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 2012 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 某学校欲开发图书管理系统，以记录图书馆所藏图书及其借出和归还情况，提供给借阅者借阅图书功能，提供给图书馆管理员管理和定期更新图书表功能。主要功能的具体描述如下：

处理借阅。借阅者要借阅图书时，系统必须对其身份(借阅者 ID)进行检查。通过与教务处维护的学生数据库、人事处维护的职工数据库中的数据进行比对，以验证借阅者 ID 是否合法。若合法，则检查借阅者在逾期未还图书表中是否有逾期未还图书，以及罚金表中的罚金是否超过限额。如果没有逾期未还图书并且罚金未超过限额，则允许借阅图书，更新图书表，并将借阅的图书存入借出图书表。借阅者归还所借图书时， 先由图书馆管理员检查图书是否缺失或损坏，若是，则对借阅者处以相应罚金并存入罚金表；然后，检查所还图书是否逾期，若是，执行“处理逾期”操作；最后，更新图书表，删除借出图书表中的相应记录。

维护图书。图书馆管理员查询图书信息；在新进图书时录入图书信息，存入图书表；在图书丢失或损坏严重时，从图书表中删除该图书记录。

处理逾期。系统在每周一统计逾期未还图书，逾期未还的图书按规则计算罚金，并记入罚金表，并给有逾期未还图书的借阅者发送提醒消息。借阅者在借阅和归还图书时，若罚金超过限额，管理员收取罚金，并更新罚金表中的罚金额度。

现采用结构化方法对该图书管理系统进行分析与设计，获得如图 1-1 所示的顶层数据流图和图 1-2 所示的 0 层数据流图。

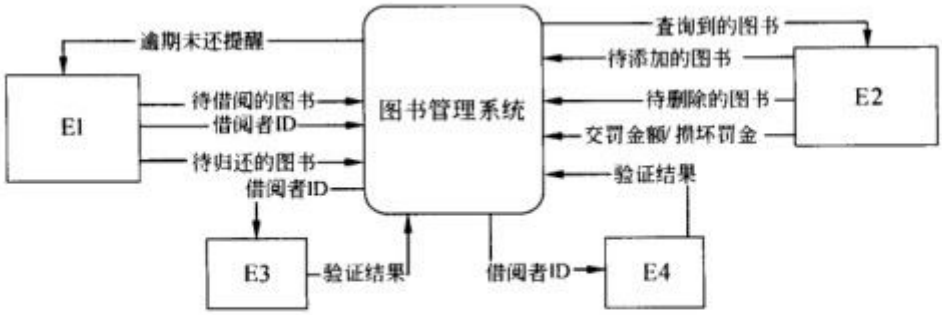


图 1-1　顶层数据流图

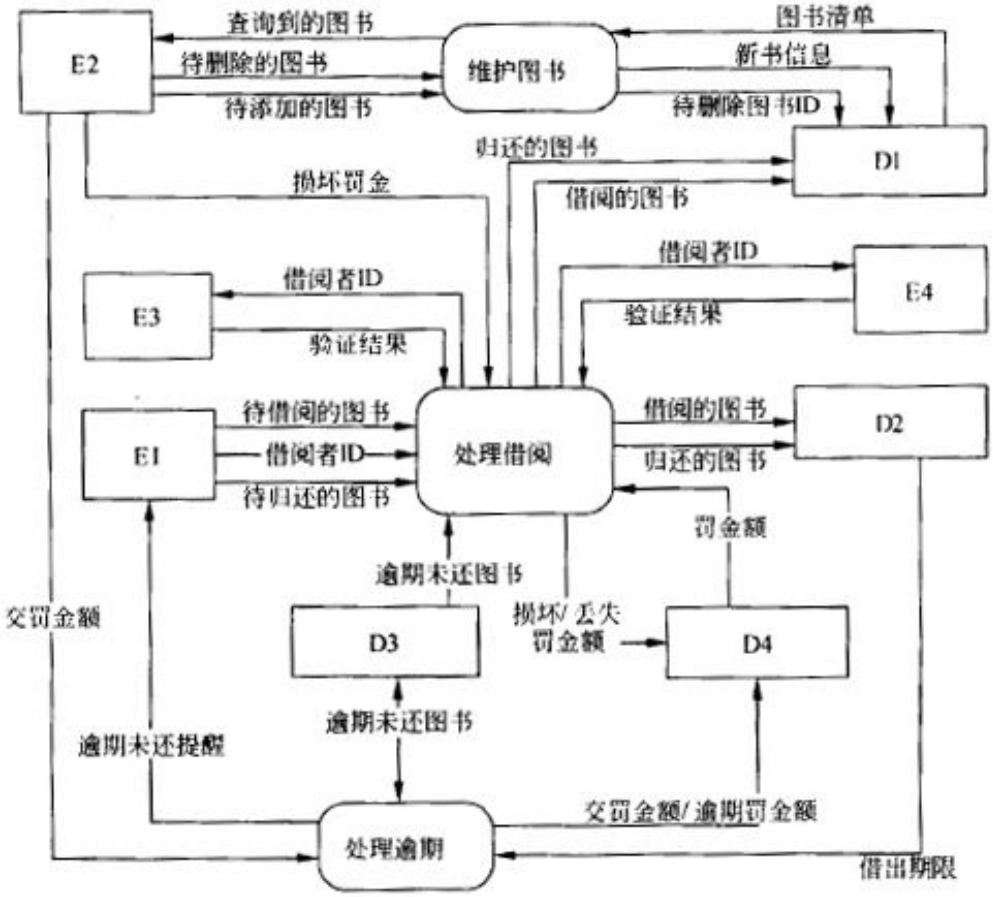


图 1-2　0 层数据流图

问题：1.1

使用说明中的词语，给出图 1-1 中的实体 E1　E4 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　D4 的名称

问题：1.3

在 DFD 建模时，需要对有些复杂加工(处理)进行进-步精化，绘制下层数据流图。针对图 1-2 中的加工“处理借阅”，在 1 层数据流图中应分解为哪些加工？(使用说明中的术语)

问题：1.4

说明【问题 3 】中绘制 1 层数据流图时要注意的问题。

试题二　某医院拟开发一套住院病人信息管理系统，以方便对住院病人、医生、护士和手术等信息进行管理。

【需求分析】

(1）系统登记每个病人的住院信息，包括：病案号、病人的姓名、性别、地址、身份证号、电话号码、入院时间及病床等信息，每个病床有唯一所属的病 K 及病房，如表 2-1 所示。其中病案号唯一标识病人本次住院的信息。

