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 2016 年 下半年 下午试卷 案例

（考试时间 150 分钟）

试题一 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】

某证券交易所为了方便提供证券交易服务，欲开发一证券交易平台，该平台的主要功能如下：

(1) 开户。根据客户服务助理提交的开户信息，进行开户，并将客户信息存入客户记录中，账户信息(余额等)存入账户记录中；

(2) 存款。客户可以向其账户中存款，根据存款金额修改账户余额；

(3) 取款。客户可以从其账户中取款，根据取款金额修改账户余额；

(4) 证券交易。客户和经纪人均可以进行证券交易(客户通过在线方式，经纪人通过电话)，将交易信息存入交易记录中；

(5) 检查交易。平台从交易记录中读取交易信息，将交易明细返回给客户。

现采用结构化方法对该证券交易平台进行分析与设计，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。

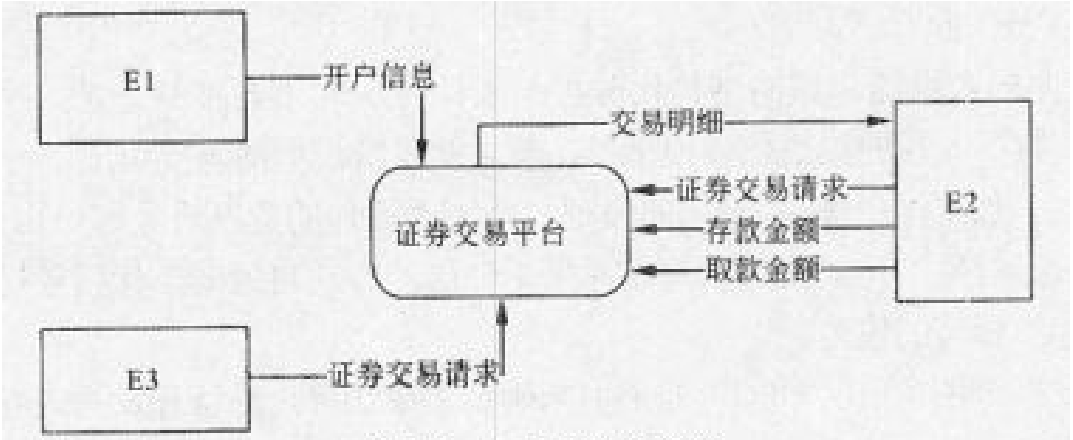


图 1-1　上下文数据流图

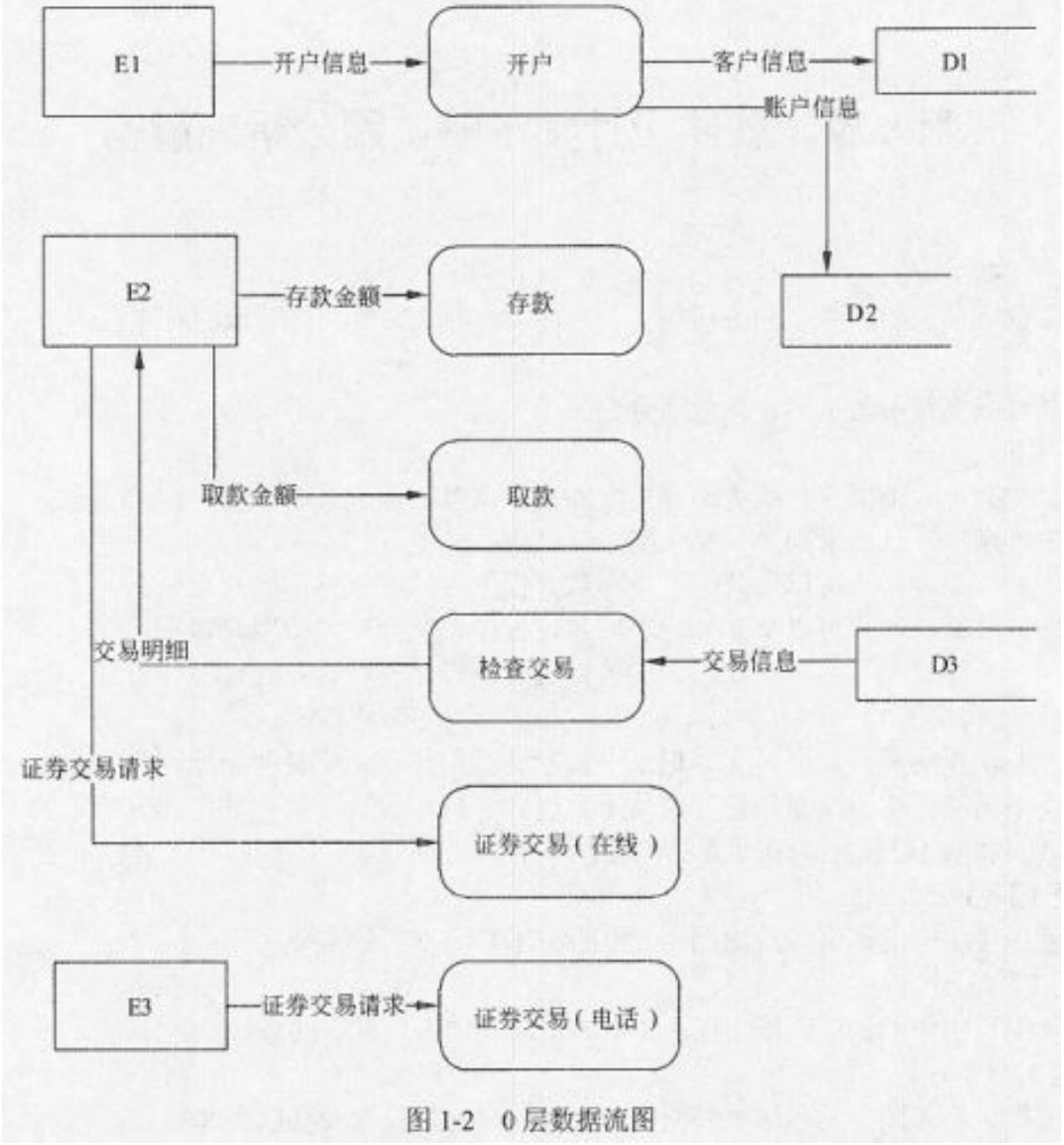


图 1-2　0 层数据流图

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
根据说明和图中的术语，补充图 1-2 中缺失的数据流及其起点和终点。
问题：1.4
实际的证券交易通常是在证券交易中心完成的，因此，该平台的“证券交易”功能需将交易信息传递给证券交易中心。针对这个功能需求，需要对图 1-1 和图 1-2 进行哪些修改，请用 200 字以内的文字加以说明。

