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 2017 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 【说明】

某医疗器械公司作为复杂医疗产品的集成商，必须保持高质量部件的及时供应。为了实现这一目标，该公司欲开发一采购系统。系统的主要功能如下:

1. 检查库存水平。采购部门每天检查部件库存量，当特定部件的库存量降至其订货点时，返回低存量部件及库存量。

2. 下达采购订单。采购部门针对低存量部件及库存量提交采购请求，向其供应商(通过供应商文件访问供应商数据)下达采购订单，并存储于采购订单文件中。

3. 交运部件。当供应商提交提单并交运部件时，运输和接收(S/R)部门通过执行以下三步过程接收货物：

(1) 验证装运部件。通过访问采购订单并将其与提单进行比较来验证装运的部件，并将提单信息发给 S/R 职员。 如果收货部件项目出现在采购订单和提单上，则已验证的提单和收货部件项目将被送去检验。否则，将 S/R 职员提交的装运错误信息生成装运错误通知发送给供应商。

(2) 检验部件质量。通过访问质量标准来检查装运部件的质量，并将已验证的提单发给检验员。如果部件满足所有质量标准，则将其添加到接受的部件列表用于更新部件库存。如果部件未通过检查，则将检验员创建的缺陷装运信息生成缺陷装运通知发送给供应商。

