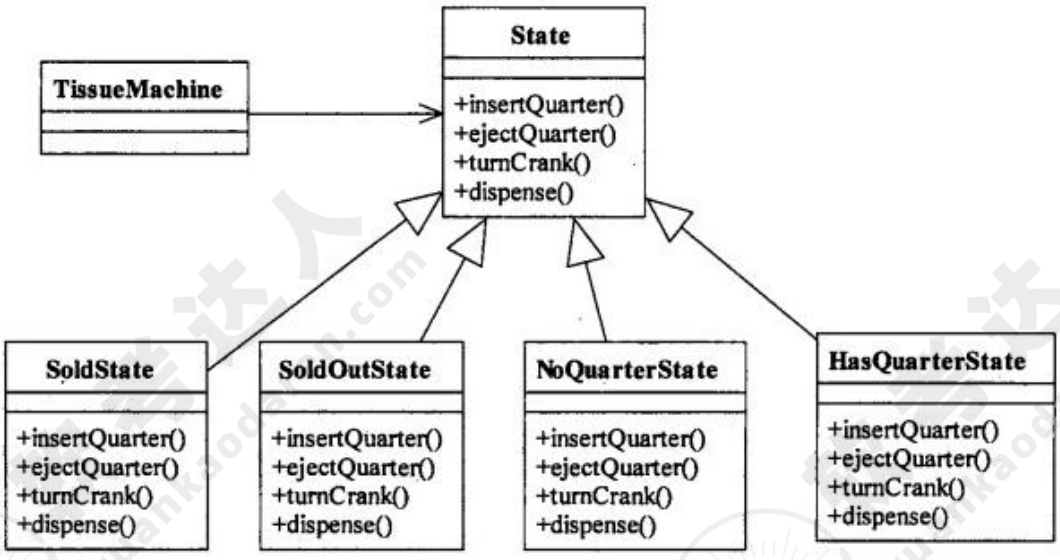

图 5-2 类图

【问题 1】

【C++代码】

```
#include <iostream>
using namespace std;
// 以下为类的定义部分
class TissueMachine;                    // 类的提前引用

class State {
public:
  virtual void insertQuarter() = 0;//投币
  virtual void ejectQuarter() = 0; //退币
  virtual void turnCrank()= 0;     //按下“出纸巾”按钮
  virtual void dispense() = 0;     //出纸巾
};
/* 类 SoldOutState、NoQuarterState、HasQuarterState、SoldState 的定义省略，
每个类中均定义了私有数据成员 TissueMachine* tissueMachine; */
class TissueMachine {
private:
   (1)  *soldOutState, *noQuarterState, *hasQuarterState,*soldState,
*state ;
   int count;                           //纸巾数
public:
  TissueMachine(int numbers);
  void setState(State* state);
  State* getHasQuarterState();
  State* getNoQuarterState();
  State* getSoldState();
  State* getSoldOutState();
  int getCount();
  //其余代码省略
};
```

```
//以下为类的实现部分
void NoQuarterState ::insertQuarter() {
tissueMachine->setState(   (2)   );
}
void HasQuarterState ::ejectQuarter() {
tissueMachine->setState(   (3)   );
}
void SoldState ::dispense() {
   if(tissueMachine->getCount() > 0) {
      tissueMachine->setState(   (4)   );
}
else {
      tissueMachine->setState(   (5)   );
}
}  //其余代码省略
```