(2）在一个病人的一次住院期间，由一名医生对该病人的病情进行诊断，并填写一份诊断书，如表 2-2 所示。对于需要进行一次或多次手术的病人，系统记录手术名称、手术室、手术日期、手术时间、主刀医生及多名协助医生，每名医生在手术中的责任不同，如表 2-3 所示，其中手术室包含手术室号、楼层、地点和类型等信息

(3）护士分为两类：病床护士和手术室护士。每个病床护士负责护理一个病区内的所有病人，每个病区由多名护士负责护理。手术室护士负责手术室的护理工作。每个手术室护士负责多个手术室，每个手术室由多名护士负责，每个护士在手术室中有不同的责任，并由系统记录其责任。

【概念模型设计】

【概念模型设计】

表 2-1　住院登记表

病案号	071002286	姓名	张三	性别	男
身份证号	0102196701011234	入院时间	2011-03-03	病床号	052401
病房	0524 室	病房类型	三人间	所属病区	05 II 区

表 2-2　诊断书

诊断时间：2011 年 03 月

病案号	071002286	姓名	张三	性别	男	医生	李**
诊断							

表 2-3　手术安排表

手术名称	***手术	病案号	071002286	姓名	张三	性别	男
手术室	032501	手术日期	2011-03-15	手术时间	8:30–10:30	主刀医生	李**
协助医生	王**(协助)，周**(协助)，刘**(协助)，高**(麻醉)						

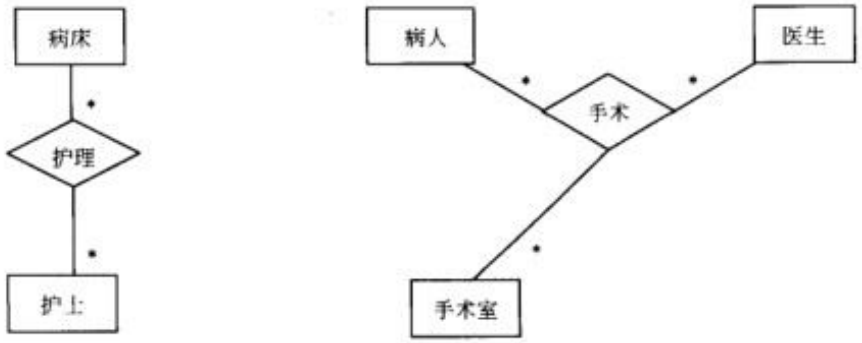


图 2-1　实体联系图

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）：

病床（病床号，病房，病房类型，所属病区）

护士（护士编号，姓名，类型，性别，级别）

病床护士（　　(1)　　）

手术室（手术室号，楼层，地点，类型）

手术室护士（　　(2)　　）

病人（　　(3)　　，姓名，性别，地址，身份证号，电话号码，入院时间）

医生（医生编号，姓名，性别，职称，所属科室）

诊断书（　　(4)　　，诊断，诊断时间）

手术安排（病案号，手术室号，手术时间，手术名称）

手术医生安排（　　(5)　　，医生责任）

问题：2.1

补充图 2-1 中的联系和联系的类型。

问题：2.2

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)～(5)补充完整，并用下划线指出主键。

问题：2.3

如果系统还需要记录医生给病人的用药情况，即记录医生给病人所开处方中药品的名称、用量、价格、药品的生产厂家等信息。请根据该要求，对图 2-1 进行修改，画出补充后的实体、实体间联系和联系的类型。

试题三　某网上购物平台的主要功能如下：

1 创建订单。顾客(Customer)在线创建订单(Order)，主要操作是向订单中添加项目、从订

单中删除项目。订单中应列出所订购的商品(Product)及其数量(quantities)。

2 提交订单。订单通过网络来提交。在提交订单时，顾客需要提供其姓名(name)、 收货地址(address)以及付款方式(formofpayment)(预付卡、信用卡或者现金)。为了制定送货计划以及安排送货车辆，系须确定订单量(volume)。除此之外，还必须记录每种商品的名称(name)、进价(costprice)、售价(saleprice)以及单件商品的包装体积(cubicvolume)。

3 处理订单。订单处理人员接收来自系统的订单；根据订单内容，安排配货，制定送货计划。在送货计划中不仅要指明发货日期(deliverydate)，还要记录每个订单的限时发送要求(DeliveryTimeWindow)。

4 派单。订单处理人员将已配好货的订单转交给派送人员。

5 送货/收货。派送人员将货物送到顾客指定的收货地址。当顾客收货时，需要在运货单(deliveryslip)上签收。签收后的运货单最终需交还给订单处理人员。

6 收货确认。当订单处理人员收到签收过的运货单后，会和顾客进行一次再确认。 现采用面向对象方法开发上述系统，得到如图 3-1 所示的用例图和图 3-2 所示的类图。