(3) 更新部件库存。库管员根据收到的接受的部件列表添加本次采购数量，与原有库存量累加来更新库存部件中的库存量。标记订单采购完成。

现采用结构化方法对该采购系统进行分析与设计，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。

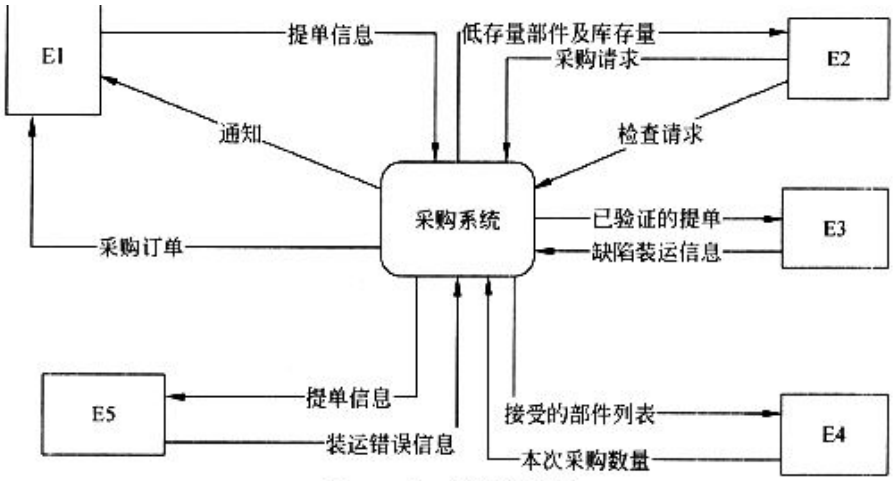


图 1-1　上下文数据流图

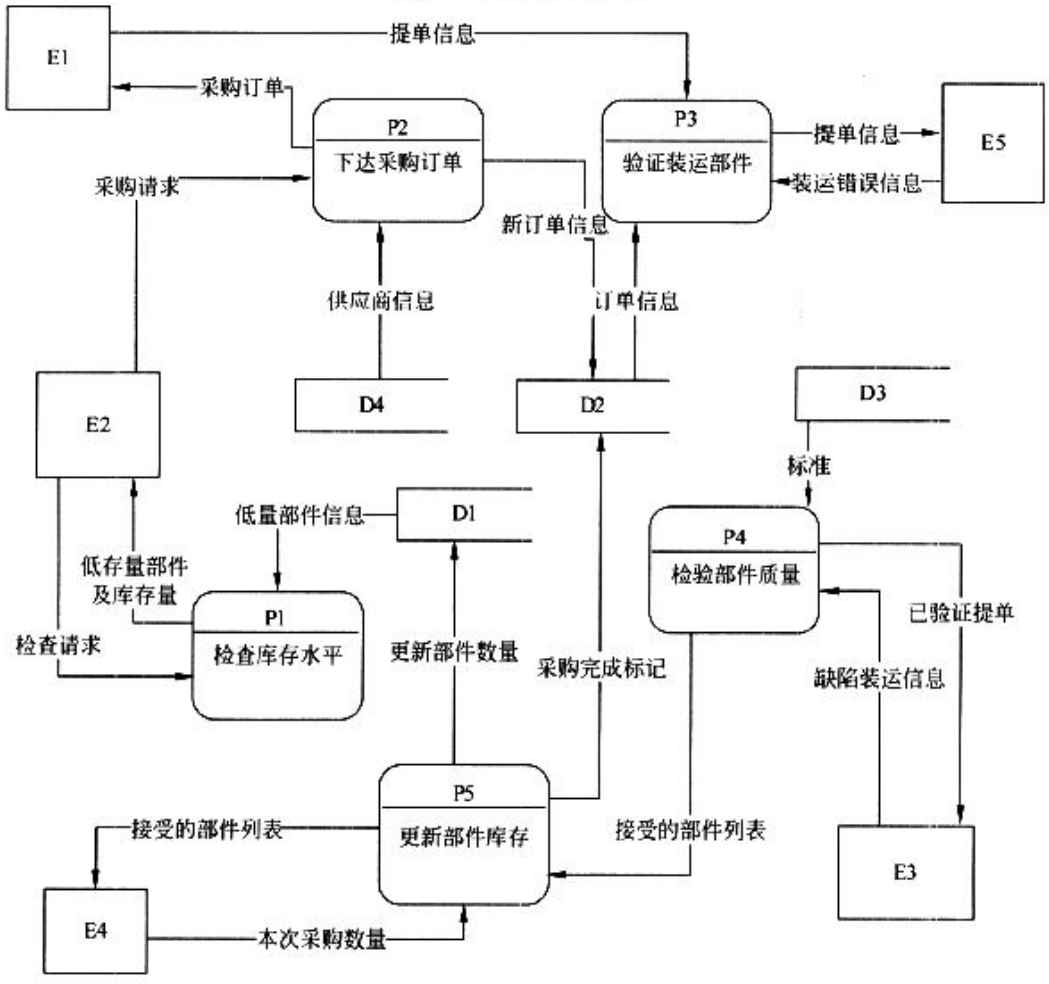


图 1-2　0 层数据流图

问题：1.1
(5 分)
使用说明中的词语，给出图 1-1 中的实体 E1　E5 的名称。
问题：1.2
(4 分)
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
(4 分)
根据说明和图中术语，补充图 1-2 中缺失的数据流及其起点和终点。
问题：1.4
(2 分)
用 200 字以内文字，说明建模图 1-1 和图 1-2 时如何保持数据流图平衡。

试题二　【说明】

某房屋租赁公司拟开发一个管理系统用于管理其持有的房屋、租客及员工信息。请根据下述需求描述完成系统的数据库设计。

【需求描述】

1. 公司拥有多幢公寓楼，每幢公寓楼有唯一的楼编号和地址。每幢公寓楼中有多套公寓，每套公寓在楼内有唯一的编号(不同公寓楼内的公寓号可相同)。系统需记录每套公寓的卧室数和卫生间数。

2. 员工和租客在系统中有唯一的编号(员工编号和租客编号)。

3. 对于每个租客，系统需记录姓名、多个联系电话、一个银行账号(方便自动扣房租)、一个紧急联系人的姓名及联系电话。

4. 系统需记录每个员工的姓名、一个联系电话和月工资。员工类别可以是经理或维修工，也可兼任。每个经理可以管理多幢公寓楼。每幢公寓楼必须由一个经理管理。系统需记录每个维修工的业务技能，如：水暖维修、电工、木工等。

5. 租客租赁公寓必须和公司签订租赁合同。一份租赁合同通常由一个或多个租客(合租)与该公寓楼的经理签订，一个租客也可租赁多套公寓。合同内容应包含签订日期、开始时

间、租期、押金和月租金。

【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】

根据概念摸型设计阶段完成的实体联系图，得出如下关系模式(不完整)：

联系电话(电话号码，租客编号)

