

全国计算机技术与软件专业技术资格（水平）考试

2006 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 7 道题，试题一至试题四是必答题，试题五至试题七选答 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

2006 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 20 日”，故在答题纸的对应栏内写上“5”和“20”（参看下表）。

例题	解答栏
(1)	5
(2)	20

试题一（15分）

阅读下列说明以及数据流图，回答问题1、问题2和问题3，将解答填入答题纸的对应栏内。

【说明】

某学校建立了一个网上作业提交与管理系统，基本功能描述如下：

(1) 帐号和密码。任课老师用帐号和密码登录系统后，提交所有选课学生的名单。系统自动为每个选课学生创建登录系统的帐号和密码。

(2) 作业提交。学生使用帐号和密码登录系统后，可以向系统申请所选课程的作业。系统首先检查学生的当前状态，如果该学生还没有做过作业，则从数据库服务器申请一份作业。若申请成功，则显示需要完成的作业。学生需在线完成作业，单击“提交”按钮上交作业。

(3) 在线批阅。系统自动在线批改作业，显示作业成绩，并将该成绩记录在作业成绩统计文件中。

【问题 1】（3 分）

如果将数据库服务器（记为 DB）作为一个外部实体，那么在绘制该系统的数据流图时，还应有哪些外部实体和数据存储？

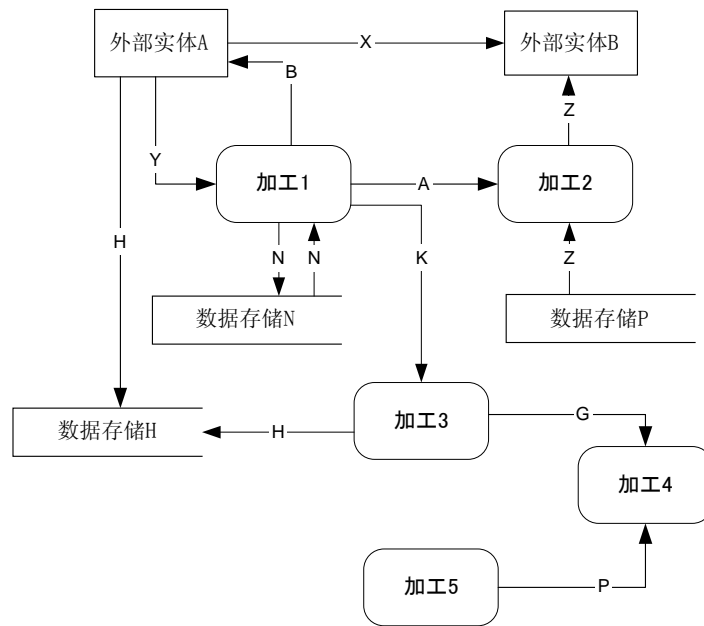
【问题 2】（7 分）

根据说明结合问题 1 的解答，指出在该系统的顶层数据流图中应有哪些数据流。请采用说明中的词汇给出这些数据流的起点、终点以及数据流名称，下表给出了数据流的部分信息，请填充空缺处。

序号	起点	终点	数据流名称
1	____(1)____	网上作业提交与管理系统	作业申请
2	____(2)____	网上作业提交与管理系统	提交的作业
3	网上作业提交与管理系统	____(3)____	需完成的作业
4	网上作业提交与管理系统	____(4)____	____(5)____
5	网上作业提交与管理系统	____(6)____	作业申请
6	网上作业提交与管理系统	____(7)____	____(8)____
7	____(9)____	网上作业提交与管理系统	选课学生名单
8	____(10)____	网上作业提交与管理系统	____(11)____
9	____(12)____	网上作业提交与管理系统	帐号和密码
10	____(13)____	网上作业提交与管理系统	帐号和密码

【问题3】(5分)

根据数据流图的设计原则，阅读下图所示的数据流图，找出其中的错误之处。



试题二(15分)