问题： 3.1

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　U2 处所对应的用例名称

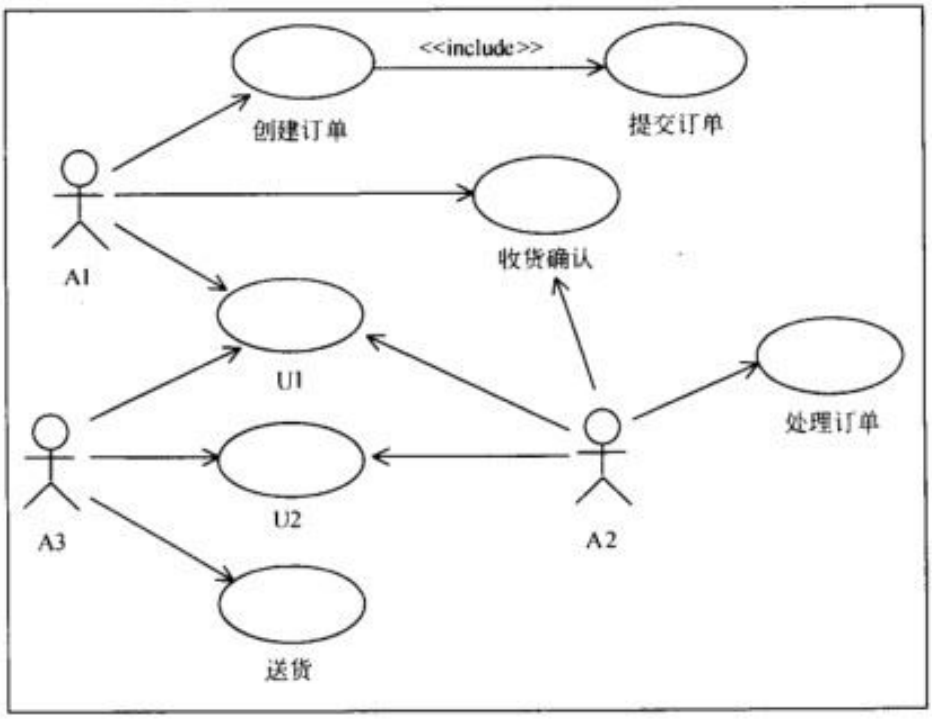


图 3-1　用例图

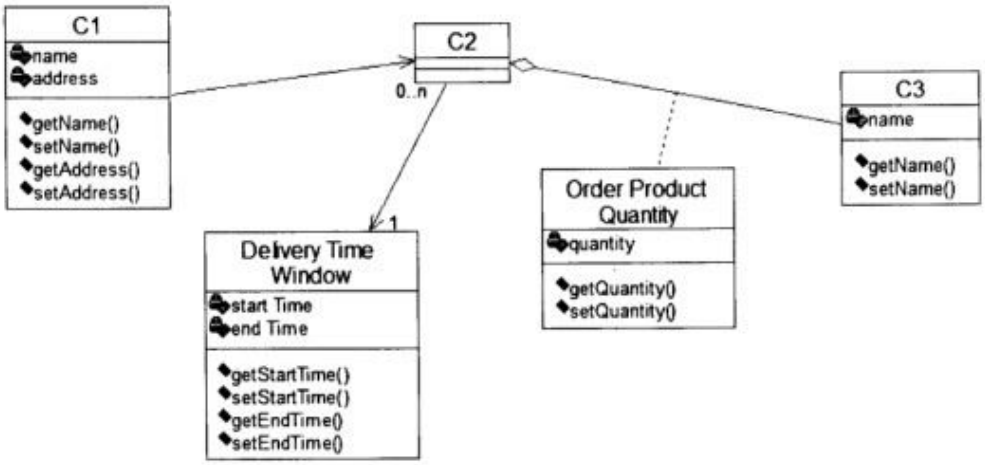


图 3-2　类图

问题：3.2

根据说明中的描述，给出图 3-2 中 C1　C3 所对应的类名以及(1)（ 4)处所对应的多重度(类名使用说明中给出的英文词汇)

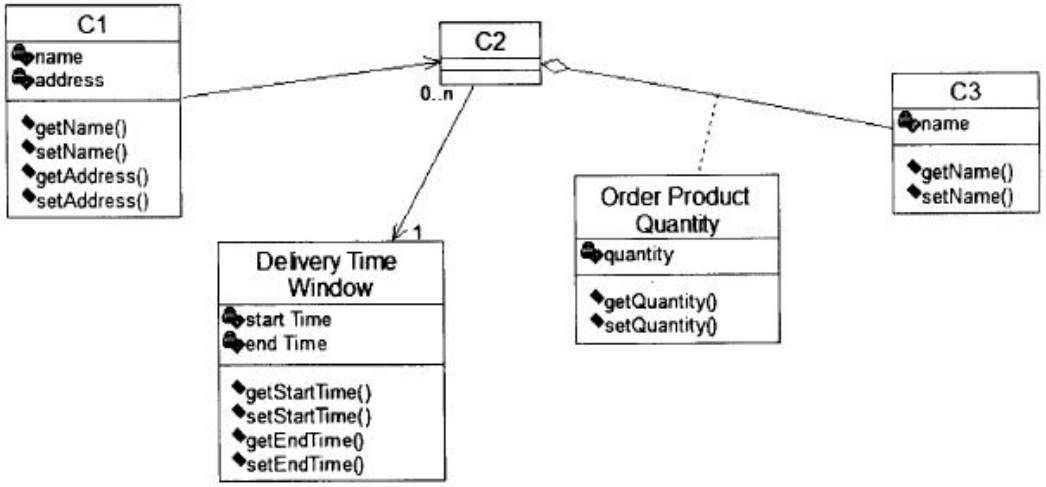


图 3-2 类图

问题： 3.3

根据说明中的描述，将类 C2 和 C3 的属性补充完整(属性名使用说明中给出的英文词汇)。

试题四 用两台处理机 A 和 B 处理 n 个作业。设 A 和 B 处理第 i 个作业的时间分别为叫 ai 和 bi 。由于各个作业的特点和机器性能的关系，对某些作业，在 A 上处理时间长，而对某些作业在 B 上处理时间长。一台处理机在某个时刻只能处理一个作业，而且作业处理是不可中断的，每个作业只能被处理一次。现要找出一个最优调度方案，使得 n 个作业被这两台处理机处理完毕的时间(所有作业被处理的时间之和)最少。

算法步骤：

(1) 确定候选解上界为最短的单台处理机处理所有作业的完成时间 m ，

(2) 用 P(x ， y ， k)=l 表示前 k 个作业可以在 A 用时不超过 x 且在 B 用时不超过 y 时间内处理完成，则 p(x ， y ， k)=p(x-ak ， y ， k-l) || p(x ， y-bk ， k-l) (||表示逻辑或操作)。