租客(租客编号，姓名，银行账号，联系人姓名，联系人电话)

员工(员工编号，姓名，联系电话，类别，月工资，(a))

公寓楼(（b），地址，经理编号)

公寓(楼编号，公寓号，卧室数，卫生间数〉

合同(合同编号，租客编号，楼编号，公寓号，经理编号，签订日期，起始日期，租期，(c)，押金)

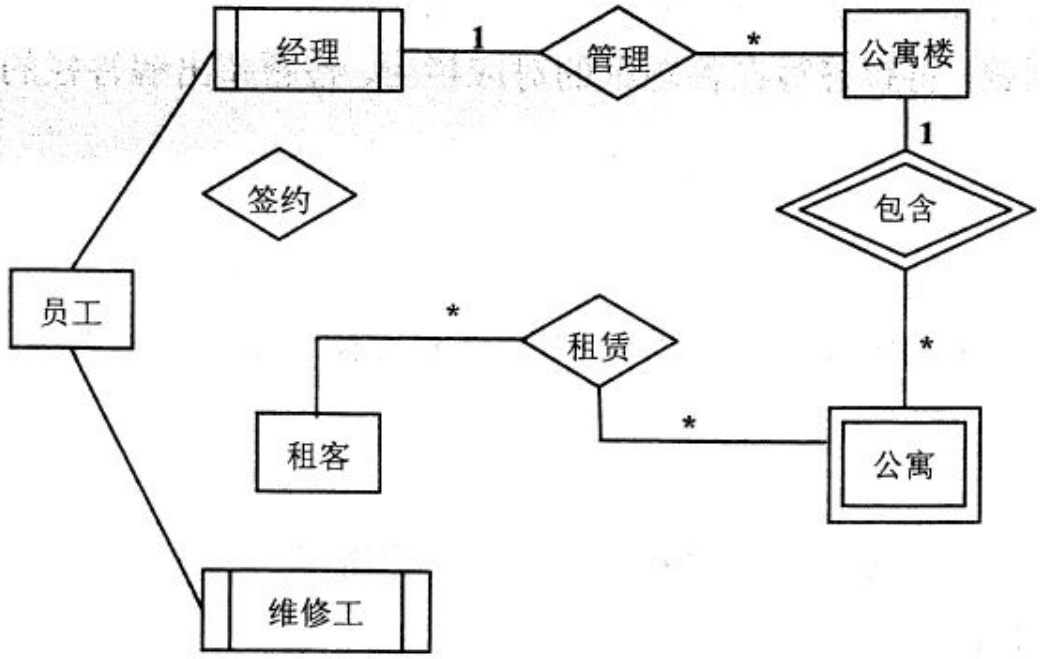


图 2-1 实体联系图

问题： 2.1
(4.5 分)
补充图 2-1 中的“签约”联系所关联的实体及联系类型。
问题： 2.2
(4.5 分)
补充逻辑结构设计中的(a)、(b)、(c)三处空缺。
问题： 2.3
(6 分)
在租期内，公寓内设施如出现问题，租客可在系统中进行故障登记，填写故障描述，每项故障由系统自动生成唯一的故障编号，由公司派维修工进行故障维修，系统需记录每次维修的维修日期和维修内容。请根据此需求，对图 2-1 进行补充，并将所补充的 ER 图内容转换为一个关系模式，请给出该关系模式。

试题三 【说明】

某玩具公司正在开发一套电动玩具在线销售系统，用于向注册会员提供端对端的玩具定制和销售服务。在系统设计阶段，″创建新订单（NewOrder）″的设计用例详细描述如表 3-1 所示，候选设计类分类如表 3-2 所示，并根据该用例设计出部分类图如图 3-1 所示。