阅读下列说明以及UML类图，回答问题1、问题2和问题3，将解答填入答题纸的对应栏内。

【说明】

某客户信息管理系统中保存着两类客户的信息：

(1) 个人客户。对于这类客户，系统保存了其客户标识（由系统生成）和基本信息（包括姓名、住宅电话和 email）。

(2) 集团客户。集团客户可以创建和管理自己的若干名联系人。对于这类客户，系统除了保存其客户标识（由系统生成）之外，也保存了其联系人的信息。联系人的信息包括姓名、住宅电话、email、办公电话以及职位。

该系统除了可以保存客户信息之外，还具有以下功能：

- (1) 向系统中添加客户（addCustomer）；
- (2) 根据给定的客户标识，在系统中查找该客户（getCustomer）；
- (3) 根据给定的客户标识，从系统中删除该客户（removeCustomer）；
- (4) 创建新的联系人（addContact）；
- (5) 在系统中查找指定的联系人（getContact）；
- (6) 从系统中删除指定的联系人（removeContact）。

该系统采用面向对象方法进行开发。在面向对象分析阶段，根据上述描述，得到如表 2-1 所示的类。

表 2-1

类名	说明
CustomerInformationSystem	客户信息管理系统
IndividualCustomer	个人客户
InstitutionalCustomer	集团客户
Contact	联系人

描述该客户信息管理系统的 UML 类图如图 2-1 所示。

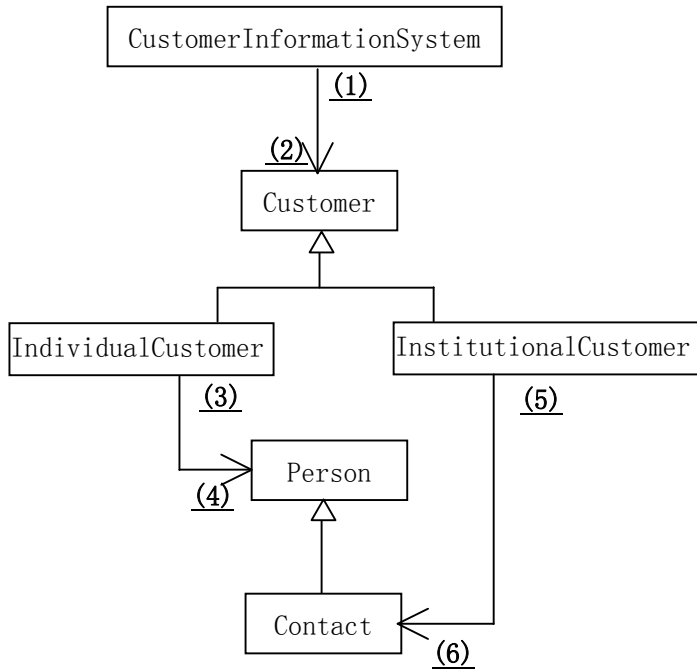


图 2-1 客户信息管理系统 UML 类图

【问题 1】(3 分)

请使用说明中的术语，给出图 2-1 中类 Customer 和类 Person 的属性。

【问题 2】(6 分)

识别关联的多重度是面向对象建模过程中的一个重要步骤。根据说明中给出的描述，完成图中的(1)~(6)。

【问题 3】(6 分)

根据说明中的叙述，抽象出如表 2-2 所示的方法，请指出图 2-1 中的类 CustomerInformationSystem 和 InstitutionalCustomer 应分别具有其中的哪些方法。

表 2-2

功能描述	方法名
向系统中添加客户	addCustomer
根据给定的客户标识，在系统中查找该客户	getCustomer
根据给定的客户标识，从系统中删除该客户	removeCustomer
创建新的联系人	addContact
在系统中查找指定的联系人	getContact
从系统中删除指定的联系人	removeContact

试题三(15分)

阅读下列说明,回答问题1、问题2和问题3,将解答填入答题纸的对应栏内。

【说明】

某单位资料室需要建立一个图书管理系统,初步的需求分析结果如下:

(1) 资料室有图书管理员若干名,他们负责已购入图书的编目和借还工作,每名图书管理员的信息包括工号和姓名;

(2) 读者可在阅览室读书,也可通过图书流通室借还图书,读者信息包括读者 ID、姓名、电话和 Email,系统为不同读者生成不同的读者 ID;

(3) 每部书在系统中对应惟一的一条图书在版编目数据(CIP,以下简称书目),书目的基本信息包括 ISBN 号、书名、作者、出版商、出版年月,以及本资料室拥有该书的册数(以下简称册数),不同书目的 ISBN 号不相同;

(4) 资料室对于同一书目的图书可拥有多册(本),图书信息包括图书 ID、ISBN 号、存放位置、当前状态,每一本书在系统中被赋予惟一的图书 ID;

(5) 一名读者最多只能借阅十本图书,且每本图书最多只能借两个月,读者借书时需由图书管理员登记读者 ID、所借图书 ID、借阅时间和应还时间,读者还书时图书管理员在对应的借书信息中记录归还时间;

(6) 当某书目的可借出图书的数量为零时,读者可以对其进行预约登记,即记录读者 ID、需要借阅的图书的 ISBN 号、预约时间。

某书目的信息如表 3-1 所示,与该书目对应的图书信息如表 3-2 所示。

表 3-1 书目信息

书名	作者	出版商	ISBN 号	出版年月	册数	经办人
《数据结构》	严蔚敏 吴伟民	清华大学出版社	ISBN7-302-02368-9	1997.4	4	01

表 3-2 图书信息

图书 ID	ISBN 号	存放位置	状态	经办人
C832.1	ISBN7-302-02368-9	图书流通室	已借出	01
C832.2	ISBN7-302-02368-9	图书阅览室	不外借	01
C832.3	ISBN7-302-02368-9	图书流通室	未借出	01
C832.4	ISBN7-302-02368-9	图书流通室	已预约	01

系统的主要业务处理如下:

(1) 入库管理:图书购进入库时,管理员查询本资料室的书目信息,若该书的书目尚未建立,则由管理员编写该书的书目信息并录入系统,然后编写并录入图书信息;否则,修改该书目的册数,然后编写并录入图书信息,对于进入流通室的,其初始状态为“未借出”,而送入阅览室的,其状态始终为“不外借”。

(2) 借书管理:读者借书时,若有,则由管理员为该读者办理借书手续,并记录该读者的借书信息,同时将借出图书的状态修改为“已借出”。

(3) 预约管理:若图书流通室没有读者要借的书,则可为该读者建立预约登记,需要记录读者 ID、书的 ISBN 号、预约时间和预约期限(最长为 10 天)。一旦其他读者归还这种书,就自动通知该预约读者。系统将自动清除超出预约期限的预约记录并修改相关信息。

(4) 还书管理：读者还书时，则记录相应借还信息中的“归还时间”，对于超期归还者，系统自动计算罚金（具体的计算过程此处省略）。系统同时自动查询预约登记表，若存在其他读者预约该书的记录，则将该图书的状态修改为“已预约”，并将该图书 ID 写入相应的预约记录中（系统在清除超出预约期限的记录时解除该图书的“已预约”状态）；否则，将该图书的状态修改为“未借出”。