(3) 得到最短处理时间为 min(max(x ， y))。

$$m = \min\left(\sum_{i=1}^{n} a_i, \sum_{i=1}^{n} b_i\right)。$$

【C 代码】

下面是该算法的 C 语言实现。

（1）常量和变量说明

n：作业数

m：候选解上界

a：数组，长度为 n，记录 n 个作业在 A 上的处理时间，下标从 0 开始

b：数组，长度为 n，记录 n 个作业在 B 上的处理时间，下标从 0 开始

k：循环变量

p：三维数组，长度为(m+1)* (m+1)* (n+1)

temp：临时变量

max：最短处理时间

(2) C 代码

```
#include <stdio.h>
int n, m;
int a[60], b[60], p[100][100][60];
void read(){  /*输入 n、a、b，求出 m，代码略*/  }
void schedule(){        /*求解过程*/
    int x,y,k;
    for(x = 0; x <= m; x++){
        for (y = 0; y <= m; y++){
            ____(1)____;
            for(k = 1; k <= n; k++)
                p[x][y][k] = 0;
        }
    }
    for(k = 1;k <= n;k++){
        for (x = 0;x <= m;x++){
            for(y = 0; y <= m; y++){
                if(x - a[k - 1] >= 0)  ____(2)____;
                if(____(3)____)  p[x][y][k] = (p[x][y][k] || p[x][y - b[k -
                    1]][k - 1]);
                }
            }
        }
    }
    void write(){        /*确定最优解并输出*/
        int x,y, temp,max = m;
        for(x = 0;x <= m;x++){
            for (y = 0;y <= m;y++){
                if( ____(4)____ ){
                    temp = ____(5)____ ;
                    if(temp < max)     max = temp;
                }
            }
        }
        printf("\n%d\n", max);
    }
    void main(){    read();    schedule();    write();    }
```

问题：4.1

根据以上说明和 C 代码，填充 C 代码中的空(1)（ 5)。

问题：4.2

根据以上 C 代码，算法的时间复杂度为(6)(用 0 符号表示)

问题：4.3

考虑 6 个作业的实例，各个作业在两台处理机上的处理时间如表 4-1 所示。该实例的最优

解为(7),最优解的值(即最短处理时间)为(8)。最优解用(x_1,x_2,x_3,x_4,x_5,x_6) 表示，其中若第 i 个作业在 A 上处理，则 $X_i=1$ ，否则 $X_i=2$ 。如(1， 1， 1， 1， 2,2)表示作业 1， 2, 3 和 4 在 A 上处理，作业 5 和 6 在 B 上处理。

表 4-1

	作业 1	作业 2	作业 3	作业 4	作业 5	作业 6
处理机 A	2	5	7	10	5	2
处理机 B	3	8	4	11	3	4

试题五 某咖啡店售卖咖啡时，可以根据顾客的要求在其中加入各种配料，咖啡店会根据所加入的配料来计算费用。咖啡店所供应的咖啡及配料的种类和价格如下表所示。

咖啡	价格/杯（￥）	配料	价格/份（￥）
蒸馏咖啡（Espresso）	25	摩卡（Mocha）	10
深度烘焙咖啡（DarkRoast）	20	奶泡（Whip）	8

现采用装饰器（Decorator）模式来实现计算费用的功能，得到如图 5-1 所示的类图。

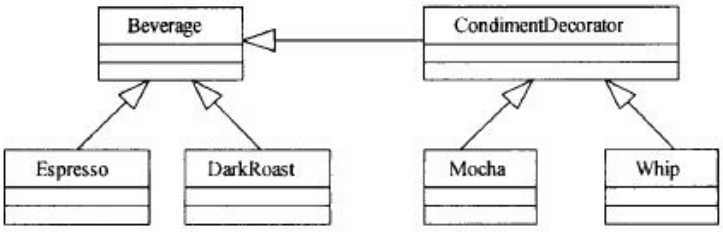


图 5-1 类图

问题： 5.1

【C++代码】

```
#include <iostream>
#include <string>
using namespace std;
const int ESPRESSO_PRICE = 25;
const int DRAKROAST_PRICE = 20;
const int MOCHA_PRICE = 10;
const int WHIP_PRICE = 8;
class Beverage {    // 饮料
  (1)  :    string description;
public:
  (2)  () {  return description;  }
     (3)  ;
};
class CondimentDecorator : public Beverage {    // 配料
protected:
     (4)  ;
};
class Espresso : public Beverage {    // 蒸馏咖啡
public:
    Espresso() { description = "Espresso";  }
    int cost() { return ESPRESSO_PRICE; }
};
class DarkRoast : public Beverage {    // 深度烘焙咖啡
public:
```

```
        DarkRoast() { description = "DardRoast"; }
        int cost() { return DRAKROAST_PRICE; }
    };
    class Mocha : public CondimentDecorator {  // 摩卡
    public:
        Mocha(Beverage* beverage) {  this->beverage = beverage; }
        string getDescription()   {   return beverage->getDescription()+ ",
Mocha";  }
        int cost() { return MOCHA_PRICE + beverage->cost(); }
    };
    class Whip : public CondimentDecorator {  // 奶泡
    public:
        Whip(Beverage* beverage) {  this->beverage = beverage; }
        string getDescription()   {  return  beverage->getDescription()+ ",
Whip";  }
        int cost() { return WHIP_PRICE + beverage->cost(); }
    };

    int main() {
        Beverage* beverage = new DarkRoast();
        beverage = new Mocha( (5) );
        beverage= new Whip( (6) );
        cout << beverage->getDescription() << " ¥" << beverage->cost() << endl;
        return 0;
    }
```

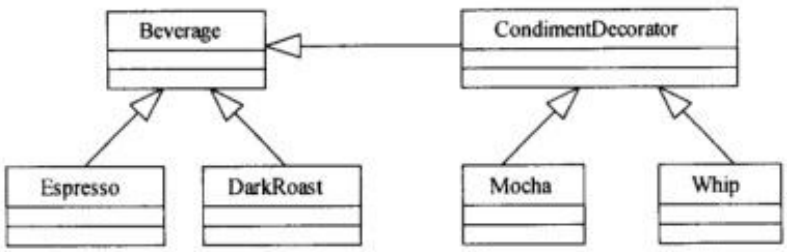
编译运行上述程序，其输出结果为：

```
DarkRoast, Mocha, Whip ¥38
```

试题六　某咖啡店售卖咖啡时，可以根据顾客的要求在其中加入各种配料，咖啡店会根据所加入的配料来计算费用。咖啡店所供应的咖啡及配料的种类和价格如下表所示。

咖啡	价格/杯（¥）	配料	价格/份（¥）
蒸馏咖啡（Espresso）	25	摩卡（Mocha）	10
深度烘焙咖啡（DarkRoast）	20	奶泡（Whip）	8