在订单处理的过程中，会员可以点击“取消订单″取消该订单。如果支付失败，该订单将被

标记为挂起状态，可后续重新支付，如果挂起超时 30 分钟未支付，系统将自动取消该订单。订单支付成功后，系统判断订单类型：

(1）对于常规订单，标记为备货状态，订单信息发送到货运部，完成打包后交付快递发货；

(2）对于定制订单，会自动进入定制状态，定制完成后交付快递发货。会员在系统中点击”收货″按钮变为收货状态，结束整个订单的处理流程。根据订单处理过程所设计的状态图如图 3-2 所示。

表 3-1　创建新订单（New Order）设计用例

用例名称	创建新订单 New Order	
用例编号	ETM-R002	
参与者	会员	
前提条件	会员已经注册并成功登录系统	
典型事件流	1. 会员（C1）单击“新订单”按钮； 2. 系统列出所有正在销售的电动玩具清单及价格（C2）； 3. 会员点击复选框选择所需电动玩具并输入对应数量，单击“结算”按钮； 4. 系统自动计算总价（C3），显示销售清单和会员预先设置个人资料中的收货地址和支付方式（C4）； 5. 会员单击“确认支付”按钮； 6. 系统自动调用支付系统（C5）接口支付该账单； 7. 若支付系统返回成功标识，系统生成完整订单信息持久存储到数据库订单表（C6）中； 8. 系统将以表格形式显示完整订单信息（C7），同时自动发送完整订单信息（C8）到会员预先配置的邮箱地址（C9）。	
候选事件流	3a.	（1）会员单击“定制”按钮； （2）系统以列表形式显示所有可以定制的电动玩具清单和定制属性（如尺寸、颜色等）（C10）； （3）会员单击单选按钮选择所需要定制的电动玩具并填写所需要定制的属性要求，单击“结算”按钮； （4）回到步骤 4。
	7a.	（1）若支付系统返回失败标识，系统显示会员当前默认支付方式（C11）让会员确认； （2）若会员单击“修改付款”按钮，调用“修改付款”用例，可以新增并存储为默认支付方式（C12），回到步骤 4； （3）若会员单击“取消订单”，则该用例终止执行。

表 3-2　候选设计类分类

接口类（Interface，负责系统与用户之间的交互）	（a）
控制类（Control，负责业务逻辑的处理）	（b）
实体类（Entity，负责持久化数据的存储）	（c）