(5) 通知处理：对于已到期且未归还的图书，系统通过 Email 自动通知读者；若读者预约的书已到，系统则自动通过 Email 通知该读者来办理借书手续。

【问题 1】（4 分）

根据以上说明设计的实体联系图如图 3-1 所示，请指出读者与图书、书目与读者、书目与图书之间的联系类型。

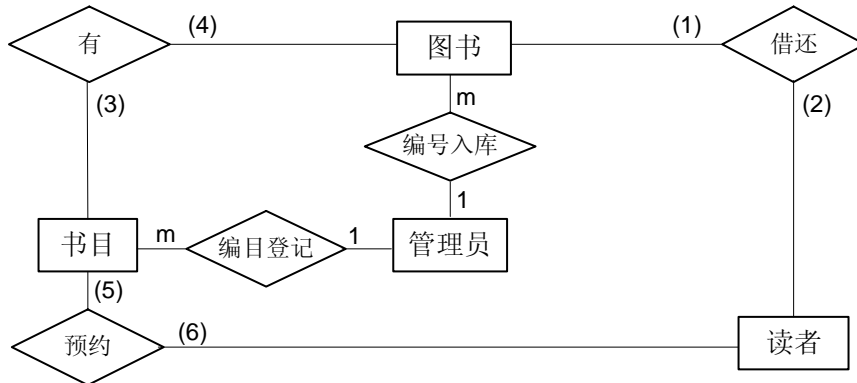


图 3-1 图书管理系统的实体联系图

【问题 2】（4 分）

该图书管理系统的主要关系模式如下，请补充“借还记录”和“预约登记”关系中的空缺。

管理员(工号, 姓名)

读者(读者 ID, 姓名, 电话, Email)

书目(ISBN 号, 书名, 作者, 出版商, 出版年月, 册数, 经办人)

图书(图书 ID, ISBN 号, 存放位置, 状态, 经办人)

借还记录(_____(a)_____, 借出时间, 应还时间, 归还时间)

预约登记(_____(b)_____, 预约时间, 预约期限, 图书 ID)

注：时间格式为“年.月.日 时:分:秒”

【问题 3】（7 分）

请指出问题 2 中给出的读者、书目关系模式的主键，以及图书、借还记录和预约登记关系模式的主键和外键。

试题四(15分)

阅读以下说明，回答问题 1、问题 2 和问题 3，将解答填入答题纸的对应栏内。

【说明】

某单位正在使用一套 C/S 模式的应用软件系统，现在需要升级为 B/S 应用模式，但需要保持业务的连续性。开发人员提出用 Web Service 作为中间层的接口进行开发。

【问题 1】(6 分)

请用 120 字以内文字，从业务的继承性、升级成本（时间、工作量）和扩展性三个方面简要说明开发人员所提方案的优点。

【问题 2】(3 分)

Web Service 的三个基本技术是 WSDL、SOAP、UDDI，它们都是以 XML 为基础定义的。请用 120 字以内文字，简要说明 WSDL、SOAP 和 UDDI 的作用。

【问题 3】(6 分)

服务注册中心、服务提供者和服务请求者之间的交互和操作构成了 Web Service 的体系结构，如图 4-1 所示。请用 180 字以内文字，说明这三者的主要功能及其交互过程。

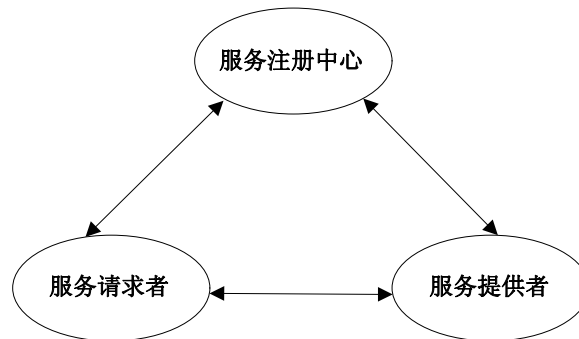


图 4-1 Web Service 的工作模式

试题五(15分)

阅读下列说明、图和C代码，将应填入(n)处的字句写在答题纸的对应栏内。

【说明 5-1】

B树是一种多叉平衡查找树。一棵m阶的B树，或为空树，或为满足下列特性的m叉树：

- ① 树中每个结点至多有m棵子树；
- ② 若根结点不是叶子结点，则它至少有两棵子树；
- ③ 除根之外的所有非叶子结点至少有 $\lceil m/2 \rceil$ 棵子树；
- ④ 所有的非叶子结点中包含下列数据信息