试题二 【说明】

某宾馆为了有效地管理客房资源，满足不同客户需求，拟构建一套宾馆信息管理系统，以方便宾馆管理及客房预订等业务活动。

【需求分析结果】

该系统的部分功能及初步需求分析的结果如下：

(1）宾馆有多个部门，部门信息包括部门号、部门名称、电话、经理。每个部门可以有多名员工，每名员工只属于一个部门；每个部门只有一名经理，负责管理本部门。

(2）员工信息包括员工号、姓名、岗位、电话、工资，其中，员工号唯一标识员工关系中的一个元组，岗位有经理、业务员。

(3）客房信息包括客房号(如 1301 、 1302 等)、客房类型、收费标准、入住状态(已入住 / 未入住)，其中客房号唯一标识客房关系中的一个元组，不同客房类型具有不同的收费标准。

(4）客户信息包括客户号、单位名称、联系人、联系电话、联系地址，其中客户号唯一标识客户关系中的一个元组。

(5）客户预订客房时，需要填写预订申请。预订申请信息包括申请号、客户号、入住时间、入住天数、客房类型、客房数量，其中，一个申请号唯一标识预订申请中的一个元组；一位客户可以有多个预订申请，但一个预订申请对应唯一的一位客户。

(6）当客户入住时，业务员根据客户的预订申请负责安排入住客房事宜。安排信息包括客房号、姓名、性别、身份证号、入住时间、天数、电话，其中客房号、身份证号和入住时间唯一标识一次安排。一名业务员可以安排多个预订申请，一个预订申请只由一名业务员安排，而且可安排多间同类型的客房。

【概念模型设计】

根据需求阶段收集的信息，设计的实体联系图如图 2-1 所示。

【关系模式设计】

部门(部门号，部门名称，经理，电话)

员工(员工号，(a)，姓名，岗位，电话，工资)

客户((b)，联系人，联系电话，联系地址)

客房(客房号，客房类型，收费标准，入住状态)

预订申请((c)，入住时间，天数，客房类型，客房数量)

安排(申请号，客房号，姓名，性别，(d)，天数，电话，业务员)

图 2-1 实体联系图

问题： 2.1

根据问题描述，补充四个联系，完善图 2-1 ，的实体联系图。联系名可用联系 1 、联系 2 、联系 3 和联系 4 代替，联系的类型为 1:1 、 1:n 和 m:n(或 1:1 ，和 1:*和*:*)。

问题：2.2
(1)根据题意，将关系模式中的空(a)～(d)补充完整，并填入答题纸对应的位置上。
(2)给出“预订申请”和“安排”关系模式的主键和外键。
问题：2.3
【关系模式设计】中的“客房”关系模式是否存在规范性问题，请用100字以内文字解释你的观点(若存在问题，应说明如何修改“客房”关系模式)。

试题三 【说明】

某种出售罐装饮料的自动售货机(VendingMachine)的工作过程描述如下：

(1) 顾客选择所需购买的饮料及数量。

(2) 顾客从投币口向自动售货机中投入硬币(该自动售货机只接收硬币)。硬币器收集投入的硬币并计算其对应的价值。如果所投入的硬币足够购买所需数量的这种饮料且饮料数量足够，则推出饮料，计算找零，顾客取走饮料和找回的硬币；如果投入的硬币不够或者所选购的饮料数量不足，则提示用户继续投入硬币或重新选择饮料及数量。

(3) 一次购买结束之后，将硬币器中的硬币移走(清空硬币器)，等待下一次交易。自动售货机还设有一个退币按钮，用于退还顾客所投入的硬币。已经成功购买饮料的钱是不会被退回的。

