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 2011 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 某医院欲开发病人监控系统。该系统通过各种设备监控病人的生命特征，并在生命特征异常时向医生和护理人员报替。该系统的主要功能如下：

(1）本地监控：定期获取病人的生命特征，如体温、血压、心率等数据。

(2）格式化生命特征：对病人的各项重要生命特征数据进行格式化，然后存入曰志文件并检查生命特征。

(3）检查生命特征：将格式化后的生命特征与生命特征范围文件中预设的正常范围进行比较。如果超出了预设范围，系统就发送一条警告信息给医生和护理人员。

(4）维护生命特征范围：医生在必要时(如，新的研究结果出现时)添加或更新生命特征值的正常范围。

(5）提取报告：在医生或护理人员请求病人生命特征报告时，从日志文件中获取病人生命特征生成特征报告，并返回给请求者。

(6）生成病历：根据日志文件中的生命特征，医生对病人的病情进行描述，形成病历存入病历文件。

(7）查询病历：根据医生的病历查询请求，查询病历文件，给医生返回病历报告。

(8）生成治疗意见：根据日志文件中的生命特征和病历，医生给出治疗意见，如处方等，并存入治疗意见文件。

(9）查询治疗意见：医生和护理人员查询治疗意见，据此对病人进行治疗。

现采用结构化方法对病人监控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。

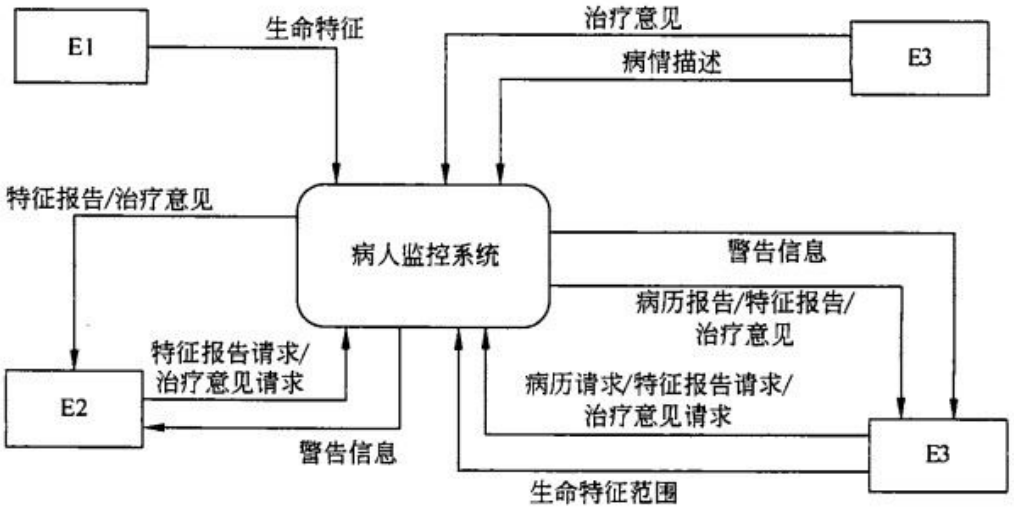


图 1-1　顶层数据流图

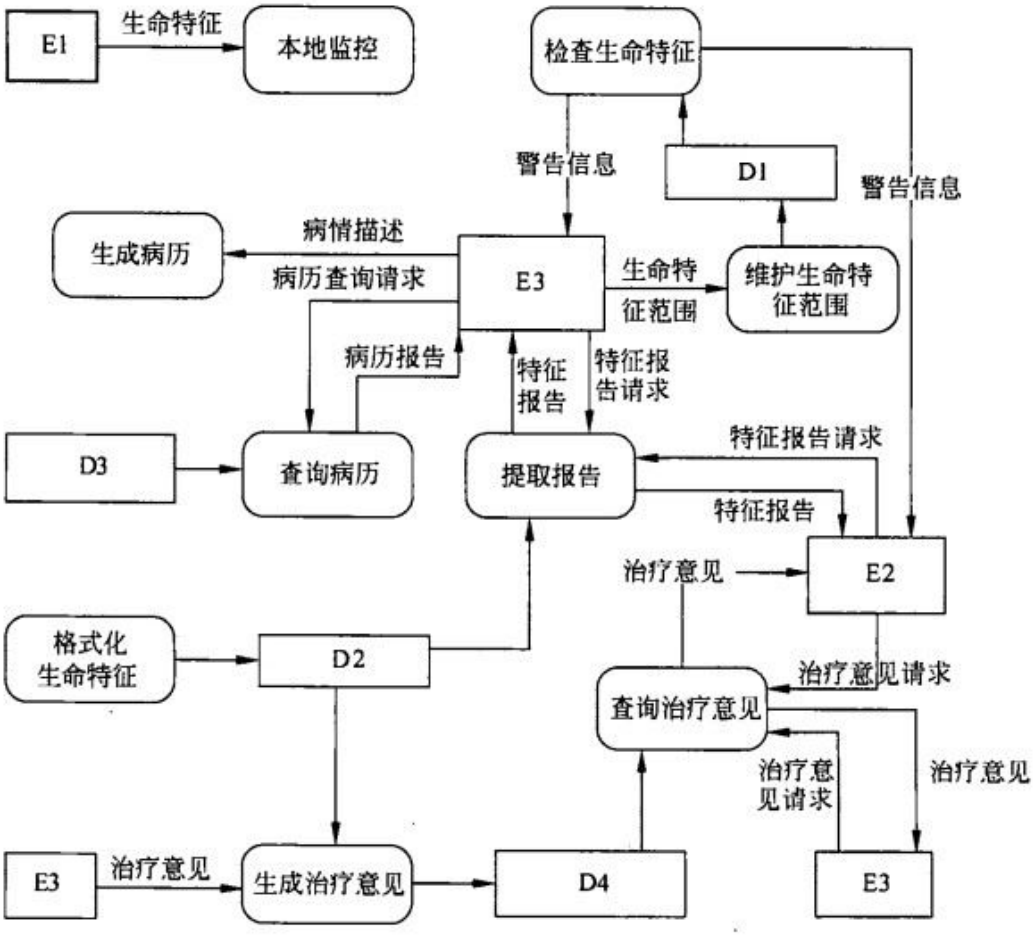


图 1-2　0 层数据流图

问题：1.1

使用说明中的词语，给出图 1-1 中的实体 E1　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　D4 的名称。

问题：1.3
图 1-2 中缺失了 4 条数据流，使用说明、图 1-1 和图 1-2 中的术语，给出数据流的名称及其起点和终点。
问题：1.4
说明实体 E1 和 E3 之间可否有数据流，并解释其原因。