$(n, A_0, K_1, A_1, K_2, A_2, \dots, K_n, A_n)$

其中： $K_i (i=1, 2, \dots, n)$ 为关键字，且 $K_i < K_{i+1} (i=1, 2, \dots, n-1)$ ； $A_i (i=0, 1, \dots, n)$ 为指向子树根结点的指针，且指针 A_{i-1} 所指子树中所有结点的关键字均小于 K_i ， A_{i+1} 所指子树中所有结点的关键字均大于 K_i ，n为结点中关键字的数目。

⑤ 所有的叶子结点都出现在同一层次上，并且不带信息（可以看作是外部结点或查找失败的结点，实际上这些结点不存在，指向这些结点的指针为空）。

例如，一棵4阶B树如图5-1所示（结点中关键字的数目省略）。

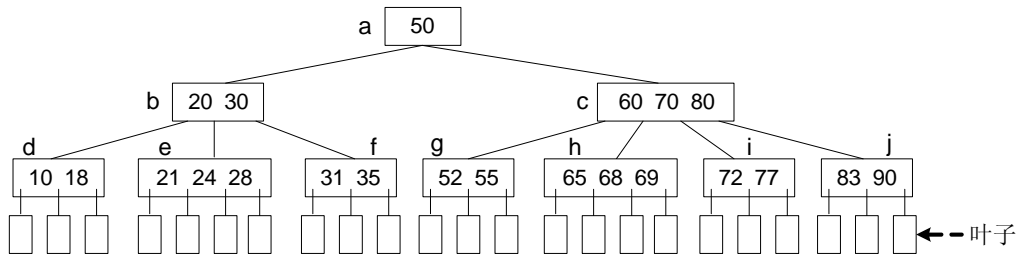


图 5-1 4阶B树示例

B树的阶M、bool类型、关键字类型及B树结点的定义如下：

```
#define M 4 /*B树的阶*/
typedef enum {FALSE = 0, TRUE = 1} bool;
typedef int ElemKeyType;
typedef struct BTreeNode{
    int numkeys; /*结点中关键字的数目*/
    struct BTreeNode *parent ; /*指向父结点的指针，树根的父结点指针为空*/
    struct BTreeNode *A[M]; /*指向子树结点的指针数组*/
    ElemKeyType K[M]; /*存储关键字的数组，K[0]闲置不用*/
}BTreeNode;
```

函数 SearchBtree(BTreeNode* root, ElemKeyType akey, BTreeNode **ptr)的功能是：在给定的一棵 M 阶 B 树中查找关键字 akey 所在结点，若找到则返回 TRUE，否则返回 FALSE。其中，root 是指向该 M 阶 B 树根结点的指针，参数 ptr 返回 akey 所在结点的指针，若 akey 不在该 B 树中，则 ptr 返回查找失败时空指针所在结点的指针。例如，在图 5-1 所示的 4 阶 B 树中查找关键字 25 时，ptr 返回指向结点 e 的指针。

注：在结点中查找关键字 akey 时采用二分法。

【函数 5-1】

```
bool SearchBtree(BTreeNode* root, ElemKeyType akey, BTreeNode **ptr)
{
    int lw, hi, mid;
    BTreeNode *p = root;
    *ptr = NULL;
    while ( p ) {
        lw = 1;    hi = ____ (1) ____;
        while (lw <= hi) {
            mid = (lw + hi) / 2;
            if (p -> K[mid] == akey) {
                *ptr = p;
                return TRUE;
            }
            else
                if (____ (2) ____ )
                    hi = mid - 1;
                else
                    lw = mid + 1;
        }
        *ptr = p;
        p = ____ (3) ____;
    }
    return FALSE;
}
```

【说明 5-2】

在 M 阶 B 树中插入一个关键字时，首先在最接近外部结点的某个非叶子结点中增加一个关键字，若该结点中关键字的个数不超过 M-1，则完成插入；否则，要进行结点的“分裂”处理。所谓“分裂”，就是把结点中处于中间位置上的关键字取出来并插入其父结点中，然后以该关键字为分界线，把原结点分成两个结点。“分裂”过程可能会一直持续到树根，若树根结点也需要分裂，则整棵树的高度增 1。