采用面向对象方法分析和设计该自动售货机的软件系统，得到如图3-1所示的用例图，其中，用例“购买饮料”的用例规约描述如下。

参与者：顾客。

主要事件流：

1．顾客选择需要购买的饮料和数量，投入硬币；

2．自动售货机检查顾客是否投入足够的硬币；

3．自动售货机检查饮料储存仓中所选购的饮料是否足够；

4．自动售货机推出饮料；

5．自动售货机返回找零。

各选事件流：

2a．若投入的硬币不足，则给出提示并退回到 1；

3a．若所选购的饮料数量不足，则给出提示并退回到 1。

根据用例“购买饮料”得到自动售货机的 4 个状态：“空闲”状态、“准备服务”状态、“可购买”状态以及“饮料出售”状态，对应的状态图如图 3-2 所示。

所设计的类图如图 3-3 所示。

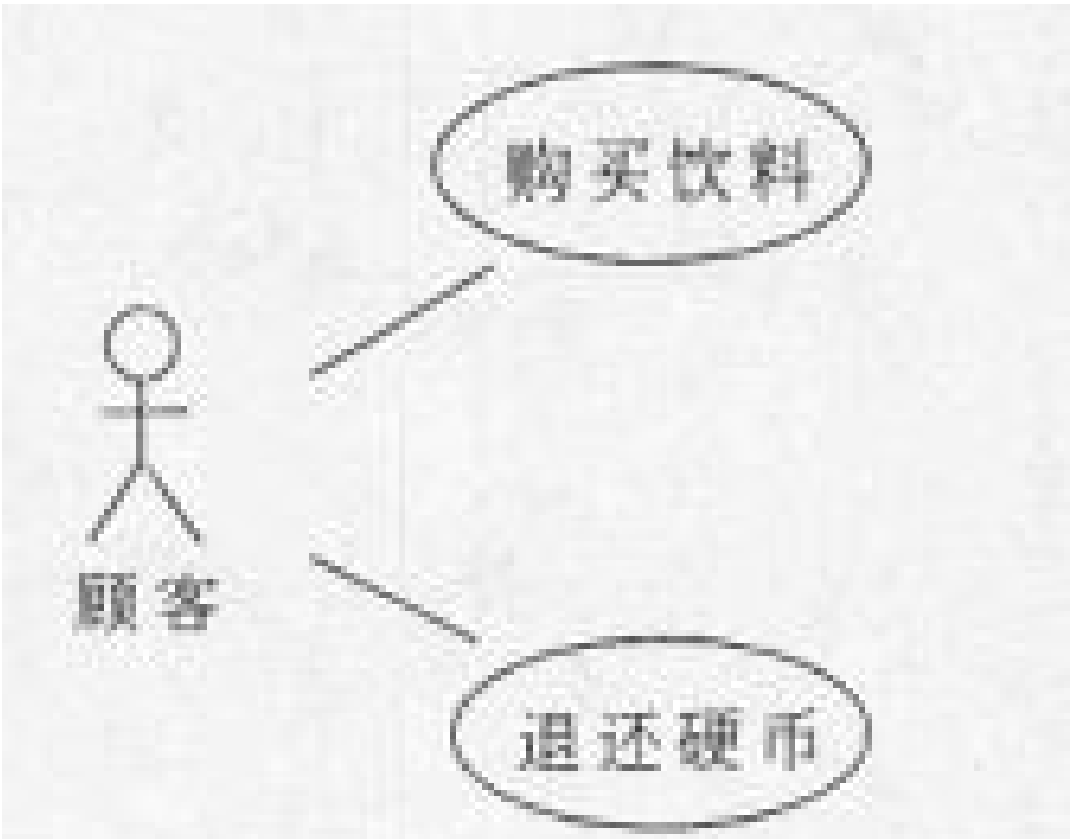


图 3-1　用例图

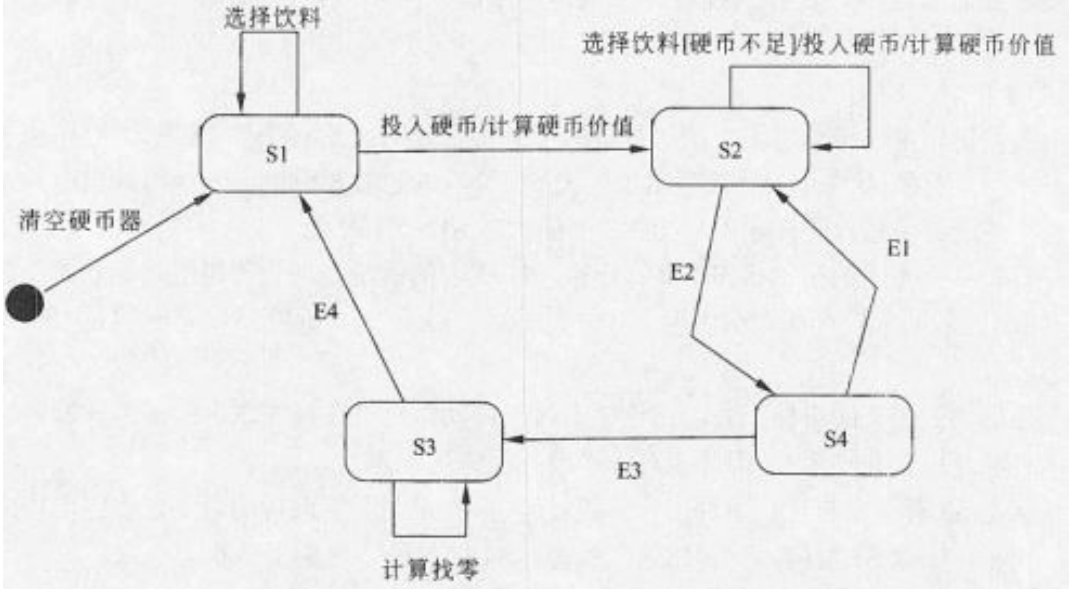


图 3-2　状态图

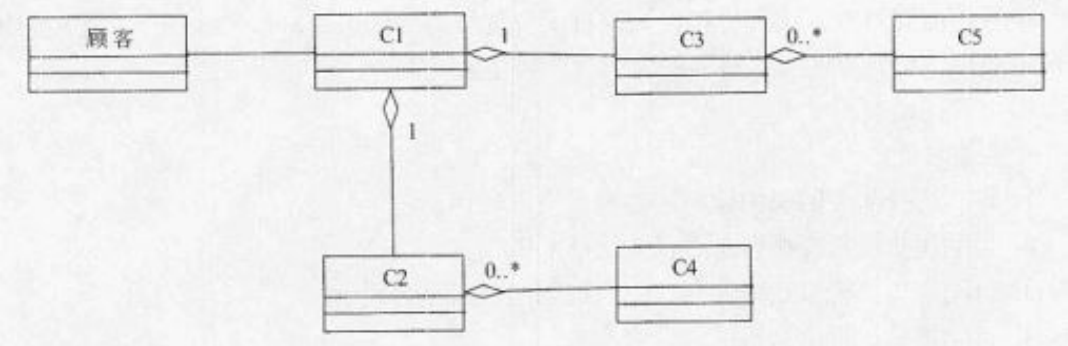


图 3-3　类图

问题：3.1
根据说明中的描述，使用说明中的术语，给出图 3-2 中的 S1 ～ S4 所对应的状态名。
问题：3.2
根据说明中的描述，使用说明中的术语，给出图 3-2 中的 E1 ～ E4 所对应的事件名
问题：3.3
根据说明中的描述，使用说明中的术语，给出图 3-3 中 C1 ～ C5 所对应的类名。

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】

模式匹配是指给定主串 t 和子串 s，在主串 t 中寻找子串 s 的过程，其中 s 称为模式。如果匹配成功，返回 s 在 t 中的位置，否则返回-1 。

KMP 算法用 next 数组对匹配过程进行了优化。 KMP 算法的伪代码描述如下：

1．在串 t 和串 s 中，分别设比较的起始下标 i=j=0 。

2．如果串 t 和串 s 都还有字符，则循环执行下列操作：

(1）如果 j=-1 或者 t[i]=s[j]，则将 i 和 j 分别加 1，继续比较 t 和 s 的下一个字符；

(2）否则，将 j 向右滑动到 next[j]的位置，即 j=next[j]。

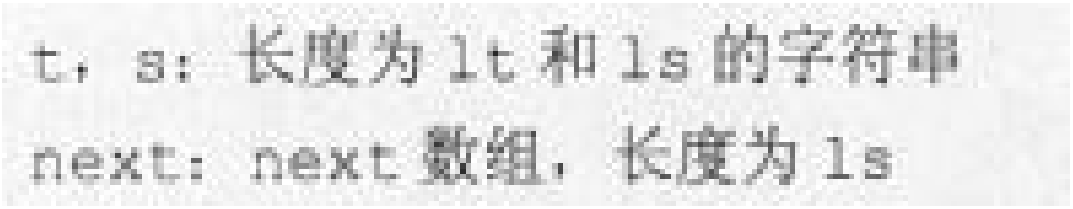
3．如果 s 中所有字符均已比较完毕，则返回匹配的起始位置(从 1 开始)；否则返回-1 .

其中， next 数组根据子串 s 求解。求解 next 数组的代码已由 get_next 函数给出。

【C 代码】

(1）常量和变量说明

(2) C 程序



```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
 /*求next[]的值*/
void get_next( int *next, char *s, int ls ) {
     int i = 0, j = -1 ;
     next[0] = -1 ; /*初始化next[0] */
     while ( i < ls ){ /*还有字符*/
        if( j == -1 || s[i] == s[j] ){/*匹配*/
             j++ ;
             i++ ;
             if( s[i] == s[j] )
                next[i] = next[j];
             else
                next[i] = j ;

          }
          else
             j = next[j] ;
     }
 }
int kmp( int *next, char *t ,char *s, int lt, int ls )
 {
     int i = 0, j = 0 ;
     while ( i < lt &&   (1)   ){
        if( j == -1 ||   (2)   ){
             i ++ ;
             j ++ ;

        } else
             (3) ;
     }
     if ( j >= ls)
        return   (4)   ;
     else
        return -1 ;
}
```

问题： 4.1
根据题干说明，填充 C 代码中的空(1)～(4).
问题： 4.2
根据题干说明和 C 代码，分析出 kmp 算法的时间复杂度为(5)(主串和子串的长度分别为 lt 和 ls ，用 O 符号表示)。
问题： 4.3
根据 C 代码，字符串“BBABBCAC”的 next 数组元素值为(6)(直接写素值，之间用逗号隔开)。若主串为“AABBCBBABBCACCD”，子串为“BBABBCAC”，则函数 Kmp 的返回值是(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】