试题二 某服装销售公司拟开发一套服装采购管理系统，以方便对服装采购和库存进行管理。

【需求分析】

(1）采购系统需要维护服装信息及服装在仓库中的存放情况。服装信息主要包括： 服装编码、服装描述、服装类型、销售价格、尺码和面料，其中，服装类型为销售分类，服装按销售分类编码。仓库信息包括：仓库编码、仓库位置、仓库容量和库管员。系统记录库管员的库管员编码、姓名和级别。一个库管员可以管理多个仓库，每个仓库有一名库管员。一个仓库中可以存放多类服装，一类服装可能存放在多个仓库中。

(2）当库管员发现有一类或者多类服装缺货时，需要生成采购订单。一个采购订单可以包含多类服装。每类服装可由多个不同的供应商供应，但具有相同的服装编码。采购订单主要记录订单编码、订货日期和应到货日期，并详细记录所采购的每类服装的数量、采购价格和对应的多个供应商。

(3）系统需记录每类服装的各个供应商信息和供应商生产服装的情况。供应商信息包括：供应商编码、供应商名称、地址、企业法人和联系电话。一个供应商可以供应多类服装，一类服装可由多个供应商供应。库管员根据入库时的服装质量情况，设定每个供应商所供应的每类服装的服装质量等级，作为后续采购服装时，选择供应商的参考标准。 .

【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)：

库管员(库管员编码，姓名，级别)

仓库信息((1)，仓库位置，仓库容量)

服装(服装编码，服装描述，服装类型，尺码，面料，销售价格)

供应商(供应商编码，供应商名称，地址，联系电话，企业法人)

供应情况((2)，服装质量等级)

采购订单((3) 采购订单明细(4))

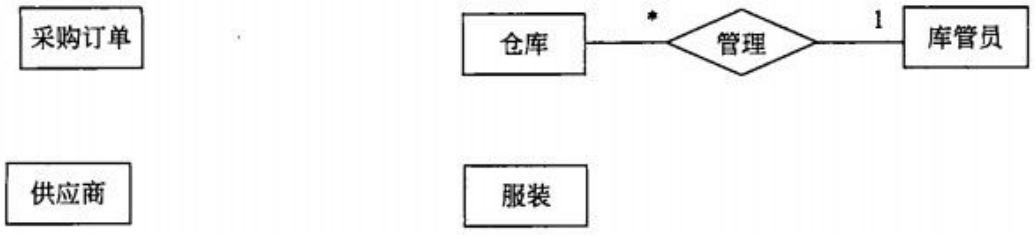


图 2-1 实体联系图

问题： 2.1
根据需求分析的描述，补充图 2-1 中的联系和联系的类型。
问题： 2.2
根据补充完整的图 2-1，将逻辑结构设计阶段生成的关系模式中的空(1)（ 4）补充完整，并给出其主键(用下划线指出)。
问题： 2.3
如果库管员定期需要轮流对所有仓库中的服装质量进行抽查，对每个仓库中的每一类被抽查服装需要记录一条检查结果，并且需要记录抽查的时间和负责抽查的库管员。 请根据该要求，对图 2-1 进行修改，画出修改后的实体间联系和联系的类型。

试题三 一个简单的图形编辑器提供给用户的基本操作包括：创建图形、创建元素、选择元素以及删除图形。图形编辑器的组成及其基本功能描述如下：

(1) 图形由文本元素和图元元素构成，图元元素包括线条、矩形和椭圆。

(2) 图形显示在工作空间中，一次只能显示一张图形(即当前图形， current)。

(3）编辑器提供了两种操作图形的工具：选择工具和创建工具。对图形进行操作时，一次只能使用一种工具(即当前活动工具， active)。

①创建工具用于创建文本元素和图元元素。

②对于显示在工作空间中的图形，使用选择工具能够选定其中所包含的元素，可以选择一个元素，也可以同时选择多个元素。被选择的元素称为当前选中元素(selected).