例如，在图 5-1 所示的 B 树中插入关键字 25 时，需将其插入结点 e 中，由于 e 中已经有 3 个关键字，因此将关键字 24 插入结点 e 的父结点 b，并以 24 为分界线将结点 e 分裂为 e1 和 e2 两个结点，结果如图 5-2 所示。

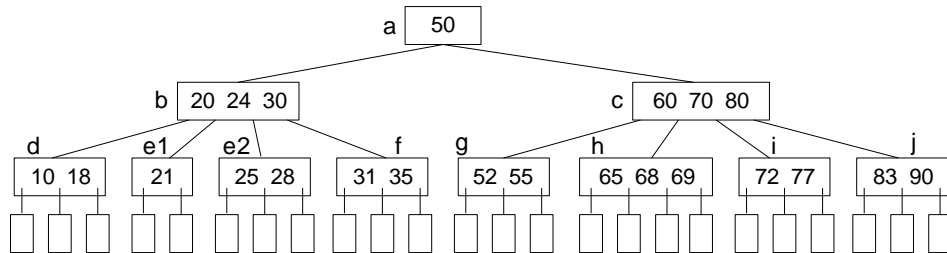


图 5-2 在图 5-1 所示的 4 阶 B 树中插入关键字 25 后的 B 树

函数 `Isgrowing(BTreeNode* root, ElemKeyType akey)` 的功能是：判断在给定的 M 阶 B 树中插入关键字 `akey` 后，该 B 树的高度是否增加，若增加则返回 `TRUE`，否则返回 `FALSE`。其中，`root` 是指向该 M 阶 B 树根结点的指针。

在函数 `Isgrowing` 中，首先调用函数 `SearchBtree`（即函数 5-1）查找关键字 `akey` 是否在给定的 M 阶 B 树中，若在则返回 `FALSE`（表明无需插入关键字 `akey`，树的高度不会增加）；否则，通过判断结点中关键字的数目考察插入关键字 `akey` 后该 B 树的高度是否增加。

【函数 5-2】

```
bool Isgrowing(BTreeNode* root, ElemKeyType akey)
{
    BTreeNode *t, *f;
    if ( !SearchBtree(__(4)__) ) {
        t = f;
        while (__(5)__) {
            t = t -> parent;
        }
        if ( !t )
            return TRUE;
    }
    return FALSE;
}
```

试题六 (15分)

阅读下列说明、图和 C++代码，将应填入__(n)__处的字句写在答题纸的对应栏内。

【说明】

某订单管理系统的部分 UML 类图如图 6-1 所示。

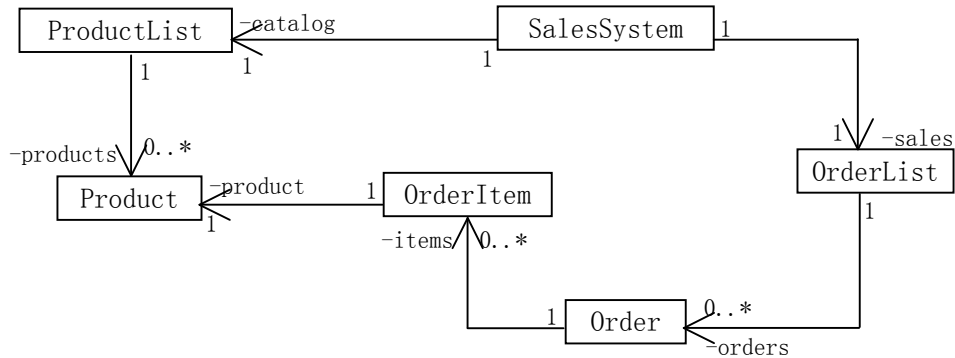


图 6-1 订单管理系统的部分 UML 类图

图 6-1 中，Product 表示产品，ProductList 表示产品目录，Order 表示产品订单，OrderItem 表示产品订单中的一个条目，OrderList 表示订单列表，SalesSystem 提供订单管理系统的操作接口。