某发票(lnvoice)由抬头(Head)部分、正文部分和脚注(Foot)部分构成。现采用装饰(Decorator)模式实现打印发票的功能，得到如图 5-1 所示的类图。

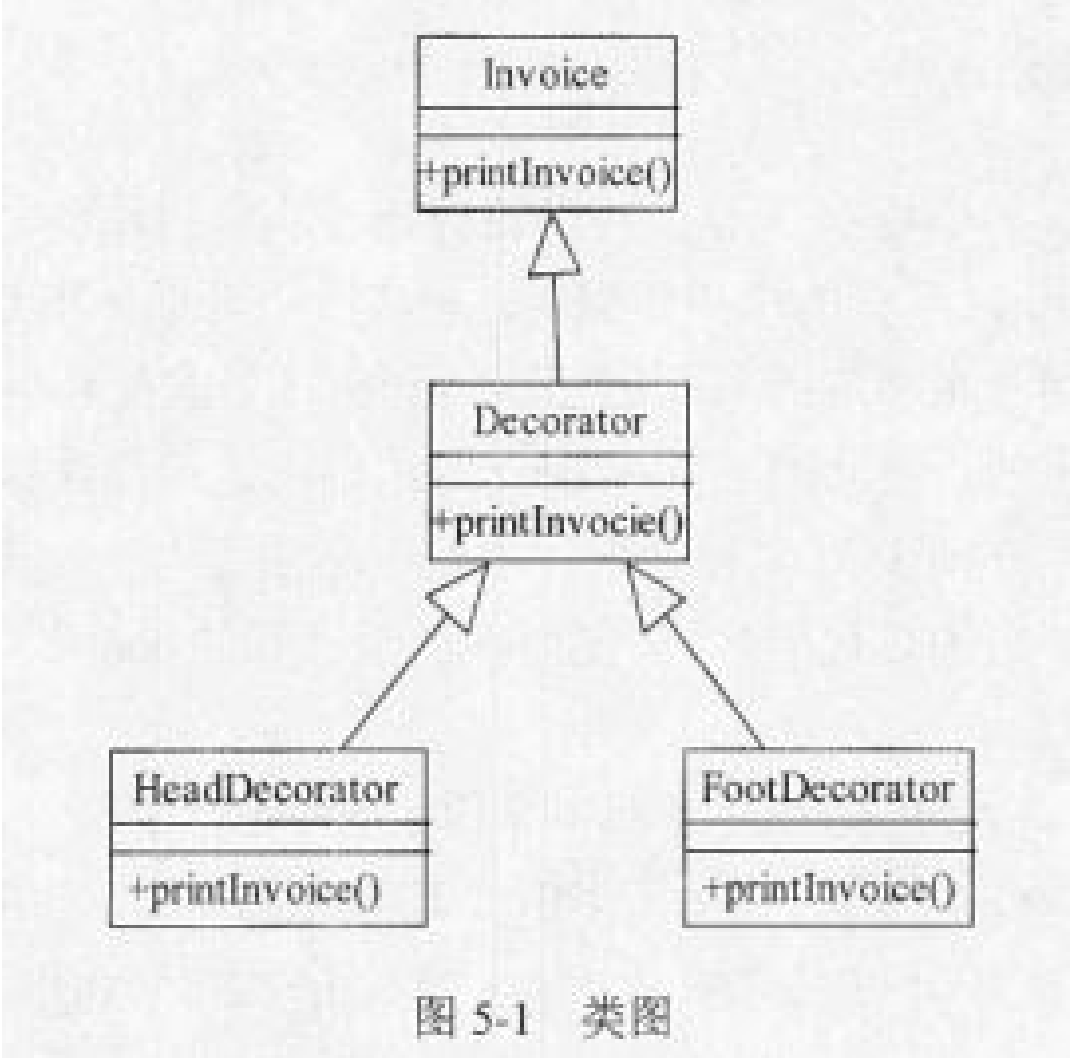


图 5-1 类图

问题： 5.1

【C++代码】

程序的输出结果为：

```
#include <iostream>
using namespace std;
class Invoice {
public:
    ___(1)___ {
        cout << "This is the content of the invoice!" << endl;
```

```
    }
};
class Decorator : public Invoice{
    Invoice *ticket;
public:
    Decorator(Invoice *t)   { ticket = t; }
    void printInvoice(){
        if(ticket != NULL)
            ____(2)____;
    }
};
class HeadDecorator : public Decorator{
public:
    HeadDecorator(Invoice *t): Decorator(t){ }
    void printInvoice() {
        cout << "This is the header of the invoice!" << endl;
        ____(3)____;
    }
};
class FootDecorator : public Decorator{
 public:
    FootDecorator(Invoice *t): Decorator(t) { }
    void printInvoice() {
        ____(4)____;
        cout << "This is the footnote of the invoice!" << endl;
    }
};
int main(void) {
    Invoice t;
    FootDecorator f(&t);
    HeadDecorator h(&f);
    h.printInvoice();
    cout << "------------------------"<< endl;
    FootDecorator a(NULL);
    HeadDecorator b( ____(5)____ );
    b.printInvoice();
    return 0;
}
```

```
This is the header of the invoice!
This is the content of the invoice!
This is the footnote of the invoice!
-------------------------
This is the header of the invoice!
This is the footnote of the invoice!
```

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】

某发票(lnvoice)由抬头(Head)部分、正文部分和脚注(Foot)部分构成。现采用装饰(Decorator)模式实现打印发票的功能，得到如图 6-1 所示的类图。