③每种元素都具有对应的控制点。拖拽选定元素的控制点，可以移动元素或者调整元素的大小。

现采用面向对象方法开发该图形编辑器，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。

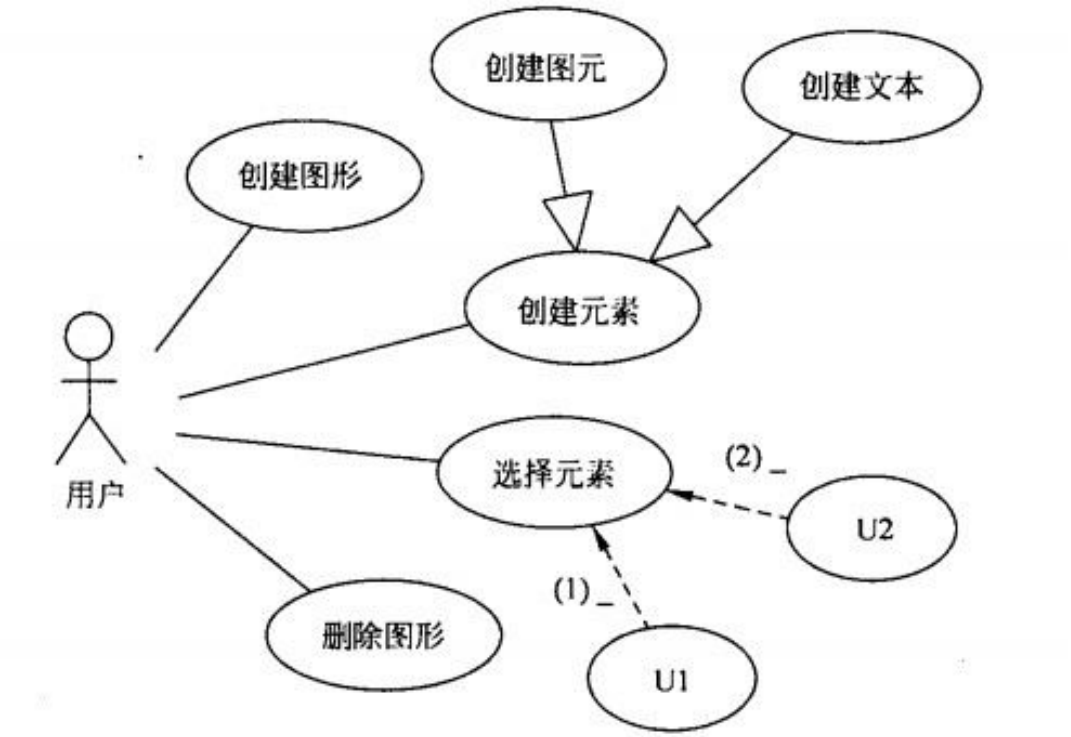


图 3-1　用例图

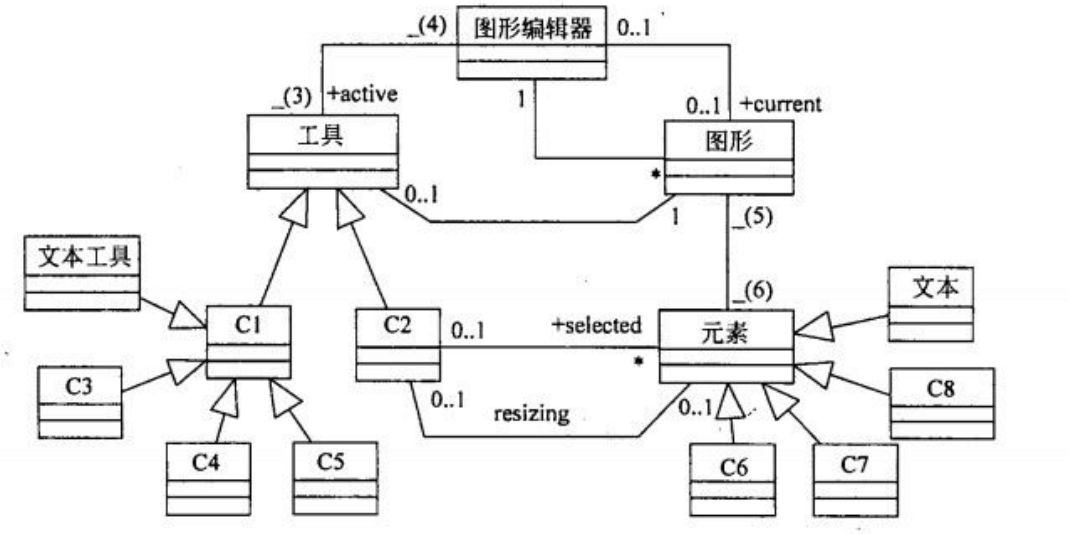


图 3-2　类图

问题：3.1

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)和(2)处所对应的关系。

问题：3.2

根据说明中的描述，给出图 3-2 中缺少的 C1 至 C8 所对应的类名以及(3)至(6) 处所对应的多重度。

问题：3.3

图 3-2 中的类图设计采用了桥接(Bridge)设计模式，请说明该模式的内涵。

试题四 某应用中需要对 100000 个整数元素进行排序，每个元素的取值在 0 5 之间。排序算法的基本思想是：对每一个元素 x，确定小于等于 x 的元素个数(记为 m)，将 x 放在输出元素序列的第 m 个位置。对于元素值重复的情况，依次放入第 m-1 、 m-2 、…个位置。例如，如果元素值小于等于 4 的元素个数有 10 个，其中元素值等于 4 的元素个数有 3 个，则 4 应该在输出元素序列的第 10 个位置、第 9 个位置和第 8 个位置上。

算法具体的步骤为：

步骤 1:统计每个元素值的个数。

步骤 2:统计小于等于每个元素值的个数。

步骤 3:将输入元素序列中的每个元素放入有序的输出元素序列。

【C 代码】

下面是该排序算法的 C 语言实现。

(1) 常量和变量说明

R：常量，定义元素取值范围中的取值个数，如上述应用中 R 值应取 6 。

i：循环变量。

n：待排序元素个数。

a：输入数组，长度为 n 。

b：输出数组，长度为 m

c：辅助数组，长度为 R, 其中每个元素表示小于等于下标所对应的元素值的个数。

(2) 函数 sort

```
1 void sort(int n, int a[], int b[]) {
2    int c[R], i;
3    for(i = 0; i < ___(1)___; i++) {
4       c[i] = 0;
5    }
6    for(i = 0; i < n; i++) {
7       c[a[i]] = ___(2)___;
8    }
9    for(i = 1; i < R; i++) {
10      c[i] = ___(3)___;
11   }
12   for(i = 0; i < n; i++) {
13      b[c[a[i]] - 1] = ___(4)___;
14      c[a[i]] = c[a[i]] - 1;
15   }
16 }
```

问题：4.1

根据说明和 C 代码，填充 C 代码中的空缺(1)（ 4)。

问题：4.2

根据 C 代码，函数的时间复杂度和空间复杂度分别为(5)和(6)(用 O 符号表示)。

问题：4.3

根据以上 C 代码，分析该排序算法是否稳定。若稳定，请简要说明(不超过 100 字)； 若不稳定，请修改其中代码使其稳定(给出要修改的行号和修改后的代码)。

试题五　某饭店在不同的时段提供多种不同的餐饮，其菜单的结构图如图 5-1 所示。

现在采用组合(Composition)模式来构造该饭店的菜单，使得饭店可以方便地在其中增加新的餐饮形式，得到如图 5-2 所示的类图。其中 MermCompcment 为抽象类，定义了添加(add)新菜单和打印饭店所有菜单信息(print)的方法接口。类 Menu 表示饭店提供的每种餐饮形式的菜单，如煎饼屋菜单、咖啡屋菜单等。每种菜单中都可以添加子菜单，例如图 5-1 中的甜点菜单。类 Menultem 表示菜单中的菜式。