现采用装饰器（Decorator）模式来实现计算费用的功能，得到如图 6-1 所示的类图。



问题：**6.1**

【Java 代码】

```
import java.util.*;
 (1)  class Beverage {      // 饮料
  String description = "Unknown Beverage";
  public  (2)  () {      return description;  }
  public  (3)  ;
}
abstract class CondimentDecorator extends Beverage {   // 配料
  (4)  ;
}
class Espresso extends Beverage {     // 蒸馏咖啡
  private final int ESPRESSO_PRICE = 25;
  public Espresso() {   description = "Espresso";  }
  public int cost() {    return ESPRESSO_PRICE;  }
}

class DarkRoast extends Beverage {    // 深度烘焙咖啡
  private final int DARKROAST_PRICE = 20;
  public DarkRoast() {  description = "DarkRoast";  }
  public int cost() {    return DARKROAST_PRICE;      }
}
class Mocha extends CondimentDecorator {   // 摩卡
  private final int MOCHA_PRICE = 10;
  public Mocha(Beverage beverage) {
     this.beverage = beverage;
  }
  public String getDescription() {
     return beverage.getDescription() + ", Mocha";
  }
```

```
  public int cost() {
     return MOCHA_PRICE + beverage.cost();
  }
}
class Whip extends CondimentDecorator {  // 奶泡
  private final int WHIP_PRICE = 8;
  public Whip(Beverage beverage) {  this.beverage = beverage; }
  public String getDescription()  {
      return beverage.getDescription()+ ", Whip";
  }
  public int cost() { return WHIP_PRICE + beverage.cost(); }
}

public class Coffee {
  public static void main(String args[]) {
    Beverage beverage = new DarkRoast();
    beverage = new Mocha( __(5)__ );
    beverage = new Whip( __(6)__ );
    System.out.println(beverage.getDescription() + " ¥" +
    beverage.cost());
  }
}
```

编译运行上述程序，其输出结果为：

```
DarkRoast, Mocha, Whip ¥38
```

试题一 答案：　解析：　E1：借阅者 E2:图书管理员 E3/E4:学生数据库/职工数据库

注：E3 和 E4 不分顺序，但必须不同。

本题考查顶层 DFD。顶层 DFD一般用来确定系统边界，将待开发系统看作一个加工，图中只有唯一的一个处理和一些外部实体，以及这两者之间的输入输出数据流。题目要求根据描述确定图中的外部实体。分析题目中描述，并结合已经在顶层数据流图中给出的数据流进行分析。从题目的说明中可以看出：和系统的交互者包括图书管理员、借阅者两类人，图书管理员需要维护图书信息、得到查询所得的图书信息，借阅者提供借阅者 ID、借阅与归还的图书。还有通过与教务处维护的学生数据库、人事处维护的职工数据库中的数据进行比对以验证借阅者 ID 是否合法的两个数据库作为外部实体。

对应图 1-1 中数据流和实体的对应关系，可知 E1 为借阅者，E2 为图书管理员，E3 和 E4 为学生数据库和职工数据库。

D1 ： 图书表 D2:借出图书表 D3:逾期未还图书表 D4:罚金表

本题考查 0 层 DFD 中数据存储的确定。说明中描述维护图书信息主要存储或者更新图书表；借阅时需要检查逾期未还图书表，确定是否有逾期未还图书以及罚金表中的罚金限额，归还时出现缺失和损坏需要处以罚金并存入罚金表；借阅与归还图书时需要存入借出图书表和更新借出图书表。在处理逾期时需要将罚金记入罚金表，要检查和更新罚金限额。根据描述和图 1-2 中的数据存储的输入输出数据流提示，可知 D1 为图书表， D2 为借出图书表， D3 为逾期未还图书表， D4 为罚金表。

检查借阅者身份或检查借阅者 ID ；检查逾期未还图书；检查罚金是否超过限额；借阅图书；归还图书。

本题考查将 0 层 DFD 中的处理进一步精化建模，绘制下层数据流图。从说明中对“处理借阅”的描述和图 1-2 可知，处理借阅需要检查借阅者身份、检查逾期未还图书、检查罚金是否超过限额、借阅图书和归还图书。描述中：检查所还图书是否逾期，若是，执行“处理逾期”操作。这里处理逾期明确说明是一个操作，而且在描述(3) 中单独描述，图 1-2 中已经建模为单独一个处理，所以在本问题中仍然不分解。

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。如果父图的一个输入(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个数据流，那么它们仍然算是平衡的。

本题考查在绘制下层数据流图时需要注意的问题。问题 3 明确给出是对复杂处理进行进一步精化，绘制下层数据流图，因此需要注意的问题是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。如果父图的一个输入(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个数据流，那么它们仍然算是平衡的。

试题二 答案： 解析：

本题考查数据库的概念结构设计，题目要求补充完整实体联系图中的联系和联系的类型。

根据题目的需求描述可知，一名病人在一次住院期间对应一张病床，而一个病床可以有多名病人曾经住过。所以，病床实体和病人实体之间存在“住院”联系，联系的类 型为多对一，表示为*:1 。

根据题目的需求描述可知，一名病人在一次住院期间，由一名医生做出诊断，并给出一份诊断书。所以，病人实体和医生实体之间存在“诊断”联系，联系的类型为多对多，表示

为*:*:*。

根据题目的需求描述可知，一名病人在一次住院期间可以进行多次手术，一次手术安排在一个手术室，由多名医生参与。所以，病人实体与医生实体和手术室实体三者之间’存在“手术”联系，三者之间联系的类型为多对多对多，表示为根据题目的需求描述可知，一名手术室护士负责多个手术室，每个手术室由多名护士负责。所以，护士实体和手术室实体之间存在“负责”联系，联系的类型为多对多，表示为*:*。

1：病区，护士号。2：手术室号，护士号，责任。3：病案号，病床号。4：病案号，医生编号。5：病案号，手术室号，手术时间，医生编号。

本题考查数据库的逻辑结构设计，题目要求补充完整各关系模式，并给出各关系模式的主键。

根据实体联系图和需求描述，每个病床护士负责护理一个病区内的所有病人，每个病区由多名护士负责护理。系统记录每个病床护士所负责护理的病区。所以，对于“病 床护士”关系模式需填写的属性为：病区，护士号。

根据实体联系图和需求描述，每个手术室护士负责多个手术室，每个手术室由多名 护士负责，每个护士在手术室中有不同的责任。因此，对于“手术室护士”关系模式，需填写的属性为：手术室号，护士号，责任。