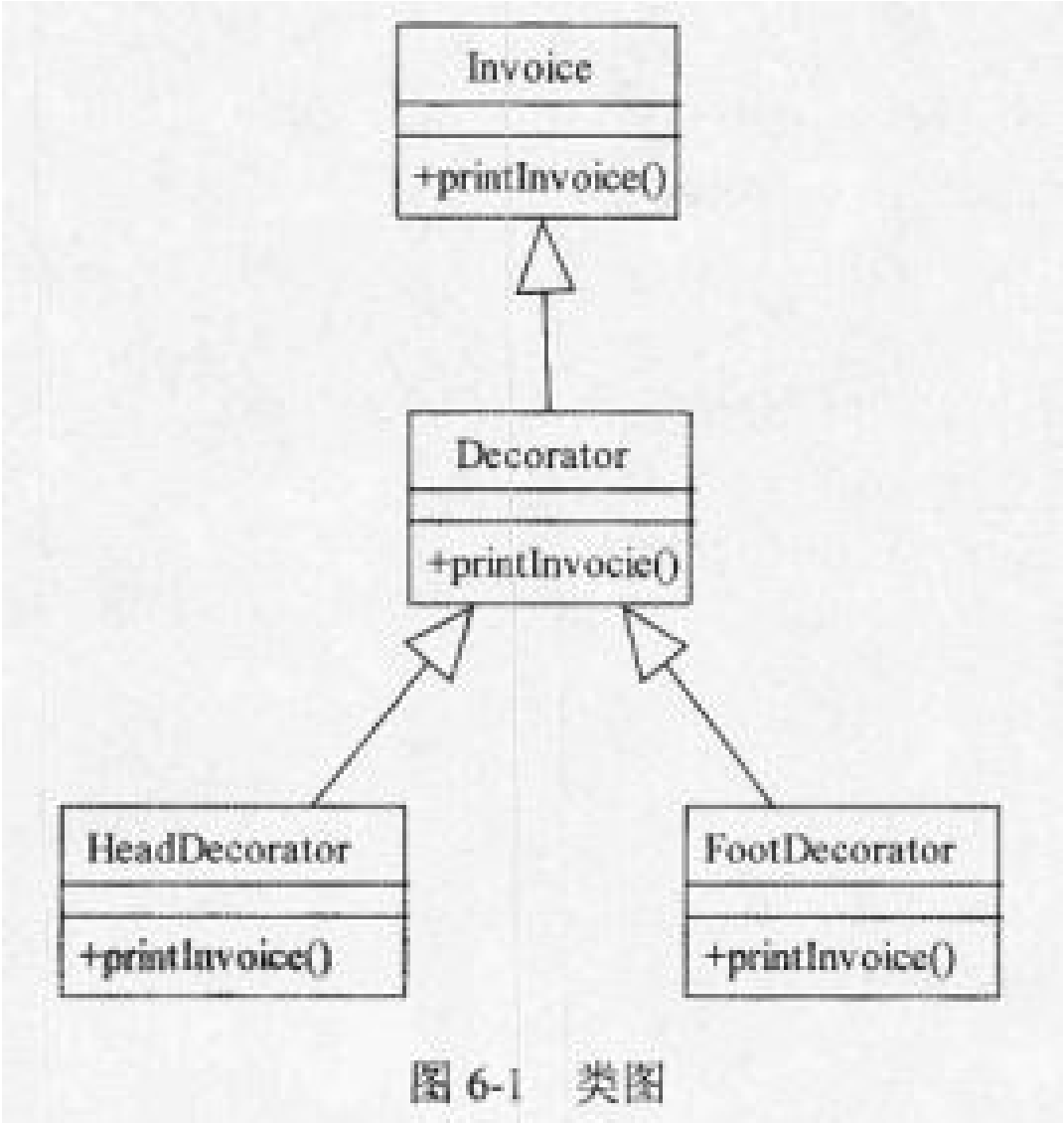


图 6-1　类图

问题： 6.1
【 java 代码】

程序的输出结果为：

```
class Invoice {
    public void printInvoice() {
        System.out.println("This is the content of the invoice!");
    }
}

class Decorator extends Invoice{
    protected Invoice ticket;
    public Decorator(Invoice t){
        ticket = t;
    }
    public void printInvoice(){
        if(ticket != null)
            ____(1)____;
    }
}
class HeadDecorator extends Decorator{
    public HeadDecorator(Invoice t) {
        super(t);
    }
    public void printInvoice() {
        System.out.println("This is the header of the invoice!");
        ____(2)____;
    }
}
class FootDecorator extends Decorator{
    public FootDecorator(Invoice t) {
        super(t);
    }
    public void printInvoice() {
        ____(3)____;
        System.out.println("This is the footnote of the invoice!");
    }
}
class Test {
    public static void main(String[] args) {
        Invoice t = new Invoice();
        Invoice ticket;
        ticket = ____(4)____;
```

```
        ticket.printInvoice();
        System.out.println("------------------------");
        ticket = ___(5)___;
        ticket.printInvoice();
    }
}
```

```
This is the header of the invoice!
This is the content of the invoice!
This is the footnote of the invoice!
------------------------
This is the header of the invoice!
This is the footnote of the invoice!
```

试题一 答案：　解析： E1：客户服务助理，E2：客户，E3：经纪人。

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，是传统的考题，考点与往年类似，要求考生细心分析题目中所描述的内容。本题题干描述较短，更易于分析。

DFD 是面向数据流建模的结构化分析与设计方法的重要工具，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。 DFD 将系统建模成输入、加工(处理)、输出的模型，即流入软件的数据对象、经由加工的转换、最后以结果数据对象的形式流出软件，并采用分层的方式自顶向下建模各层数据流图，来表示不同详细程度的模型。

上下文数据流图(顶层 DFD)通常用来确定系统边界，将待开发系统看作一个大的加工，然后根据哪些外部实体为系统提供输入数据流，以及哪些外部实体接受系统发送的数据流，建模出的上下文图中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。系统边界的变化可能使外部实体成为系统内部加工或内部加工变为外部实体。

在上下文图中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，将上下文 DFD 中的加工分解成多个加工，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，建模 0 层 DFD 。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。建模时，需要注意加工和数据流的正确使用，一个加工必须既有输入又有输出；数据流必须和加工相关，即从加工流向加工、数据源流向加工或加工流向数据源。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条数据流。

本问题考查的是上下文 DFD，要求确定外部实体。在上下文 DFI)中，待系统名称“证券交易平台”作为唯一加工的名称，外部实体为这个唯一加工提供输入数据流或者接收其输出数据流。通过考查系统的主要功能，发现系统中涉及到客户服务助理、客户和经纪人，没有提到其他与系统交互的外部实体。

根据描述(1) 中“客户服务助理提交的开户信息”，(2) 中“客户可以向其账户中存款”、(3) 中“客户可以从其账户中取款”，(1) 中“客户和经纪人均可以进行证券交易”，以及(5) 中“将交易明细返回给客户”等信息，对照图 1-1，从而即可确定 E1 为“客户服务助理”实体， E2 为“客户”实体， E3 为“经纪人”实体。
D1：客户记录， D2：账户记录， D3：交易记录。
本问题要求确定图 1-2 中 0 层数据流图中的数据存储。重点分析说明中与数据存储有关的描述。说明(1) 中“并将客户信息存入客户记录中，账户信息(余额等)存入账户记录中”，可知 D1 为客户记录、 D2 为账户记录；说明(5) 中“平台从交易记录中读取交易信息”，可知 D3 为交易记录。

本问题要求补充缺失的数据流及其起点和终点。对照图 1-1 和图 1-2 的输入、输出数据流，数量和名称均相同，所以需要从内部确定缺失的数据流。
考查说明中的功能，先考查说明(2) /(3) 中“客户可以向其账户中存款/取款，根据存款金额修改账户余额”，加工存款与取款分别需要有到数据存储账户记录(D2)标识余额的数据流，图 1-2 中加工存款与取款没有到数据存储账户记录的数据流。再考查说明(4) 中“客户和经纪人均可以进行证券交易(客户通过在线方式，经纪人通过电话)，将交易信息存入交易记录中”，图 1-2 中加工证券交易(在线)和证券交易(电话)分别需要有到交易记录标识交易信息的数据流。