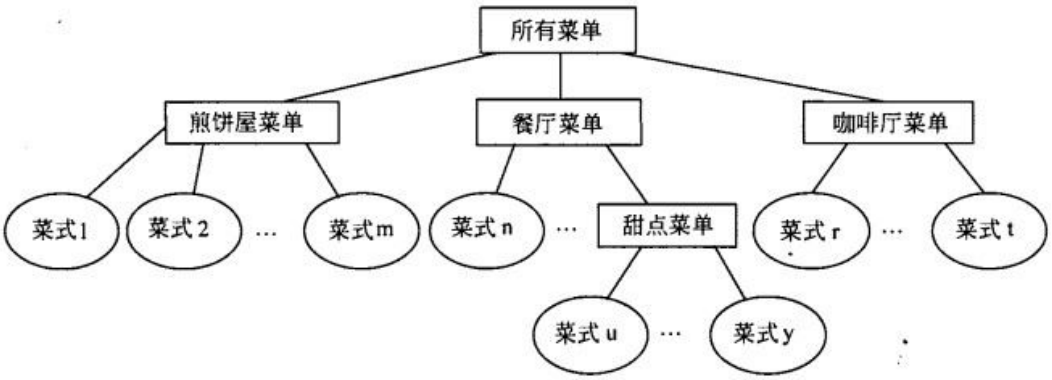


图 5-1　菜单结构图

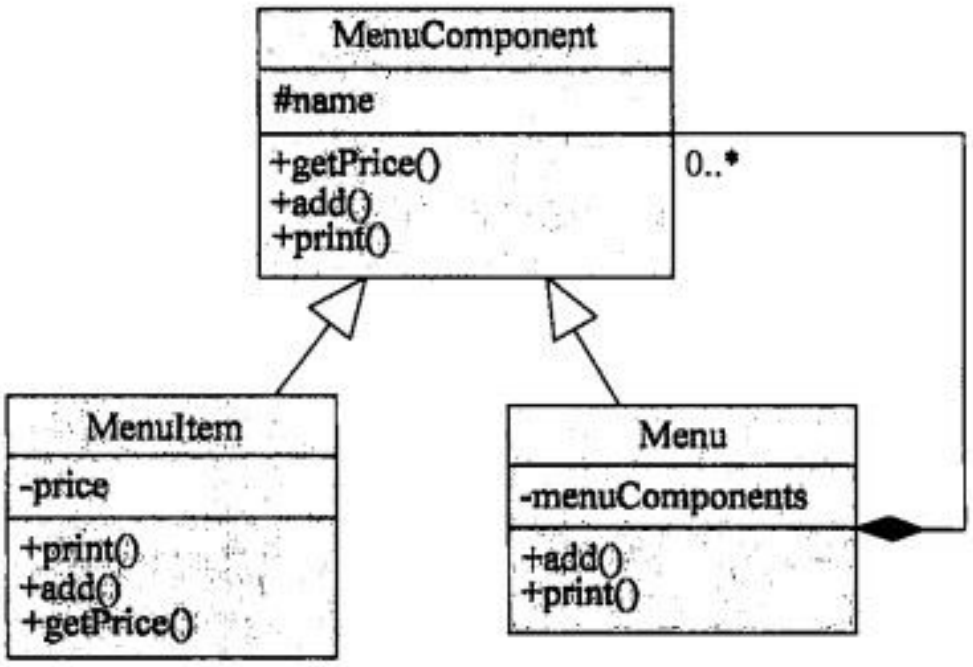


图 5-2　类图

问题：**5.1**

【C++代码】

```
#include <iostream>
#include <list>
#include <string>
using namespace std;
class MenuComponent {
protected:  string name;
public:
   MenuComponent(string name) {  this->name = name; }
   string getName() { return name; }
   __(1)__;                          // 添加新菜单
   virtual void print() = 0;         // 打印菜单信息
};
class MenuItem : public MenuComponent {
private:  double price;
public:
   MenuItem(string name, double price) : MenuComponent(name) { this->price
   = price; }
   double getPrice() { return price; }
   void add(MenuComponent* menuComponent) { return ; }  // 添加新菜单
   void print() { cout << "    " << getName() << ", " << getPrice() << endl; }
};
class Menu : public MenuComponent {
```

```
private:   list<__(2)__>  menuComponents;
public:
   Menu(string name): MenuComponent(name){}
   void add(MenuComponent* menuComponent)                    // 添加新菜单
   {    __(3)__;   }
   void print() {
     cout << "\n" << getName() << "\n----------------------------" << endl;
     std::list<MenuComponent *>::iterator iter;
     for(iter = menuComponents.begin(); iter != menuComponents.end(); iter++)
          __(4)__->print();
   }
};
void main() {
  MenuComponent* allMenus = new Menu("ALL MENUS");
  MenuComponent* dinerMenu = new Menu("DINER MENU");
  …… // 创建更多的 Menu 对象，此处代码省略
  allMenus->add(dinerMenu);          // 将 dinerMenu 添加到餐厅菜单中
  …… // 为餐厅增加更多的菜单，此处代码省略
  __(5)__->print();                  // 打印饭店所有菜单的信息
}
```

试题六　某饭店在不同的时段提供多种不同的餐饮，其菜单的结构图如图 6-1 所示。

现在采用组合(Composition)模式来构造该饭店的菜单，使得饭店可以方便地在其中增加新的餐饮形式，得到如图 6-2 所示的类图。其中 MenuComponent 为抽象类，定义了添加(add)新菜单和打印饭店所有菜单信息(print)的方法接口。类 Menu 表示饭店提供的每种餐饮形式的菜单，如煎饼屋菜单、咖啡屋菜单等。每种菜单中都可以添加子菜单，例如图 6-1 中的甜点菜单。类 MenuItem 表示菜单中的菜式。