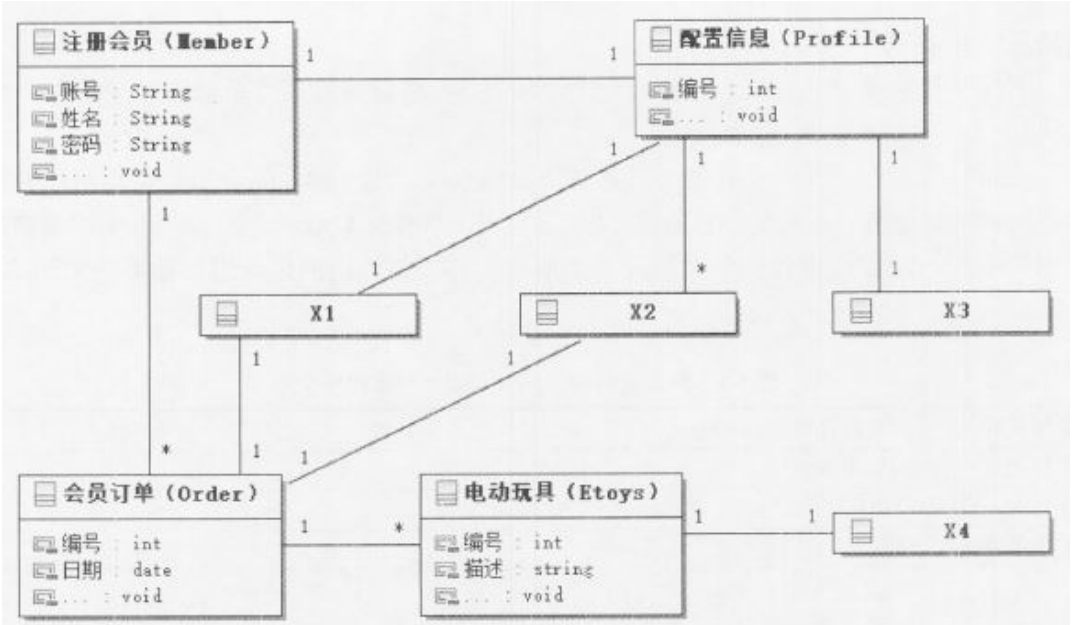


图 3-1　部分类图

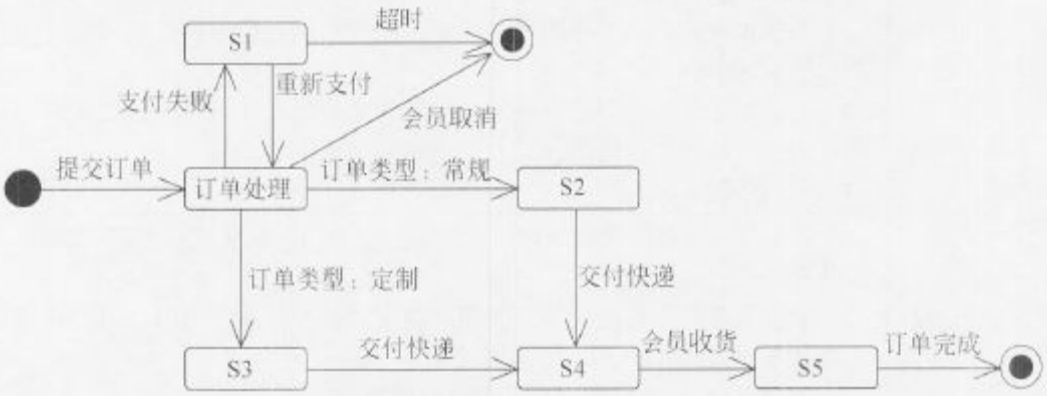


图 3-2　订单状态图

问题：3.1

(6 分)

根据表 3-1 中所标记的候选设计类，请按照其类别将编号 C1　C12 分别填入表 3-2 中的(a)、(b)和(c)处。

问题：3.2

(4 分)

根据创建新订单的用例描述，请给出图 3-1 中 X1　X4 处对应类的名称。

问题：3.3

(5 分)

根据订单处理过程的描述，在图 3-2 中 S1　S5 处分别填入对应的状态名称。

试题四　【说明】

假币问题：有 n 枚硬币，其中有一枚是假币，已知假币的重量较轻。现只有一个天平，要求用尽量少的比较次数找出这枚假币。

【分析问题】

将 n 枚硬币分成相等的两部分:

(1）当 n 为偶数时，将前后两部分，即 1...n/2 和 n/2+1...n，放在天平的两端，较轻的一端里有假币，继续在较轻的这部分硬币中用同样的方法找出假币；

(2）当 n 为奇数时，将前后两部分，即 1..(n -1)/2 和(n+1)/2+1...n，放在天平的两端，较轻的一端里有假币，继续在较轻的这部分硬币中用同样的方法找出假币；若两端重量相等，则中间的硬币，即第 (n+1)/2 枚硬币是假币。

【C 代码】

下面是算法的 C 语言实现，其中:

coins[]：硬币数组

first,last：当前考虑的硬币数组中的第一个和最后一个下标

```
#include<stdio.h>

int getCounterfeitCoin(int coins[], int first, int last)
{
    int firstSum = 0, lastSum = 0;
    int i;
    if(first == last - 1){                          /*只剩两枚硬币*/
        if(coins[first] < coins[last])
            return first;
        return last;
    }

    if((last - first + 1) % 2 == 0){     /*偶数枚硬币*/
        for(i = first;i < __(1)__;i++){
            firstSum += coins[i];
        }
        for(i = first + (last - first) / 2 + 1;i < last + 1;i++){
            lastSum += coins[i];
        }
        if( __(2)__ ){
            return getCounterfeitCoin(coins, first, first + (last - first) / 2);
        }else{
            return getCounterfeitCoin(coins, first + (last - first) / 2 + 1,
            last);
        }
    }
    else{        /*奇数枚硬币*/
        for(i = first;i < first + (last - first) / 2;i++){
            firstSum += coins[i];
        }
        for(i = first + (last - first) / 2 + 1;i < last + 1;i++){
            lastSum += coins[i];
        }
        if(firstSum < lastSum){
            return getCounterfeitCoin(coins, first, first + (last - first)
            / 2 - 1);
        }else if(firstSum > lastSum){
            return getCounterfeitCoin(coins, first + (last - first) / 2 + 1,
            last);
        }else{
            return __(3)__;
        }
    }
}
```