在图 1-1 中，将“证券交易中心”作为外部实体，添加从“证券交易平台”到此外部实体的数据流“交易信息”。

在图 1-2 中，将证券交易中心作为外部实体，添加从加工“证券交易(在线)”到此外部实体的数据流“交易信息”，添加从加工“证券交易(电话)”到此外部实体的数据流“交易信息”。

DFD 中，外部实体可以是用户，可以是其他与本系统交互的系统。如果某功能交互的是外部系统，本题中证券交易通常是在证券交易中心完成的，即证券交易中心。此时证券交易中心即为外部实体，而非本系统内部加工，因此需要对图 1-1 和图 1-2 进行修改，添加外部实体“证券交易中心”，并将数据流交易信息的终点全部改为证券交易中心。在图 1-1 中，将“证券交易中心”作为外部实体，添加从“证券交易平台”到此外部实体的数据流“交易信息”。在图 1-2 中，将“证券交易中心”作为外部实体，添加从加工“证券交易(在线)”到此外部实体的数据流“交易信息”，添加从加工“证券交易(电话)”到此外部实体的数据流“交易信息”。

数 据 流	起 点	终 点
余额	存款	D2 或账户记录
余额	取款	D2 或账户记录
交易信息	证券交易（在线）	D3 或交易记录
交易信息	证券交易（电话）	D3 或交易记录

注：以上数据流与顺序无关。

试题二 答案： **解析：** 完善后的实体联系图如下所示(所补苯的联系和类型如虚线所示)：

本题考查数据库系统中实体联系模型(E-R 模型)和关系模式设计方面的基础知识。

①根据题意“每个部门可以有多名员工，每名员工只属于一个部门”，所以部门和员工之间有一个“所属”联系，联系类型为 1 ：*。

②根据题意“每个部门有一名经理，只负责管理本部门的事务”，所以部门和经理之间有一个“负责”联系，联系类型为 1 ： 1 。

③根据题意“一个客户可以有多个预订申请，但一个预订申请对应唯一的一个客户号”，所以客户和预订申请之间有一个“预订”联系，联系类型为 1 ：*。

④根据题意“一个业务员可以安排多个预订申请，一个预订申请只由一个业务员安排，而且可安排多个同类型的客房”，即一份预订申请可以预订多间同类型的客房，所以业务员与客房和预订申请之间的“安排”联系类型为 1 ：*：*。

根据上述分析，完善图 2-1 所示的实体联系图如图 2-2 所示。

(1)

(a)部门号

(b)客户号，单位名称

(c)申请号，客户号

(d)身份证号，入住时间

(2)

“预订申请”关系模式：

主键为申请号

外键为客户号

“安排”关系模式：

主键为客房号，身份证号，入住时间

外键为申请号，客房号，业务员

由于部门和员工之间有一个 1-2^{-8} 的“所属”联系需要将一端的码“部门号”并入多端，故员工关系模式中的空(a)应填写“部门号”。

根据题意，客户信息包括客户号、单位名称、联系人、联系电话、联系地址，给定的客户关系模式中，不含客户号、单位名称，故空(b)应填写“客户号，单位名称”。

由宁预订申请信息包括申请号、客户号、预订入住时间、入住天数、客房类型、客房数量，故空(c)。应填写“申请号，客户号”。

根据题意“客房号、身份证号和入_时间唯一标识安排联系的每一个元组”，所以空(d)应填写“身份证号，入住时间”。

根据题意，“一个申请号对应唯一标识预订申请中的每一个元组”，所以预订申请关系模式的主键为申请号；又因为客户号是蜂户关系的主键，根据外键定义可知，客户号是预订申请关系的外建。

根据题意“客房号、身份证号和入住时间唯一标识安排联系的每一个元组”，所以安排关系模式的主键为客房号，身份证号，入住时间；外键为申请号，客房号，业务员，因为申请号和客房号为预约申请和客房关案的主键，而“业务员”是员工关系子实体必须参考员工关系的主键“员工号”，所以业务员也是外键。

存在问题。

关系模式存在传递依赖，没有达到 3NF 。

应将客房关系模式分解为客房 1(客房号，客房类型，入住状态)客房 2(客房类型，收费标准)。

客房关系模式存在问题。因为客房号为主键，所以客房号可以决定全属性，即客房号→(客房类型，收费标准，入住状态)。又因为客房类型一收费标准，所以该关系模式存在传递依赖，没有达到 3NF ，应将客房关系模式分解为客房 1(客房号，客房类型，入住状态)，客房 2(客房类型，收费标准)。