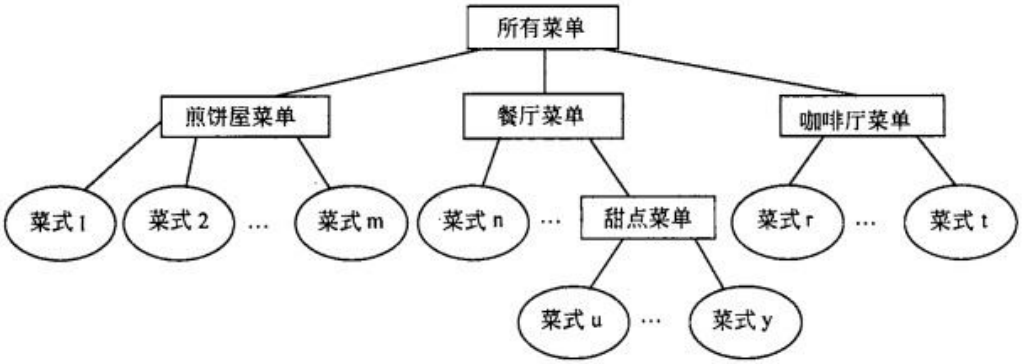


图 6-1　菜单结构图

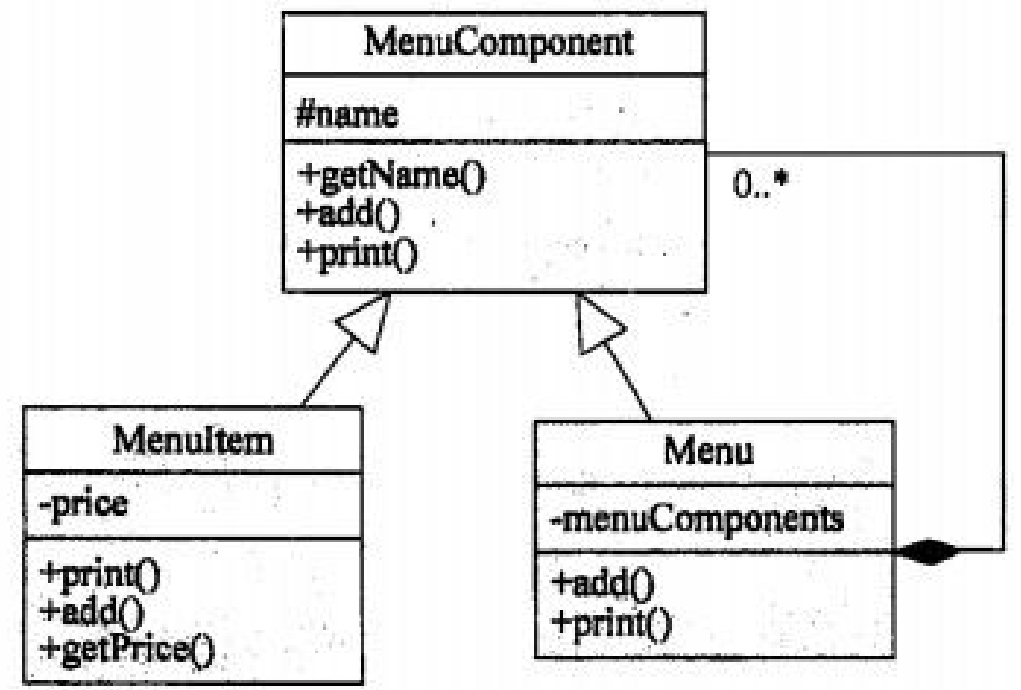


图 6-2　类图

问题： 6.1

【 Java 代码】

```
import java.util.*;
 (1)  MenuComponent {
   protected String name;
    (2)  ;  //添加新菜单
   public abstract void print();                          //打印菜单信息
   public String getName() { return name; }
}
class MenuItem extends MenuComponent {
   private double price;
   public MenuItem(String name, double price) {
       this.name = name;   this.price = price;
   }
   public double getPrice()  { return price; }
   public void add(MenuComponent menuComponent){ return ; } // 添加新菜单
   public void print() {
     System.out.print(" " + getName());
     System.out.println("," + getPrice());
   }
}
class Menu extends MenuComponent {
   private List<MenuComponent> menuComponents = new ArrayList<Menu-
   Component>();
   public Menu(String name) { this.name = name; }
   public void add(MenuComponent menuComponent) {            //添加新菜单
     menuComponents.  (3)  ;
   }
    public void print() {
      System.out.print("\n" + getName());
      System.out.println("," + "----------------------");
      Iterator iterator = menuComponents.iterator();
      while(iterator.hasNext()) {
         MenuComponent menuComponent = (MenuComponent)iterator.next();
          (4)  ;
      }
    }
}
class MenuTestDrive {
   public static void main(String args[]) {
       MenuComponent  allMenus = new Menu("ALL MENUS");
MenuComponent  dinerMenu = new Menu("DINER MENU");
       …… // 创建更多的 Menu 对象，此处代码省略
       allMenus.add(dinerMenu);    // 将 dinerMenu 添加到餐厅菜单中
       …… // 为餐厅增加更多的菜单，此处代码省略
        (5)  ;    // 打印饭店所有菜单的信息
   }
}
```

试题一 答案：　解析：　本问题考查顶层 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，因此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。题目要求根据描述来确定图中的外部实体。分析题目中的描述，并结合已经在顶层数据流图中给出的数据流进行分析。从中可以看出，与系统的交互者包括病人、医生和护理人员。其中，本地监控定期获取病人的生命特征，病人是生命特征数据来源，医生和护理人员提出相关请求，并得到相关报告结果，如请求病人生命特征报告，并获得相关报告。医生还需要在必要时添加或更新生命特征范围。对应图 1-1 中数据流和实体的对应关系，可知 E1 为病人， E2 为护理人员， E3 为医生。

本问题考查 0 层 DFD 中数据存储的确定。根据说明中描述：(2) 格式化生命特征： 对病人的各项重要生命特征数据进行格式化，然后存入日志文件并检查生命特征；(4) 维护生命特征范围：医生在必要时(如新的研究结果出现时)添加或更新生命特征值的正常范围；(6) 生成病历：根据日志文件中的生命特征，医生对病人的病情进行描述，形成病历存入病历文件；(8) 生成治疗意见：根据日志文件中的生命特征和病历，医生给出治疗意见，如处方等，并存入治疗意见文件。因此， D1 为生命特征范围文件， D2 为日志文件， D3 为病例文件， D4 为治疗意见文件。

本问题考查绘制 DFD 时的注意事项。在 DFD 中，每条数据流的起点和终点之一必须是加工(处理)。本题中，医生和护理人员根据查询到的治疗意见对病人进行治疗属于系统之外的行为，所以两个实体之间不可以有数据流。

数据流名称	起　点	终　点
重要生命特征	本地监控	格式化生命特征
格式化后的生命特征	格式化生命特征	检查生命特征
病例	生成病历	D3 或 病历（文件）
生命特征	D2 或 日志（文件）	生成病例

试题二 答案：　解析：

本问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，一个库管员可以管理多个仓库，每个仓库有一名库管员。所以，仓库实体和库管员实体之间存在“管理”联系，联系的类型为多对一(*:1)。

根据题目的需求描述可知，一个仓库中可以存放多类服装，一类服装可能存放在多个仓库中。所以，仓库实体和服装实体之间存在“存放”联系，联系的类型为多对多(*:*)。

根据题目的需求描述可知，一个采购订单可以包含多类服装，每类服装可由多个不同的供应商供应。所以，采购订单实体与服装实体和供应商实体三者之间存在“采购” 联系，三

者之间联系的类型为多对多对多(*:*:*)。

根据题目的需求描述可知，一个供应商可以供应多类服装，一类服装可由多个供应商供应。所以，供应商实体和服装实体之间存在“供应”联系，联系的类型为多对多(*:*)。

(1) 仓库编码，库管员编码

(2) 供应商编码，服装编码

(3) 订单编码，订货日期，应到货日期

(4) 订单编码，服装编码，供应商编码，数量，采购价格

本问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，系统记录库管员的库编码、姓名和级别。所以，对于“库管员”关系模式，需补充属性“库管员编码”。

根据实体联系图和需求描述，仓库信息主要包括：仓库编码、仓库位置、仓库容量和库管员。对于“仓库信息”关系模式，由于仓库实体与库管员实体有多对一联系，需记录对应的库管员，并且需补充属性——仓库编码。因此，“仓库信息，，关系模式，需补充属性“仓库编码”和“库管员编码”。

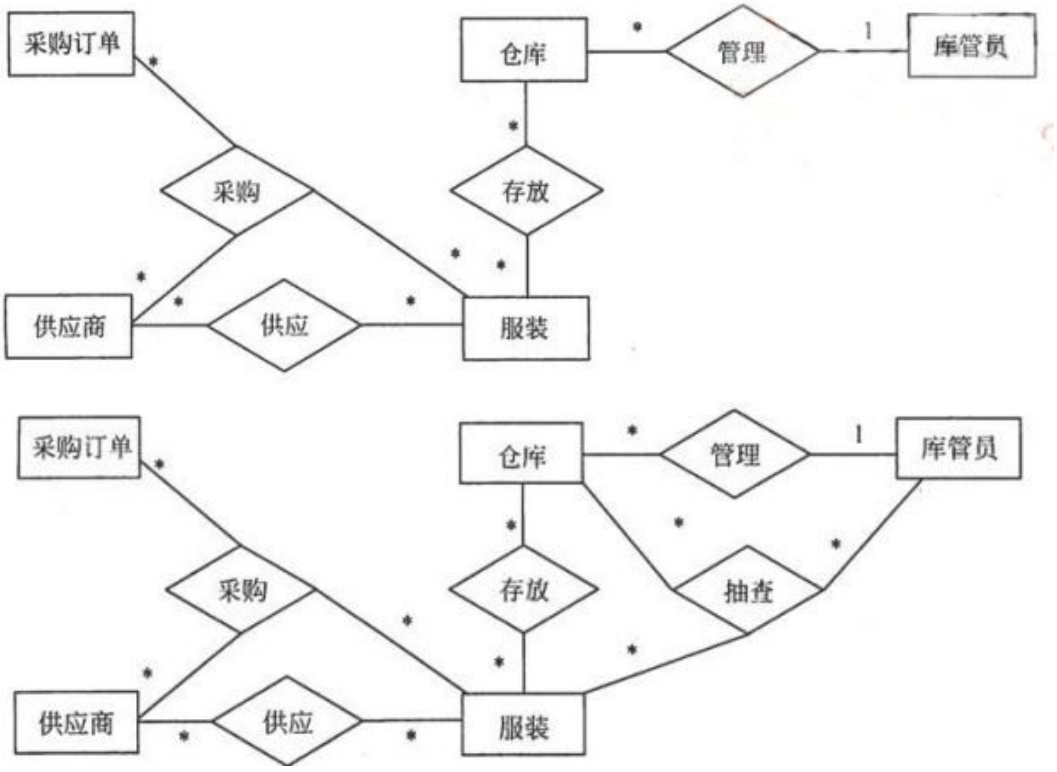
根据实体联系图和需求描述，供应商信息包括：供应商编码、供应商名称、地址、企业法人和联系电话。所以，对于“供应商”关系模式，需补充属性“供应商编码”。