根据实体联系图和需求描述，病案号唯一标识病人本次住院的信息。病人的住院信息包括病床信息。所以，对于“病人”关系模式需补充的属性为：病案号，病床号。

根据实体联系图和需求描述，一名病人在一次住院期间，由一名医生做出诊断，并给出一份诊断书。所以，对于“诊断”关系模式需补充的属性为：病案号，医生编号。

根据实体联系图和需求描述，一名病人在一次住院期间，可能需要进行一次或多次手术，每次手术安排在一间手术室，由多名医生(包括主刀医生)参与。所以，对于“手术医生安排”关系模式需补充的属性为：病案号，手术室号，手术时间，医生编号。

病床护士关系模式的主键：病区，护士号手术室护士关系模式的主键：手术室号，护士号

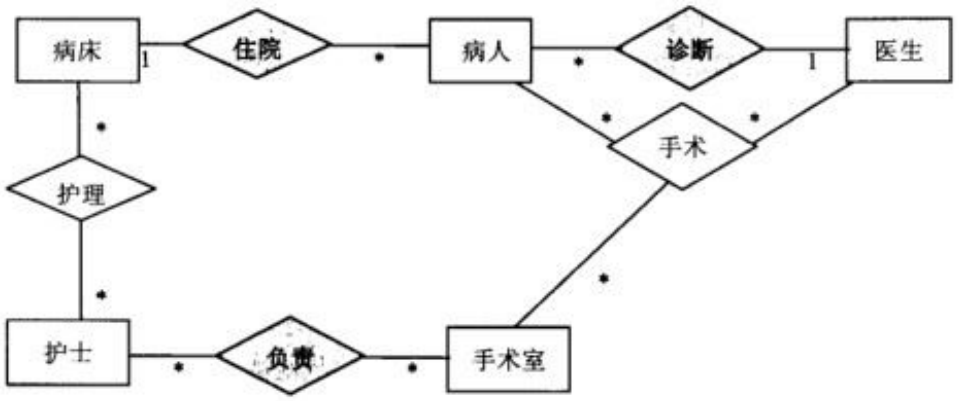
病人关系模式的主键：病案号

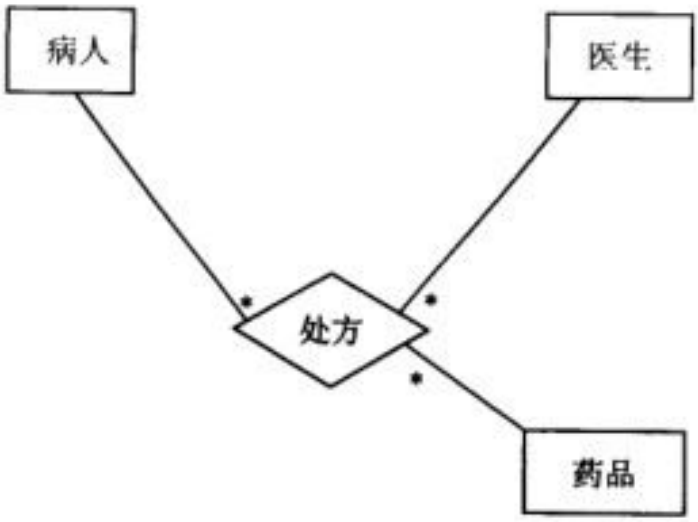
诊断书关系模式的主键：病案号采购订单关系模式的主键：订单编码

手术医生安排关系模式的主键：病案号，手术室号，手术时间，医生编号码

本题考查数据库的概念结构设计，根据新增的需求新增实体联系图中的实体及联系和联系的类型。

根据问题描述，系统需记录医生给病人开处方的药品信息，则需新增“药品”实体，并在病人实体与医生实体和药品实体三者之间存在“处方”联系，联系的类型是多对多对多(*:*:*)。





试题三 答案：　解析：　A1：顾客 A2:订单处理人员 A3:派送人员 U1：收货 U2:派单

本题要求将图 3-1 所给出的用例图补充完整。用例图的构成要素有参与者(Actor)、用例(UseCase)以及用例之间的关系。由图 3-1 可知，题目中“网上购物平台”的用例模型中共有 3 个参与者、 7 个用例。图中给出了其中的 5 个用例，需要补充所缺少的两个用例和三个参与者。解答此题时，应首先确定参与者，然后再找到与每个参与者对应的用例。

参与者表示要与本系统发生交互的一个角色单元。与系统交互的外部人员、角色、其他的计算机系统、物理实体等通常都可以看作是参与者。从“说明”中可以看出，在本题的描述中出现了 “顾客”、“订单处理人员”、“派送人员”这三种角色。而用例图中恰好缺少了三个参与者，所以可以确定这三种角色就应该与 A1　A3 对应。接下来要确定的就是它们之间的对应关系，这就需要明确参与者与用例之间的关系了。解答时可以依据用例图逐个考查其中的参与者。

先从 A1 开始。 A1 参与的用例分别为“创建订单”、“收货确认”以及需要补充的用例 U1 。用例总是由参与者启动的。以“创建订单”、“收货确认”为关键词，在“说明”中查找出现这两个关键词的语句的主语。这个主语就是该用例的参与者。由“说明”可见， A1 应该表示的是“顾客”。采用同样的方法，可以确定 A2 表示“订单处理人员”，A3 表示“派送人员”。

由于缺少的用例 U1 和 U2 与参与者 A1 和 A3 相关，因此重点考查“说明”中出现了 “顾客”和“派送人员”的语句。比对“说明”和用例图可知，功能“派单”、“收货”没有出现在用例图中。“订单处理人员”和“派送人员”都参与了“派单”功能，所以 U2 处应该是用例“派单”。而“收货”功能涉及了三种角色：派送人员、顾客(两者完成货物的交接)、订单处理人员(收回顾客签字后的运货单，此时一次完整的收货活动才算结束)，所以 U1 处应该是“收货”。

Cl ： CustomerC2 ： OrderC3 ： Product

本题考查的是类图建模。解题的重点在于根据类图中提供的类、类的属性以及类之 间的关联关系推断出需要补充的类。

先看类 Cl ， C1 的关键属性都已经给出了。由“说明”可知，属性 address 表示的是“收货地址”。而收货地址的最初始来源是顾客，所以类 C1 代表的应该是 Customer(顾客)。

另外，从“说明”中可知，“订单”和“商品”是“网上购物平台”系统的关键概念，所以应该在类图上有相应的类来表示。由此可以推断出， C2 和 C3 就应该与这两个概念相关。观察类图，在 C2 和 C3 之间存在着一个关联类 OrderProduct,而 C3 和 C2 之间又存在着一个聚集关系， C3 是构成 C2 的部分对象。根据“说明”，订单中包含了订购的商品，所以 C2 应该是 Order （订单)， C3 应该是 Product （商品)。