试题六

某大型商场内安装了多个简易的纸巾售卖机，自动出售 2 元钱一包的纸巾，且每次仅售出一包纸巾。纸巾售卖机的状态图如图 6-1 所示。

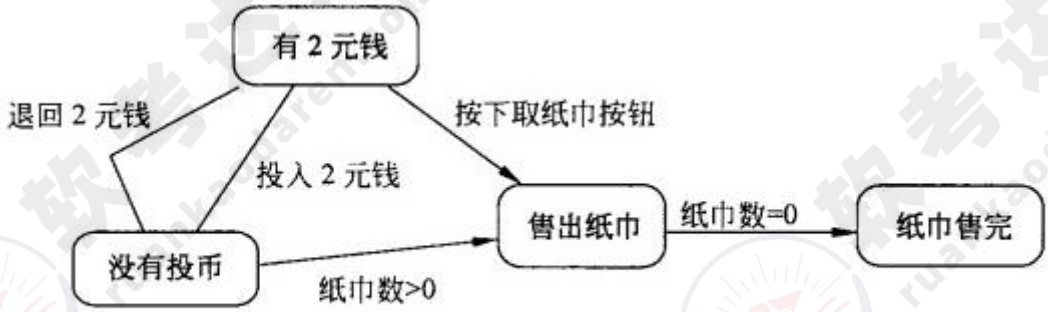

图 6-1　纸巾售卖机状态图

采用状态（State)模式来实现该纸巾售卖机，得到如图 6-2 所示的类图。其中类 State 为抽象类，定义了投币、退币、出纸巾等方法接口。类 SoldState、SoldOutState、NoQuarterState 和 HasQuarterState 分别对应图 6-1 中纸巾售卖机的 4 种状态：售出纸巾、纸巾售完、没有投币、有 2 元钱。

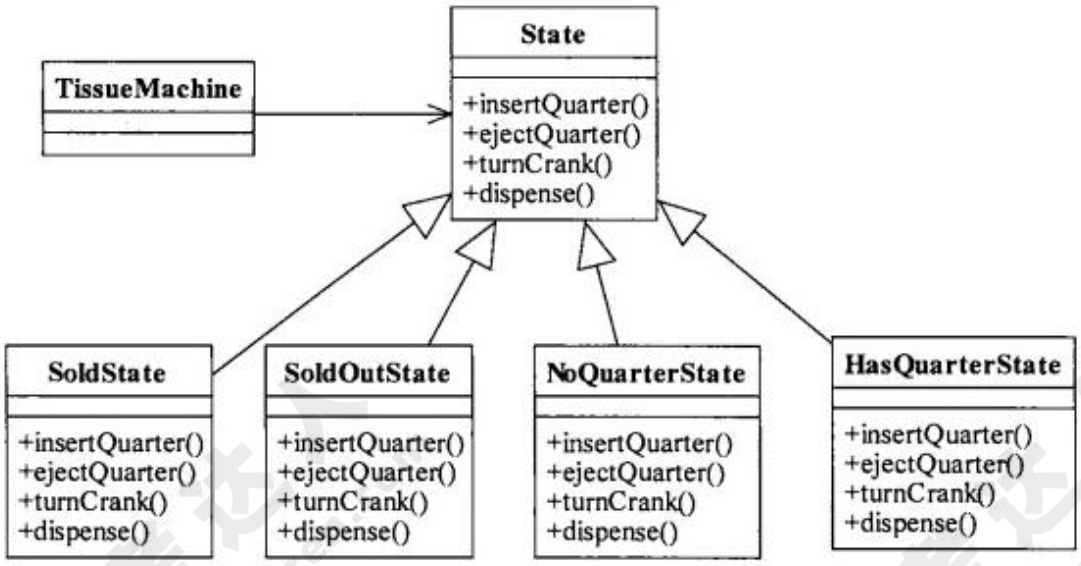

图 6-2　类图

【问题 1】

【Java 代码】

```
import java.util.*;
interface State {
  public void insertQuarter();          //投币
  public void ejectQuarter();           //退币
  public void turnCrank();              //按下“出纸巾”按钮
  public void dispense();               //出纸巾
}
class TissueMachine {
   (1)  soldOutState, noQuarterState, hasQuarterState, soldState,
state;
  state = soldOutState;
  int count = 0;                        //纸巾数
  public TissueMachine(int numbers) { /* 实现代码省略 */ }
 public State getHasQuarterState()  { return hasQuarterState;  }
  public State getNoQuarterState()   { return noQuarterState;  }
  public State getSoldState()       { return soldState;      }
  public State getSoldOutState()     { return soldOutState;    }
  public int getCount()            { return count;         }
 //其余代码省略
 }

 class NoQuarterState implements State {
  TissueMachine tissueMachine;
  public void insertQuarter() {
    tissueMachine.setState(  (2)  );
  }
//构造方法以及其余代码省略
}
class HasQuarterState implements State {
  TissueMachine tissueMachine;
  public void ejectQuarter() {
    tissueMachine.setState(  (3)  );
  }
//构造方法以及其余代码省略
}
class SoldState implements State {
  TissueMachine tissueMachine;
  public void dispense() {
    if(tissueMachine.getCount() > 0)  {
```

```
        tissueMachine.setState(__(4)__);
    } else {
        tissueMachine.setState(__(5)__); }
    }
}
```