根据实体联系图和需求描述，“供应情况”关系模式，需记录供应商和服装的多对多联系，即一个供应商可以供应多类服装，一类服装可由多个供应商供应。所以，对于“供应商”关系模式，需补充属性“供应商编码”和“服装编码”。

根据实体联系图和需求描述，采购订单主要记录订单编码、订货日期和应到货日期。所以，对于“采购订单”关系模式需补充属性：订单编码、订货日期和应到货日期。由于采购订单还需详细记录所采购的每类服装的数量、采购价格和对应的多个供应商。因此，“采购订单明细”关系模式，需记录采购订单实体与服装实体和供应商实体三者之间 存在的多对多对多联系。对于“采购订单明细”关系模式，需补充属性：订单编码、服装编码、供应商编码、数量和采购价格。

本问题考查的是数据库的概念结构设计，根据新增的需求增加实体联系图中的实体的联系和联系的类型。

根据问题描述，多个库管员需对每个仓库中的每一类被抽查服装记录一条抽查结果。则在库管员实体与仓库实体和服装实体三者之间存在“抽查”联系，联系的类型是多对多对多(*:*:*)。



试题三 答案：　解析：　本问题主要考查用例之间的关系。在 UML 中，用例之间有 3 种关系：包含(include)、 概括(generalize)和扩展(extend)。

如果多个用例中都含有相同的事件流，那么可以将其抽取出来放在一个单独的用例中，其他用例都可以通过包含(include)这个用例来使用其中的事件流。包含关系可以避免在多个用例的描述中重复拷贝相同的事件流。

概括关系是指子用例(childusecase)继承父用例(parentusecase)的行为，而子用例本身还可以增加新的行为或重置父类的某些行为。这种关系与面向对象程序设计中的“继承”很类似。

一个用例(基础用例， baseusecase)中加入一些新的动作后则构成了另外一个用例（扩展用例， extendingusecase)，那么这两个用例之间的关系就是扩展关系。扩展关系与概括关系有相似之处，但是比概括关系更为严格。基础用例必须声明特定的扩展点，而扩展用例只能在这些扩展点上添加新行为。

由说明可知，图形编辑器的基本操作为创建图形、创建元素、选择元素和删除图形。对照图 3-1，可知这些最终都被确定为用例。除此之外，用例“创建图元”、“创建文本” 与用例“创建元素”之间是概括关系，即能创建的元素分别是图元和文本。图 3-1 中缺少了两个用例，而这两个用例都是与“选择元素”相关的。因此需要仔细阅读说明中关于“选择元素”的描述，其中最关键的一句描述为“拖拽选定元素的控制点，可以移动元素或者

调整元素的大小”。这句话中出现了两个动词短语“移动元素”、“调整元素大小”，这两个动作都是要先选择对应元素之后，才能实施的。因此，可以推出， U1 和 U2 应对应“移动元素”和“调整元素大小”。

下一步就是确定“移动元素”、“调整元素大小”与“选择元素”之间的关系。由说明可知，必须先选择元素才能通过拖拽控制点来对元素进行相应的操作。因此，“移动元素”和“调整元素大小”是对“选择元素”的扩展，因此这三个用例之间应该是扩展关系。(1)和(2）处应填写 extend 。

本问题考查类图，考点是类层次结构及多重度。图 3-2 中有两个非常明显的继承结构，需要考生将其填充完整。这两个继承结构的最顶层父类分别是“工具”和“元素”，这就需要仔细阅读说明中与这两个词汇相关的描述。说明中第一次出现“工具”这个词，是在句子“编辑器提供了两种操作图形的工具：选择工具和创建工具”。这是典型的一般/特殊关系的描述，由此可以推断出， C1 和 C2 应该对应“选择工具”和“创建工具”， 到底是怎样的对应关系，还需要进一步的细节信息。说明中的①给出“创建工具用于创建文本元素和图元元素”，而 C1 的一个子类就是“文本工具”，所以可以确定 C1 是“创建工具”，C2 是“选择工具”。那么 C3 C5 应该就是与创建图元元素相关的工具了，而图元分为三类：线条、矩形和椭圆。所以 C3 C5 分别对应“线条工具”、“矩形工具” 和“椭圆工具”。

现在图 3-2 中左边的继承结构已经填充完整了。右边的继承结构就可以对应地填写出来了，C6 C8 分别对应的是类“线条”、“矩形”和“椭圆”。

确定多重度时，需要在说明寻找关联两端的类相关的描述。“对图形进行操作时，一次只能使用一种工具(即当前活动工具， active)”，即在图形编辑器中一次只能使用一个工具，而任何一个工具只属于这个图形编辑器。所以(3）处应填 0..1 ，(4）处应填 1 。