三个类都确定之后，下面来识别它们之间的多重度。

首先顾客(C1)和订单(C2)之间的关联，一名顾客可以在线创建多个订单，但是一个订单只能由一名顾客来创建。所以(1) 处应填写“1”，(2) 处应填写“0..n”。

订单(C2)和商品(C3)，一个订单上可以订购多件商品，而一件商品可以出现在一个或多个订单中，也可能没有任何顾客订购某种商品。因此 C2 和 C3 之间是一种多对多联系。在图 3-2 中，采用了关联类的方法来表示多对多联系。这里(3) 处应填写“0..n”，

(4) 处应填写“l..n”。这里下限 1 表示一个订单中至少应包含一件商品。

C2 ： volume 、 deliverydate 、 formofpayment

C3: cubicvolume 、 costprice 、 saleprice

本题考查类的关键属性的识别。由“说明”中给出的描述可知，类 C2 的属性至少应包括 volume 、 deliverydate 、 formofpayment ；类 C3 的属性至少应包括 cubicvolume 、costprice 、 saleprice 。

（1）1　　（2）0..n 或 0..*　（3）0..n 或 0..*　　（4） 1..n 或 1..*

试题四 答案：　解析：

schedule 函数计算 p 数组中元素的值。第一个三重 for 循环进行 p 数组的初始化，当 k = 0 时， p[i][j][k] = 1;而当 k 关 0 时， p[i][j][k] = 0,因此(1) 处应填 “p[x][y][0] = 1 ”。

在接下来的三重 for 循环中，自底向上计算 p[i][j]M 的值，计算根据题中定义来进行，(2) 处应填 “p[x][y][k]=p[x-a[k-l]][y][k-l]”。对应于上一行的条件，(3) 处应填“y - b[k—1]>=0”。

write 函数中，已经计算出 p 数组元素的值，因此此时每一个 p[i][j][n] = 1 表示一个可行解，因次(4) 处应填“p[x][y][n] = l”或等价的表达式。其中该解对应的处理时间为 i 和 j 中的较大值，因此(5) 处应填“(x>=y)?x:y”。在所有的可行解中，确定最小的处理时间即为问题的最优解。

根据上述 C 代码，函数 schedule 有两处三重 for 循环，时间复杂度为 $O(m^2n)$。函数 write 有一处两重 for 循环，时间复杂度为 $O(m^2)$。因此整个算法的时间复杂度为 $O(m^2n + m^2)$, 由于 m^2 在数量级上小于 m^2n, 因此算法的时间复杂度可以表示为 $O(m^2n)$。

(7) (1,1,2,2,1,1)

(8) 15

根据题意和算法，可以得到若作业 1 、 2 、 5 和 6 在处理机 A ，而作业 3 和 4 在处理 机 B 上处理，可以得到最优解。此时在处理机 A 上的处理时间为 14, 而在处理机 B 上 的处理时间为 15, 因此最优解的值，即最短的处理时间为 15, 而最优解为(1,1,2,2,1,1)。

（1）p[x][y][0] = 1

（2）p[x][y][k] = p[x – a[k – 1]][y][k – 1]

（3）y – b[k – 1] >= 0

（4）p[x][y][n] == 1 或 p[x][y][n] 或 p[x][y][n] != 0

（5）(x >= y)? x:y　　$O(m^2n)$

试题五 答案：　解析：　1.protected

2.virtual string getDescription

3.virtualintcost() = 0

4.Beverage* beverage

5.beverage

本题考查装饰器(Decorator)模式的概念及应用。

Decorator 模式动态地给一个对象添加一些额外的职责，就增加功能来说，装饰模式提供了比继承更有弹性的替代方案。

Decorator 模式的优点是有效避免了使用继承方式扩展对象功能而带来的灵活性差、子类无限制扩展的问题；装饰者与被装饰者之间虽然都是同一类型，但是它们彼此是完全独立并可以独立任意改变的。

Decorator 模式的适用场合是：想透明并且动态地给对象增加新的职责；给对象增加的职责，在未来存在增加或减少的可能。

Decorator 模式的类图如下所示：

题目利用 Decorator 模式来计算各种配料组合的咖啡的价格。 Beverage 相当于抽象的 Component 类，最终要计算出 Beverage 的价钱。 Espresso 和 DarkRoast 是 4 个具体的组件，代表一种咖啡类型。 Macha 和 Whip 是配料装饰者，可以添加到不同类型的咖啡中。 CondimentDecorator 相当于 Decorator,是装饰者共同实现的接口。在本题中，确定装饰者共同实现的接口是什么，是一个重要的考查点。

下面来分析程序。

第(1) 空要求给出类 Beverage 的数据成员 description 的访问控制。 C++语言提供了 3 种访问控制 private 、 public 和 protected 。 private 成员只能在定义它的类中或其友元访问； public 成员可以被任意访问； protected 成员只能在定义它的类及其子类中访问。从程序上下文可知，在 Beverage 的子类 Espresso 和 DarkRoast 中都需要访问 description,所以第(1) 空应填“protected”。

第(2) 、(3) 空要求确定 Beverage 中提供的公共接口。解答时应全面阅读程序，通过子类的代码来推断父类所提供的接 U 是什么。

首先来观察 Beverage 的两个子类。在 Espresso 和 DarkRoat 中都出现的成员函数是 cost,其功能是给出咖啡的价格；而在这两个类中， cost 的实现代码又不相同。这就提示我们这里采用了 C++语言的动态多态机制，意味着需要在这两个类的父类中定义一个虚拟函数，其函数原型就应该是 intcost0 。现在回到第(2) 、(3) 空。第(2) 空给出了成员函数的实现体，说明(2) 处的成员函数是与 description 相关的。也就是说，第(2) 空处的成员函数不可能是 cost 。这样就可以确定第(3) 空应该是 cost 成员函数。仔细观察第(3) 空，这条语句的结束符是“；”，即第(3) 空中只能出现函数的接口声明。只有接口声明的虚