问题： 4.1

(6 分)

根据题干说明，填充 C 代码中的空(1) (3)。

问题： 4.2
(6 分)
根据题干说明和 C 代码，算法采用了(4)设计策略。
函数 getCounterfeitCoin 的时间复杂度为(5)(用 0 表示)。
问题： 4.3
(3 分)
若输入的硬币数为 30，则最少的比较次数为(6)，最多的比较次数为(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】

某快餐厅主要制作并出售儿童套餐，一般包括主餐(各类比萨)、饮料和玩具，其餐品种类可能不同，但其制作过程相同。前台服务员(Waiter)调度厨师制作套餐。现采用生成器(Builder）模式实现制作过程，得到如图 5-1 所示的类图。

【C++代码】

```
#include
using namespace std;

class Pizza {
private: string parts;
public:
        void setParts(string parts)
        {
                this->parts = parts;
        }

        string getParts()
        {
                return(parts);
        }
};
class PizzaBuilder {
protected: Pizza* pizza;
public:
        Pizza* getPizza()
        {
                retum pizza;
        }
```

```
        void createNewPizza()
        {
                pizza = new Pizza();
        }

        (1);
}
class HawaiianPizzaBuilder : public PizzaBuilder {
public:
        void buildParts()
        {
                pizza->setParts("cross +mild + ham&pineapple");
        }
};
class SpicyPizzaBuider : public PizzaBuilder {
public:
        void buildParts()
        {
                pizza->setParts("pan baked +hot + ham&pineapple");
        }
}
class Waiter{
private:
        PizzaBuilder * pizzaBuilder;
public:
        voidsetPizzaBuilder(PizzaBuilder* pizzaBuilder) /*设置构建器*/
        {
                (2)
        }

        Pizza* getPizza()
        {
                return(pizzaBuilder->getPizza());
        }

        voidconstruct() /*构建*/
        {
                pizzaBuilder->createNewPizza();
                (3)
        }
};

int main()
{
        Waiter        *waiter = new Waiter();
        PizzaBuilder  *hawaiian_pizzabuilder = new HawaiianPizzaBuilder()
        (4);
        (5);
        cout << "pizza: " < getPizza()->getParts() << end1;
}
```

程序的输出结果为:

pizza: cross+mild+ham&pineapple

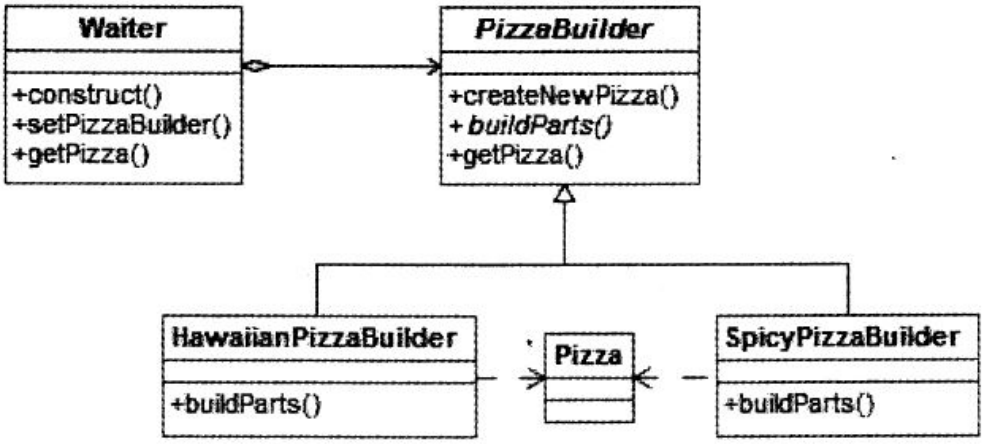


图 5-1　类图

问题：5.1
请填写(1)(2)(3)(4)(5)