一个图形可以包含多个元素，对于一个图形中的特定元素来说，只能属于这个图形。所以(5）处应填 1 ，(6）处应填 1..*或*。

桥接模式将抽象部分与它的实现部分分离，使它们都可以独立地变化，对一个抽象的实现部分的修改应该对使用它的程序不产生影响。

本问题考查桥接模式，该模式将抽象部分与其实现部分分离，使它们都可以独立地变化。在以下情况中可以使用 Bridge 模式：

(1）不希望在抽象以及抽象的实现部分之间有一个固定的绑定关系。例如这种情况可能是因为，在程序运行时刻可以选择或切换实现部分。

(2）类的抽象以及它的实现都应该可以通过生成子类的方法加以扩充，使用 Bridge 模式可以对不同的抽象接口和实现部分进行组合，并分别对它们进行扩充。

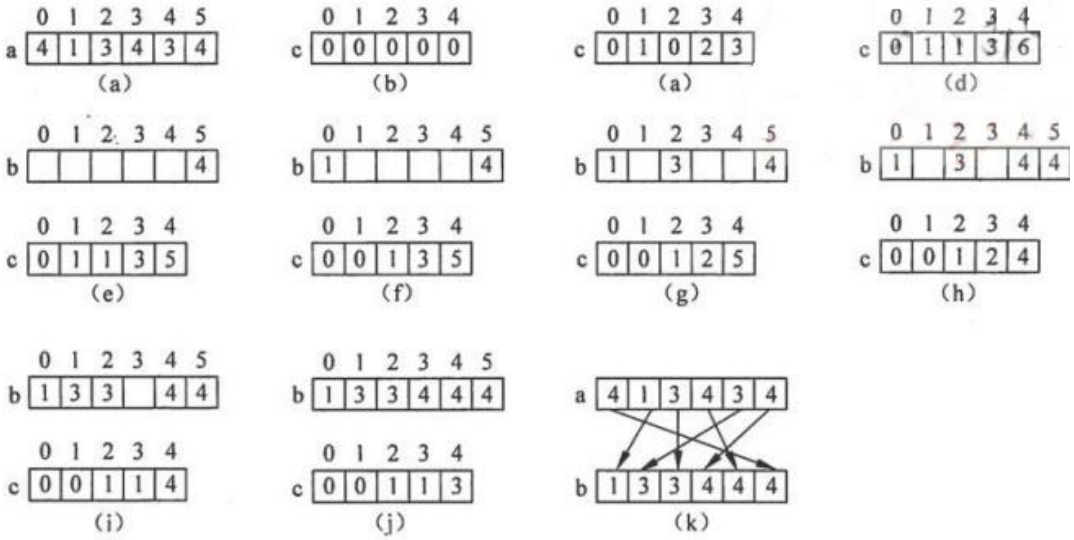
(3）对一个抽象的实现部分的修改应该对用户不产生影响，即客户的代码不必重新编译。

试题四 答案：　解析：　根据题中说明，第 3 到第 5 行代码进行 c 数组的初始化， c 数组的长度为 R，在 C 语言中，下标从 0 开始，因此空格(1) 中填写 R 。第 6 到第 8 行检查 a 数组的每一个元素。如果元素的值为 i，则增加 c[i]的值。因此 c[a[i]] = c[a[i]] + 1，空格(2) 填写 c[a[i]]+l 。 完成第 6 行到第 8 行的代码后， c[i]中就存放了等于 i 的元素的个数。第 9 到第 11 行， 通过在数组 c 中记录计数和， c[i] = c[i - 1] + c[i], 可以确定对每一个 i=0， 1，…， R-1, 有多 少个元素是小于或等于 i 的。因此空格(3) 填写 c[i - 1] + c[i]o 第 12 行到第 15 行把数组 a 中每个元素 a[i]放在输出数组 b 中与其相应的最终位置上， b[c[a[i]] - 1] =a[i]，因 此空格(4) 填写 a[i]。由于可能存在相同元素，因此每次将一个值 a[i]放入数组 b 中时， 都要减小 c[i]的值。下面以一个例子来说明排序过程。

设 a= {4， 1， 3,4,3,4}， R = 5, 即待排序的元素值在{0， 1， 2,3,4}中，其排序过程如下图 所示。

图中(a)为输入数组 a，(b)为初始化后的 c 数组，(c)为统计数组 a 中每个元素 后的 c 数组，(d)为计数和，即统计小于等于 i 的元素个数后的 c 数组。(e)到(j)是 将 a 数组中的元素依次放到 b 数组的过程，(k)是数组 a 和数组 b 的元素对应关系。

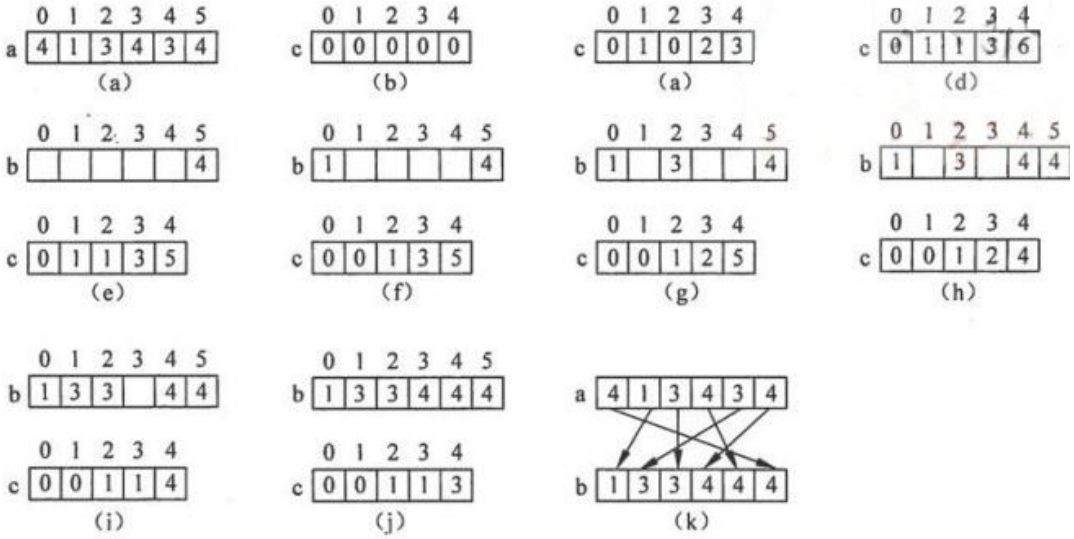
从图(k)可以看出，算法不稳定。算法不稳定的原因在于将数组 a 中元素放到数组 b 中时，是从数组 a 的第一个元素开始，依次取出元素放到数组 b 中。这样，相. 同的两个元素值，在数组 a 中的相对位置和在数组 b 中的相对位置正好相反。若从数组 a 的最后一个元素开始，依次向前取元素放到 b 数组中，可以保持相同元素的相对位置。因此将第 12 行的代码 for(i = 0; i=0;i−−), 则排序算法是稳定的。



（5）$\Theta(n+R)$ 或者 $\Theta(n)$ 或 n 或线性

（6）$\Theta(n+R)$ 或者 $\Theta(n)$ 或 n 或线性

根据上述 C 代码，第 3 到第 5 行代码的 for 循环所花时间为 $\Theta(R)$。第 6 到第 8 行的 for 循环所花时间为 $\Theta(n)$。第 9 到第 11 行的 for 循环所花时间为 $\Theta(R)$。第 12 到第 15 行 for循环所花时间为 $\Theta(n)$。因此整个算法的时间复杂度为 $\Theta(n+R)$。若 R 远小于 n 或者 $R=\Theta(n)$ 时，时间复杂度可以表示为 $\Theta(n)$。



试题五 答案：　解析：　(1) virtual void add(MenuComponent* menuComponent) = 0

(2) MenuComponent *

(3) menuComponents.push_back(menuComponent)

(4) (*iter)