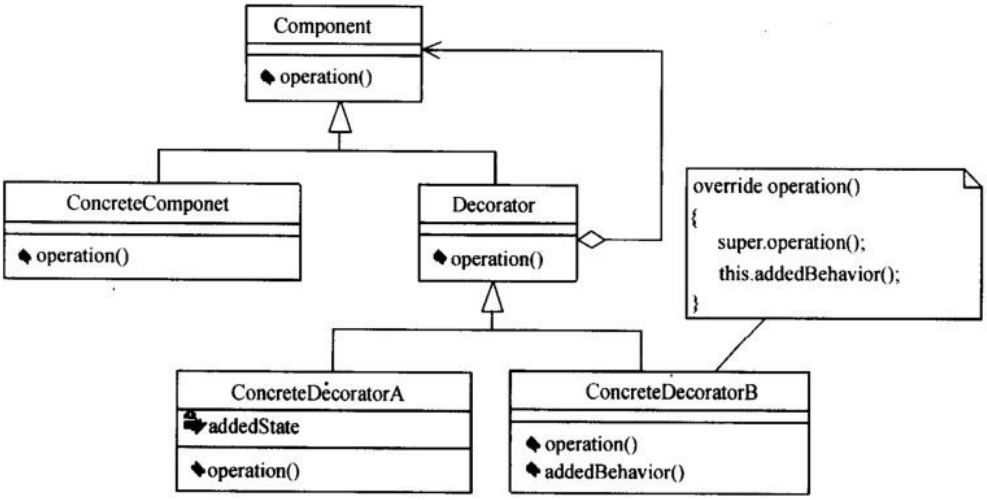
函数，在 C++中只能是纯虚拟函数。所以第(3) 空应填写“virtualintcost() = 0”。

如何来确定第(2) 空？在类 Espresso 和 DarkRoast 中已经找不到相关信息了，我们考查剩余的类。

类 CondimentDecorator 是 Beverage 的子类，那么第(2) 和(3) 空处的成员函数都会被它继承。而类 Mocha 和 Whip 又是 CondimentDecorator 的子类，第(2) 和(3) 空处的成员函数同样也会被这两个类继承。观察 Mocha 和 Whip 的代码，可以发现在这两个子类中也重置了纯虚拟函数 cost 。除此之外还可以发现，在这两个子类中都出现了成员函数 string getDescription() ，并且其在两个类中的实现代码也不相同。这说明， getDescription 也应该是父类中所定义的虚拟函数。从子类沿着继承路径追袖到父类，可以确定第(2) 空就应该是虚拟函数 getDescription 最初的定义之处。所以第(2) 空应填写 “virtual stringgetDescription”。

第(4)空考查的是类 CondimentDecorator 中定义的数据成员，要求确定其类型和名称。由于声明成了 protected, 因此在其子类 Mocha 和 Whip 中可以找到答案。在这两个类中都出现了 “this->beverage”表达式，说明 beverage 应该是该类的数据成员。但在这两个类中都没有定义数据成员，很显然，这个数据成员应该是从父类继承而来的。所以第(4)空应填 “Beverage* beverage”。

第(5) 和(6) 空考查的是 Decorator 模式的使用方法。从主函数可知，其基本过程是现制作一杯烘焙咖啡(DarkRoast), 然后再向这杯咖啡中加入 1 份 Mocha 和 1 份 Whip, 最后这杯咖啡的价格应该是烘焙咖啡的价格+1 份 Mocha 的价格+1 份 Whip 的价格。(5) 和(6) 处都应该填写“beverage”。



试题六 答案： **解析：** 1.abstract.2.String getDescription.3.abstract int cost().4.Beverage beverage.5.beverage.6.beverage

本题考查装饰器(Decorator)模式的概念及应用。

Decorator 模式动态地给一个对象添加一些额外的职责，就增加功能来说，装饰模式提供了比继承更有弹性的替代方案。

Decorator 模式的优点是有效避免了使用继承方式扩展对象功能而带来的灵活性差、子类无限制扩展的问题；装饰者与被装饰者之间虽然都是同一类型，但是它们彼此是完全独立并可以独立任意改变的。

Decorator 模式的适用场合是：想透明并且动态地给对象增加新的职责；给对象增加的职责，在未来存在增加或减少的可能。

Decorator 模式的类图如下所示：

题目利用 Decorator 模式来计算各种配料组合的咖啡的价格。 Beverage 相当于抽象的 Component 类，最终要计算出 Beverage 的价钱。 Espresso 和 DarkRoast 是 4 个具体的组件，代 k 一种咖啡类型。 Macha 和 Whip 是配料装饰者，可以添加到不同类型的咖啡中。 CondimentDecorator 相当于 Decorator, 是装饰者共同实现的接口。在本题中，确定装饰者共同实现的接口是什么，是一个重要的考查点。

下面来分析程序。

Decorator 模式中的 Component 通常都用抽象类来实现。所以第(1) 空应填写 “abstract”。

第(2) 、(3) 空要求确定 Beverage 中提供的公共接口。解答时应全面阅读程序，通过子类的代码来推断父类所提供的接口是什么。

首先来观察 Beverage 的两个子类。在 Espresso 和 DarkRoat 中都出现的成员函数是 cost, 其功能是给出咖啡的价格；而在这两个类中， cost 的实现代码又不相同。这意味着需要在这两个类的父类中定义一个抽象函数，其函数原型就应该是 intcost() 。现在回到第(2) 、(3) 空。第(2) 空给出了成员函数的实现体，说明(2) 处的成员函数是与 description 相关的。也就是说，第(2) 空处的成员函数不可能是 cost 。这样就可以确定第(3) 空应该是 cost 成员函数。所以第(3) 空应填写“abstractintcost() ’ ’ 。

如何来确定第(2) 空？在类 Espresso 和 DarkRoast 中已经找不到相关信息了，我们考查剩余的类。

类 CondimentDecorator 是 Beverage 的子类，那么第(2) 和(3) 空处的成员函数都会被它继承。而类 Mocha 和 Whip 又是 CondimentDecorator 的子类，第(2) 和(3) 空处的成员函数同样也会被这两个类继承。观察 Mocha 和 Whip 的代码，可以发现在这两个子类中也重置了抽象函数 cost 。除此之外还可以发现，在这两个子类中都出现了成员函数 String getDescription0 。从子类沿着继承路径追溯到父类，可以确定第(2) 空就应 该是成员函

数 getDescription 最初的定义之处。所以第(2) 空应填写“ StringgetDescriptio n’’ 。