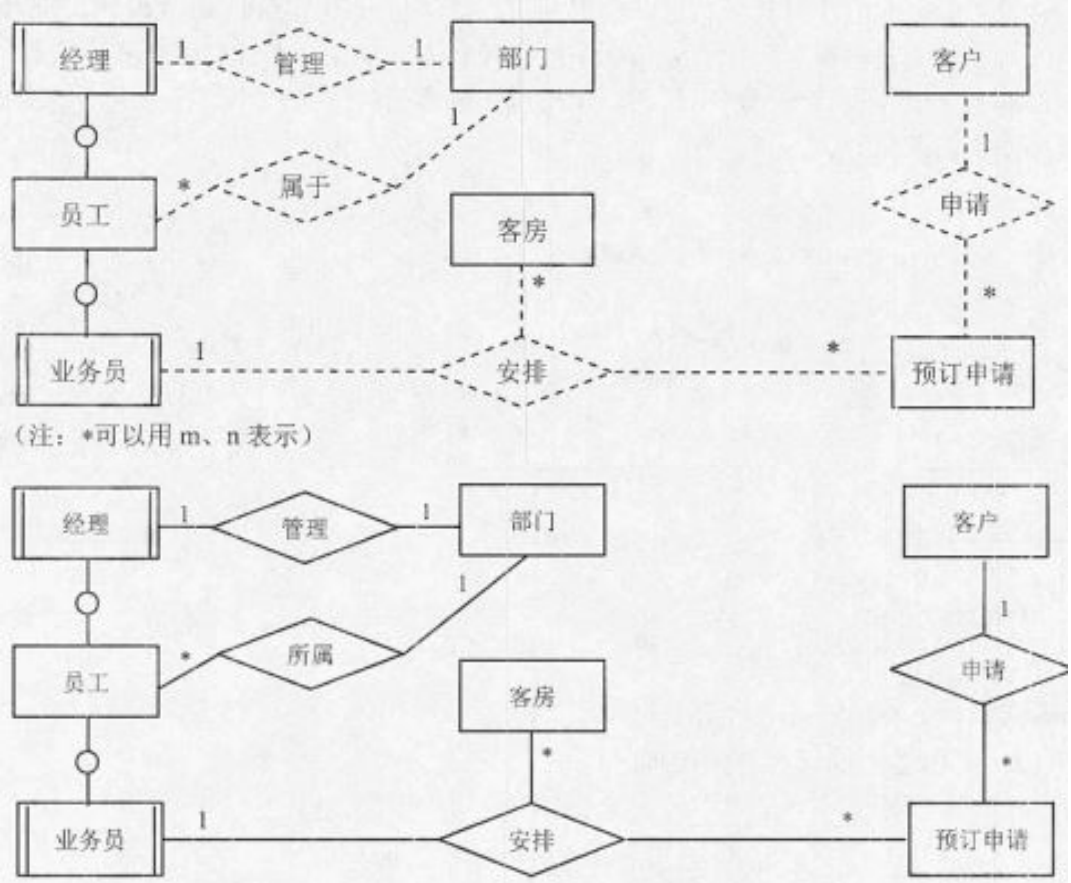


图 2-2　完善的实体联系图

试题三 答案：　解析：　S1：空闲，S2：准备服务，S3：饮料出售，S4：可购买。

本题属于经典的考题，主要考查面向对象分析与设计的基本概念。在建模方面，本题涉及到了用例图、状态图和类图。用例是描述系统功能需求的一种常用方法，用例规约是创建需求模型，进行系统设计的依据。本题的考点就是由用例规约创建状态图和类图。

题目说明中已经给出了自动售货机的 4 个状态分别是：“空闲”状态、“准备服务”状态、“可购买”状态以及“饮料出售”状态。解答本题需要根据用例规约推出这 4 个状态之间的迁移关系，这样才能与图 3-2 中的状态 S1　S4 对应。

首先从状态图的初始状态“●”开始，S1 代表的就是自动售货机的初始状态。在上述 4 个状态中，只有在“空闲”下，才能开始一次售卖，所以 S1 对应的是“空闲”状态。

根据 S2 相关的事件来看，在该状态时，自动售货机在接收顾客的请求(顾客选择的饮料以及投入的硬币)，因此应对应“准备服务”状态。

状态 S3 有一个自迁移事件“计算找零”，根据说明和用例规约可知，饮料出售之后进行找零，所以 S3 对应“饮料出售”状态。S4 则对应“可购买”状态。

E1：所选购的饮料数量不足

E2：选择饮料[硬币足够购买饮料]

E3：所选购的饮料数量足够/推出饮料

E4：取走饮料/找零并清空硬币器

确定了状态图中的各个状态，接下来就需要进行状态之间迁移事件的获取。E2 是从“准备服务”状态变换到“可购买”状态的事件，“选择饮料[硬币不足]”时仍然停留在“准备服务”状态，对应用例规约中的 2a；根据用例规约若硬币足够则进入下一步，所以 E2 所对应的事件是“选择饮料[硬币足够购买饮料]”。

E1 事件的触发将使得自动售货机从“可购买”状态变换到“准备服务”状态，对应用例规约中的 3a，所以 E1 对应的事件应是“所选购的饮料数量不足”。

E3 事件的触发将使得自动售货机从“可购买”状态年移到“饮料出售”状态：。根据说明，能够售出饮料必须满足两个条件：该饮料数量，足够以及顾客投入的硬币足够。硬币是否足够以及饮料数量不足在状态 S2 已经进行了判断，因此 E3 对应的事件应是“所选购的饮料数量足够/推出饮料”。

E4 对应的事件是自动售货机完成售卖，回到“空闲”状态时需处理的事件，根据说明可知，E4 对应的事件应是“取走饮料/找零并清空硬币器”。

C1：自动售货机 C2：硬币器 C3：饮料储存仓 C4：硬币 C5：饮料

或者

C1：自动售货机 C2：饮料储存仓 C3：硬币器 C4：饮料 C5：硬币

本题要求根据说明和用例约创建对应的类模型。根据说明和用例规约可知自动售货机有几个重要的组成元素：饮料、硬币、硬币器和饮料存储仓。1 台自动售货机有 1 个硬币器、1 个饮料存储仓；硬币器可以接收多枚硬币，饮料存储仓中可以容纳多种饮料。由此可知，图 3-3 中的两个 0..*聚集关系应该分别对应“硬币器-硬币”和“饮料存储仓-饮料”这两对“部分-整体”关系；而 C1 就是自动售货机。

试题四 答案：　解析：　(1) j

(2) t[i]==s[j];

(3) j=next[j]

(4) i-j+1

本题考查算法设计与分析以及用 C 程序设计语言实现算法的能力。

KMP 算法是一个非常经典的模式匹配算法。其核心思想是核心思想：匹配过程中字符对不相等时，不需回溯主串，而是利用已经得到的部分匹配结果将模式向右滑动尽可能远的一

段距离继续比较。滑动的距离由 next 数组给出。该算法提出之后，有一些改进的思想，使得 next 数组的计算有多种方式。本题干不需要考生考虑如何计算 next 数组，已经直接给出计算该数组的 C 代码。只需要根据已经计算的 next 数组进行模式匹配即可。

在 C 函数 kmp 中， while 循环是判断串 s 和 t 是否还有字符，因此空(1) 处应填写“j

(5) O(ls+lt)

在 kmp 函数中，只有一个 while 循环，该算法的时间复杂度为 O(ls+lt)。

(6) -1，-1， 1，-1，-1， 2， 0， 0

(7) 6

根据 C 函数 get_next，得到“BABBCAC”的 next 数组的值为-1，-1， 1，-1，-1， 2，0， 0。对主串为揶 ABB“CBBABBCACCD”和上述模式串，得到匹配位置为 6，这里需要注意的是，位置从 1 开始。

试题五 答案： 解析： (1) virtual void printInvoice()

(2) ticket->printInvoice() ;

(3) Decorator::printInvoice()

(4)Decorator::printInvoice()

(5) &a

本题考查装饰(Decorator)模式的基本概念和应用。

装饰模式属于结构型设计模式，其设计意图是动态地给一个对象添加一些额外的职责。就增加功能而言，装饰模式比生成子类更加灵活。装饰模式的结构如图 5-2 所示。

其中：

•Component 定义一个对象接口，可以给这些对象动态地添加职责。

•ConcreteComponent 定义一个对象，可以给这个对象添加一些职责。

•Decorator 维持一个指向 Component 对象的指针，并定义一个与 Component 接口一致的接口。

•ConcreteDecorator 向组件添加职责。

装饰模式适用于：

•在不影响其他对象的情况下，以动态、透明的方式给单个对象添加职责。

•处理那些可以撤销的职责。

•当不能采用生成子类的方式进行扩充时。一种情况是，可能有大量独立的扩展，为支持每一种组合将产生大量的子类，使得子类数目呈爆炸性增长。另一种情况可能是，由于类定

义被隐藏，或类定义不能用于生成子类。

本题将装饰模式用于实现打印发票问题。图 5-1 的类图中，类 Invoice 对应图 5-2 中的 Component ，其功能是打印发票的内容； HeadDecorator 和 FoctDecorator 是两个 ConcreteDecorator ，向组件中添加打印发票抬头和发票脚注的功能。