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】

某快餐厅主要制作并出售儿童套餐，一般包括主餐(各类比萨)、饮料和玩具，其餐品种类可能不同，但其制作过程相同。前台服务员(Waiter) 调度厨师制作套餐。现采用生成器(Builder) 模式实现制作过程，得到如图 6-1 所示的类图。

【 Java 代码】

```
class Pizza {
        privateStringparts;
                public void setParts(String parts) {
                this.parts = parts;
        }
        public String toString() {
                return this.parts;
        }
}
```

```
	abstract class PizzaBuilder {
		protected Pizza pizza;
		public Pizza getPizza() {
			return pizza;
		}
		public void createNewPizza() {
			pizza = new Pizza();
		}
		public (1);
	}
	class HawaiianPizzaBuilder extends PizzaBuilder {
		public void buildParts() {
			pizza.setParts("cross+mild+ham&pineapple");
		}
	}
	class SpicyPizzaBuilder extends PizzaBuilder {
		public void buildParts() {
			pizza.setParts("pan baked+hot+pepperoni&salami");
		}
	}
	class Waiter {
		private PizzaBuilder pizzaBuilder;
		publicvoidsetPizzaBuilder(PizzaBuilderpizzaBuilder) { /*设置构建器*/
			(2);
		}
		public Pizza getPizza() { return pizzaBuilder.getPizza(); }
		publicvoidconstruct() { /*构建*/
			pizzaBuilder.createNewPizza();
			(3);
		}
	}
	class FastFoodOrdering {
		public static viod mainSting[]args) {
			Waiter waiter = new Waiter();
			PizzaBuilder hawaiian_pizzabuilder = new
HawaiianPizzaBuilder();
			(4);
			(5);
			System.out.println("pizza : " + waiter.getPizza());
		}
	}
```

程序的输出结果为：

Pizza:cross+mild+ham&pineapple

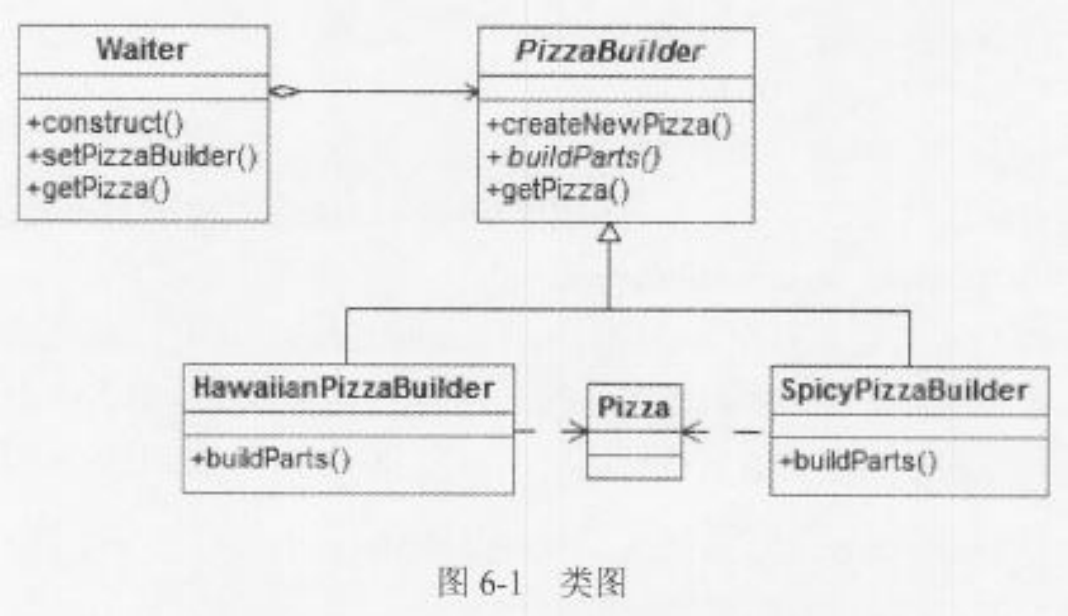


图 6-1 类图

问题： 6.1
请填写(1)(2)(3)(4)(5)