请完善类 Order 的成员函数 getOrderedAmount() 和类 SalesSystem 的 statistic() 方法，各个类的属性及部分方法定义参见下面的 C++代码。

【C++代码】

```
class Product { //产品类
private:
    string pid; //产品识别码
    string description; //产品描述
    double price; //产品单价
public:
    void setProductPrice(double price); //设置产品单价
    string getProductId(); //获取产品识别码
    string getProductDescription (); //获取产品描述
    double getProductPrice(); //获取产品单价
    //其他成员省略
};

class ProductList { //产品列表类
private:
    vector <Product> products;
public:
    ProductList();
```

```

    Product getProductByIndex(int i);           //获得产品列表中的第 i 件产品
    void addProduct(Product t);               //在产品列表中加入一件产品
    Product * getProductByID(string pid);     //获得识别码为 pid 的产品指针
    unsigned int getProductAmount();          //获得产品列表中的产品数量
};

class OrderItem {                             //订单条目类
private:
    Product *productPtr;                       //指向被订购产品的指针
    int quantity;                              //订购数量
public:
    OrderItem (Product *,int);
    Product * getProductptr ();               //获取指向被订购产品的指针
    int getQuantity ();                       //获取被订购产品的数量
};

class Order {                                 //订单类
private:
    unsigned int orderid;                      //订单识别号
    vector<OrderItem> items;                  //订单内容(订单项)
public:
    Order(unsigned int orderid);
    //获得识别码为 tid 的产品在当前订单中被订购的数量
    int getOrderedAmount(string tid);
    void additem(Product *productPtr,unsigned int n);//在订单中增加一个订单项
    //其他成员省略
};

class OrderList {                             //订单列表类
private:
    vector<Order> orders;
public:
    OrderList();
    //Begin() 返回指向订单列表第一个元素的迭代器(指针)
    virtual vector<Order>::iterator OrderList::Begin();
    //End() 返回指向订单列表最后一个元素之后的迭代器(指向一个不存在的元素)
    virtual vector<Order>::iterator OrderList::End();
    void addOrder(Order t);                   //在订单列表中加入一份订单
    //其他成员省略
};

```

```

class SalesSystem{
private:
    ProductList catalog;    //产品目录
    OrderList sales;       //订单列表
public:
    SalesSystem();
    void statistic();      //统计所有产品的订购情况
    //其他成员省略
};

//在订单中查找识别码为 tid 的产品的订购数量，若该产品没有被订购，则返回 0
int Order::getOrderedAmount(string tid)
{
    for (int k = 0; k < items.size(); k++){
        if ( ____ (1) ____ == tid)
            return ____ (2) ____;
    }
    return 0;
}

// 方法 statistic() 依次统计产品目录中每个产品的订购总量，并打印输出
// 每个产品的识别码、描述、订购总量和订购金额
void SalesSystem::statistic()
{
    unsigned int k, t, ordered_qty = 0;
    vector<Order>::iterator it;    Product p;
    cout << "产品识别码\t 描述\t\t 订购数量\t 金额" << endl;

    for (k = 0; k < catalog.getProductAmount(); k++) { //遍历产品列表
        p = ____ (3) ____;    //从产品列表取得一件产品信息存入变量 p
        ordered_qty = 0;
        //通过迭代器变量 it 遍历订单列表中的每一份订单
        for (it = sales.Begin(); ____ (4) ____; it++) {
            //根据产品识别码获得产品 p 在当前订单中被订购的数量
            t = ____ (5) ____ (p.getProductId());
            ordered_qty += t;
        }
        cout << p.getProductId() << "\t\t" << p.getProductDescription() << "\t\t";
        cout << ordered_qty << "\t\t" << p.getProductPrice() * ordered_qty << endl;
    }
}
}