(5) allMenus

Composite 模式将对象组合成树形结构以表示“整体-部分”的层次结构，其中的组合对象使得你可以组合基元对象以及其他的组合对象，从而形成任意复杂的结构。 Composite 模式使得用户对单个对象和组合对象的使用具有一致性。

Composite 模式的结构下图所示。

其中：

•类 Component 为组合中的对象声明接口，在适当的情况下，实现所有类共有接口的缺省行为，声明一个接口用于访问和管理 Component 的子部件；

•类 Leaf 在组合中表示叶节点对象，叶节点没有子节点；并在组合中定义图元对象的行为：

•类 Composite 定义有子部件的那些部件的行为，存储子部件，并在 Component 接口中实现与子部件有关的操作：

•类 Client 通过 Component 接口操纵组合部件的对象。

下列情况可以使用 Composite 模式：

(1) 表示对象的整体-部分层次结构；

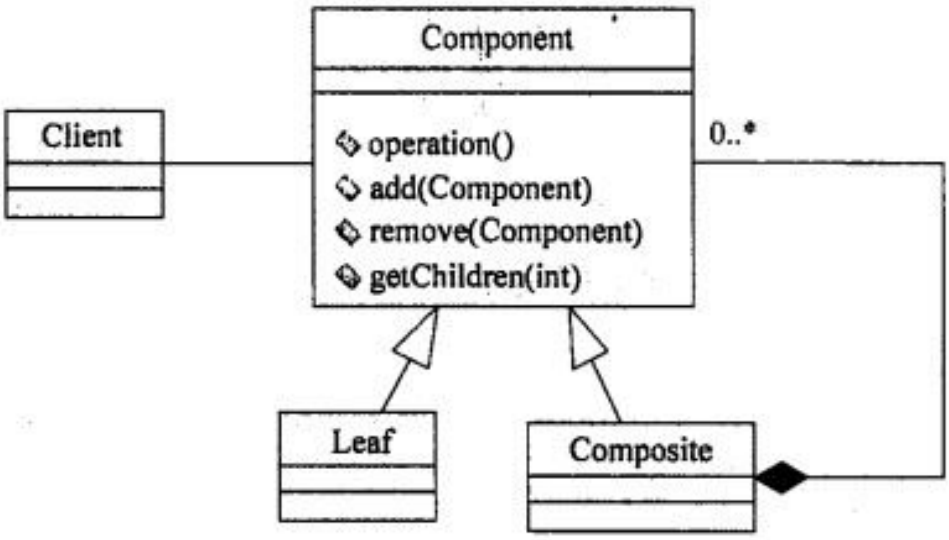
(2) 希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所 有对象。

试题五将组合模式应用到饭店菜单的构造中。图 5-2 中的类 MenuComponent 对应上图中的 Component，Menultem 对应 Leaf，Menu 对应 Composite 。在实现时，通常都会把 Component 定义为抽象类。

在 C++中，抽象类是指至少包含一个纯虚拟函数的类。类 MenuComponent 中已经包含了一个纯虚拟函数 print ，所以 MenuComponent 已经是一个抽象类了。(1) 处根据注释，这里应该定义功能为“添加新菜单”的成员函数。在子类 Menultem 和 Menu 中可以看到，都有 add 成员函数，说明子类中重置了父类中的成员函数。所以(1) 处的成员函数也应该定义为纯虚拟函数，即 virtualvoidadd(MenuComponent* menuComponent) = 0 。

由图 5-2 可以看出， Menu 中包含了 MenuComponent 的对象集合。程序中用 C++标准模板库中的 list 来实现这个聚集关系。因此(2) 处应填入 MenuComponent *。由于使用了 list ，就可以利用 list 中提供的各种方法了。 list 中用于添加元素的方法是 push_back ，所以(3) 处应填入 menuComponents.push_back(menuComponent)»

(4) 处出现在方法 print 中，其功能是打印出所有菜单的信息。这里使用了 list 中的迭代器类 iterator,遍历每个子菜单，并调用子菜单中定义的 print 方法打印该子菜单的 信息。(4) 处应填入*iter 。为了能够在 main 中打印出所有的菜单信息，必须使用表示菜单结构中最顶层菜单的对象来调用 print ，因此(5) 处应填入 allMenus 。



试题六 答案：　解析： (1) abstractclass 或 publicabstractclass

(2) public abstract void add(MenuComponent menuComponent)

或 abstractvoidadd(MenuComponentmenuComponent)

或 protectedabstractvoidadd(MenuComponentmenuComponent) ,

(3) add(menuComponent)

(4) menuComponent.print()

(5) allMenus.print()

Composite 模式将对象组合成树形结构以表示“整体-部分”的层次结构，其中的组合对象使得你可以组合基元对象以及其他的组合对象，从而形成任意复杂的结构。 Composite 模式使得用户对单个对象和组合对象的使用具有一致性。

Composite 模式的结构下图所示。

其中：

•类 Component 为组合中的对象声明接口，在适当的情况下，实现所有类共有接口的缺省行为，声明一个接口用于访问和管理 Component 的子部件；

•类 Leaf 在组合中表示叶节点对象，叶节点没有子节点；并在组合中定义图元对象的行为；

•类 Composite 定义有子部件的那些部件的行为，存储子部件，并在 Component 接口中实现与子部件有关的操作；

•类 Client 通过 Component 接口操纵组合部件的对象。

下列情况可以使用 Composite 模式：

(1）表示对象的整体-部分层次结构；

(2）希望用户忽略组合对象与单个对象的不同，用户将统一地使用组合结构中的所有对象。

试题六将组合模式应用到饭店菜单的构造中。图 6-2 中的类 MenuComponent 对应上图中的 Component ， Menultem 对应 Leaf ， Menu 对应 Composite 。在实现时，通常都会把 Component 定义为抽象类。

在 Java 中，用 abstract 关键字限定的类即为抽象类，所以(1）处应填入 abstractclass 。

(2）处根据注释，这里应该定义功能为“添加新菜单”的成员函数。在子类 Menultem 和 Menu 中可以看到，都有 add 成员函数，说明子类中重置了父类中的成员函数。所以(2）处应填入 publicabstractvoidadd(MenuComponentmenuComponent)。

由图 6-2 可以看出， Menu 中包含了 MenuComponent 的对象集合。程序中用 Java 中的 list 来实现这个聚集关系，这样就可以利用 list 中提供的各种方法了。 list 中用于添加元素的方法是 add, 所以(3）处应填入 add(menuComponent)。

(4）处出现在方法 print 中，其功能是打印出所有菜单的信息。这里使用了 list 中的迭代器类 iterator, 遍历每个子菜单，并调用子菜单中定义的 print 方法打印该子菜单的信息。

(4)处应填入 menuComponent.print() 。

为了能够在 main 中打印出所有的菜单信息，必须使用表示菜单结构中最顶层菜单的对象来调用 print ，因此(5）处应填入 allMenus.print() 。