方法 printlnvoice 是 Invoice 中定义的接口， Component 类中应定义一个与之一致的接口。在 C++中，父类和子类之间共享接口，通常采用虚拟函数。由此可知，空(1) 处应填写″virtualvoidprintInvoice() ”。这个接口在类 Decorator 、 HeadDecorator 和 FootDecorator 中分别进行了重置，分别对应代码中的空(2) （ 4)。

类 Decorator 中保持了一个指向 Component 对象的指针 ticket ，用来接收所要装饰的组件 Invoice 。因此空(2) 处应填写“ticket->printInvoice() ”。类 HeadDecorator 和 FootDecomtor 是在打印发票内容的基础上，打印发票的抬头和脚注，所以空(3) 、(4)处都应填写“Decorator : : printInvoice() ”。

最后一空考查的是装饰模式的调用，由 main() 函数中给出的第一次调用可以获得一些提示，推断出空(5) 出应填写“&a”。

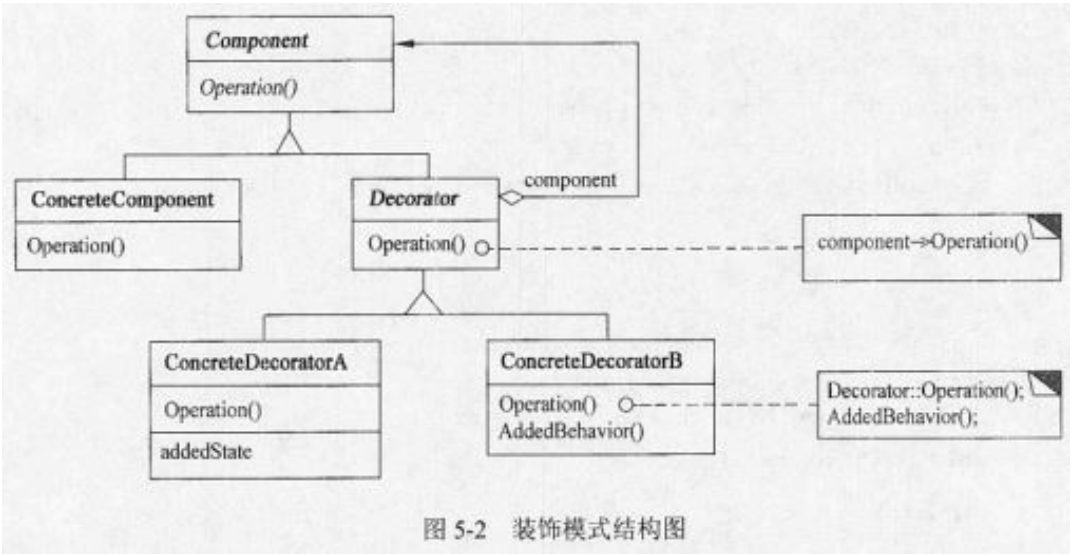


图 5-2 装饰模式结构图

试题六 答案： **解析：** (1) ticket.printInvoice()

(2) super.printInvoice()

(3) super.printInvoice()

(4)new HeadDecorator(new FootDecorator(t))

(5) new FootDecorator(new HeadDecorator(null))

本题考查装饰(Decorator)模式的基一概念和应用。

装饰模武属于结构型设计模式，其设计意图是动态地给一个对象添加些额外的职责，就增

加功能而倉装饰模式比生成子类更加灵活。装饰模式的结构如图 6-2 所示。

其中：

•Component 定义一个对象接口，可以给这些对象动态地添加职责。

•ConcreteComponent 定义一个对象，可以给这个对象添加一些职责。

•Decorator 维持一个指向 Component 对象的指针，并定义一个与 Component 接口一致的接口。

•ConcreteDecorator 向组件添加职责。

装饰模式适用于：

•在不影响其他对象的情况下，以动态、透明的方式给单个对象添加职责。

•处理那些可以撤销的职责。

•当不能采用生成子类的方式进行扩充时。一种情况是，可能有大量独立的扩展，为支持每一种组合将产生大量的子类，使得子类数目呈爆炸性增长。另一种情况可能是，由于类定义被隐藏。或类定义不能用于生成子类。

本题将装饰模式用于实现打印发票问题。图 6-1 的类图中，类 Invoice 对应图 6-2 中的 Component ，其功能是打印发票的内容； HeadDecorator 和 FootDecorator 是两个 ConcreteDecorator ，向组件中添加打印发票头和发票脚注的功能。

方法 printlnvoice 是 Invoice 中定义的接口， Component 类中应定义一个与之一致的接口。这个接口在类 Decorator 、 HeadDecorator 和 FootDecorator 中分别进行了重新定义，分别对应代码中的空(1) （ 3）。

类 Decorator 中保持了一个 Component 对象——ticket ，用来接收所要装饰的组件 Invoiceo 因此空(1) 处应填写“ticket.printInvoice() ”。类 HeadDecorator 和 FootDecorator 是在打印发票内容的基础上，打印发票的抬头和脚注，所以空(2) 、(3) 处都应填写“super.printInvoice() ”。

空(4) （ 5）考查的是装饰模式的调用，分别应填写为“newHeadDecorator(new FootDecorator(t))”和“newHeadDecorator(newFootDecorator(null))”。

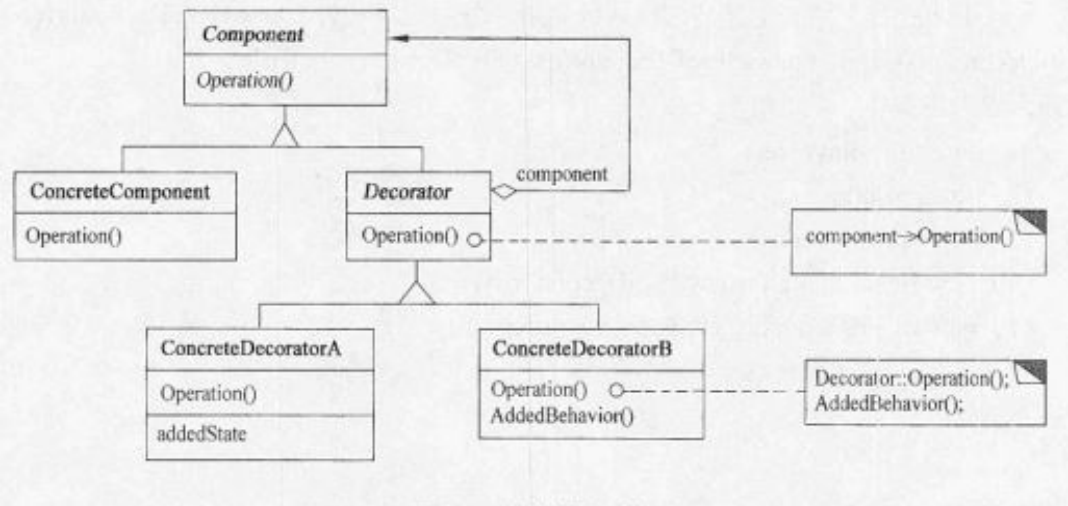


图 6-2　装饰模式结构图