```

试题七（15分）

阅读下列说明、图以及 Java 程序，将应填入 (n) 处的字句写在答题纸的对应栏内。

【说明】

某订单管理系统的部分 UML 类图如图 7-1 所示。

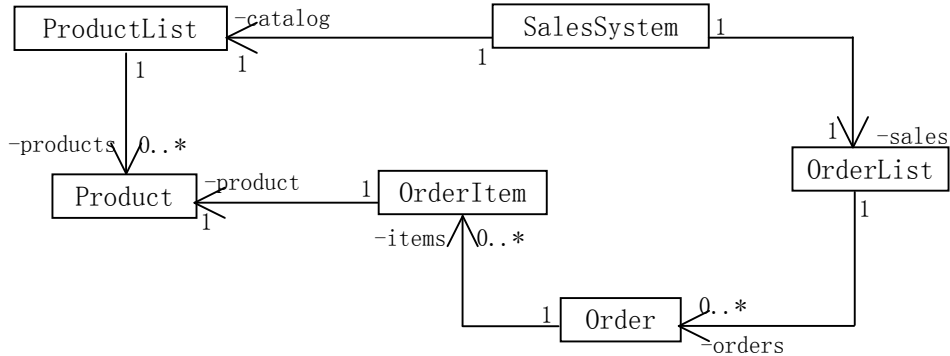


图 7-1 订单管理系统的部分 UML 类图

图 7-1 中，Product 表示产品，ProductList 表示所销售产品的列表，Order 表示产品订单，OrderItem 表示产品订单中的一个条目，OrderList 表示订单列表，SalesSystem 提供订单管理系统的操作接口。各个类的部分属性和方法说明如表 7-1 所示。

表 7-1

类	成员	说明
ProductList	ArrayList<Product> products	
Product	String code	产品编号
	String description	产品描述
	double price	产品单价
	Boolean equals(Object object)	若两个产品相同则返回 true，否则返回 false
OrderItem	Product product	订单项中的产品
	int quantity	产品的订购数量
	Product getProduct()	获取订单项中的产品
Order	ArrayList<OrderItem> items	订单中包含的订单项
OrderList	ArrayList<Order> orders	订单
	void addOrder(Order order)	向订单列表中添加新订单
	int getNumberOfOrders()	获取订单列表中的订单总数
SalesSystem	ProductList catalog	产品目录
	OrderList sales	订单列表
	void statistic()	依次统计产品目录中每个产品的订购总量，并打印出每个产品的编号、说明、订购总量和订购金额

可以使用类 `java.util.ArrayList<E>` 来实现对象的聚集关系，如图 7-1 中 `OrderList` 与 `Order` 之间的聚集关系。

`for-each` 循环提供了一种遍历对象集合的简单方法。在 `for-each` 循环中，可以指定需要遍历的对象集合以及用来接收集合中每个元素的变量，其语法如下：

`for(用来接收集合中元素的变量 : 需要遍历的对象集合)`

如果要使用 `for-each` 循环来遍历对象集合，那么包含该对象集合的类必须实现接口 `java.util.Iterable<T>`。

Java 程序 7-1 和 Java 程序 7-2 分别给出了类 `OrderList` 和方法 `statistic` 的 Java 代码。

【Java 程序 7-1】

```
import java.util.*;
public class OrderList ____ (1) ____ {
    private ArrayList<Order> orders;
    public OrderList() {
        this.orders = new ArrayList<Order>();
    }
    public void addOrder(Order order) {
        this.orders.add(order);
    }
    public Iterator<Order> iterator() {
        return ____ (2) ____;
    }
    public int getNumberOfOrders() {
        return this.orders.size();
    }
}
```

【Java 程序 7-2】

```
import java.util.*;
public class SalesSystem {
    private ProductList catalog;
    private OrderList sales;
    private static PrintWriter stdOut = new PrintWriter(System.out, true);
    public void statistic() {
        for (Product product : ___(3)___) {
            int number = 0;
            for (Order order : ___(4)___) {
                for (___(5)___: order) {
                    if (product.equals(item.getProduct()))
                        number += item.getQuantity();
                }
            }
            stdOut.println(product.getCode() + " "
                + product.getDescription() + " "
                + number + " " + number * product.getPrice());
        }
    }
    // 其余的方法未列出
}
```