试题一 答案： 解析： E1：供应商

E2：采购部

E3：检验员

E4：库管员

E5： S/R 职员

本题考查采用结构化方法进行系统分析与设计中对数据流图(DFD)的应用，考点与往年类似，要求考生细心分析题目中所描述的内容。题干描述清晰，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工具。 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，并采用自顶向下分层且逐层细化的方式，建模不同详细程度的数据流图模型。

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，然后根据为系统提供输入数据流，以及接受系统发送的数据流，来确定系统的外部实体，以及外部实体和加工之间的输入输出数据流。

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数据流以及结果加工变换后的输出数据流，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。

本问题考查的是上下文 DFD，要求确定外部实体。在上下文 DFD 中，待系统名称 “采购系统”作为唯一加工的名称，外部实体为这一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，系统中涉及到供应商、采购部、检验员、库管员以及 S/R 职员。根据说明 1 中“采购部门每天检查部件库存量”，说明 2 中“向其供应商下达采购订单”、说明 3．(1）中“并将提单信息发给 S/R 职员”，3．(2）中“并将已验证的提单发给检验员”，以及 3．(3）中“库管员根据收到的接收的部件列表添加本次采购数量”等信息，对照图 M，从而即可确定 E1 为“供应商”实体，E2 为“采购部”实体，E3 为“检验员”实体，E4 为“库管员”实体，E5 为“S/R 职员”实体。

D1：库存

D2：采购订单

D3：质量标准

D4：供应商

(注：名称后面可以带有“文件”)

本问题要求确定图 1-20 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“每天检查部件库存量”以及说明 3．(3）中“与原有库存量累加来更新库存部件中的库存量”，可知 D1 为库存；说明 2 中“向其供应商(通过供应商文件访问供应商数据)下达采购订单，并存储于采购订单文件中”，可知 D2 为采购订单、D4 为供应商；说明 3．(2）中“通过访问质量标准来检查装运部件的质量”，可知 D3 为质量标准。

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，缺少了从加工到外部实体 E1（供应商)的数据流—“通知”，根据说明中发给供应商的通知分为两种情况，一种是在验证装运部件时出现不符合采购订单和提单信息的情况下，“将 S/R 职员提交的装运错误信息生成装运错误通知发送给供应商”；另一种情况是在检验部件质量时，“如果部件未通过检查，则将检验员创建的缺陷装运信息生成缺陷装运通知发送给供应商”。所以缺少了两条数据流，加工“验证装运部件”流出的数据流“装运错误通知”和加工“检验部件质量”流出的数据流“缺陷装运通知”，这两条数据流的综合即为上下文 DFD 中的“通知”。

再考查说明中的功能判定是否缺失内部的数据流，不难发现缺失的数据流。先考查说明 3．(2）中“如果收货部件项目出现在采购订单和提单上，则已验证的提单和收货部件项目将

被送去检验”，发现在图 1-2 中缺失，起点为“验证装运部件”，终点为“更新部件库存”。再说明 3．(3) 中“与原有库存量累加来更新库存部件中的库存量”，加工“更新部件库存”需要从数据存储“库存(D1)”中取出原有部件库存量，与“接收到的部件量”累加后得到“更新部件数量”更新库存部件中的库存量，图 1-2 中缺失了从 D1 到 P5 的数据流“原有部件库存量”。

图 1-1（或父图)中某加工的输入输出数据流必须与图 1-2（或子图)的输入输出数据流在数量和名字上相同；图 1-1（或父图)中的一个输入(或输出)数据流对应于图 1-2（或子图)中几个输入(或输出)数据流，而图 1-2（或子图)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。

在自顶向下建模分层 DFD 时，会因为加工的细分而发生数据流的分解情况，需要注意保持数据流图之间的平衡(本题中图 1-1 和图 1-2)。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。

数据流	起点	终点
装运错误通知	P3 或 验证装运部件	E1 或 供应商
缺陷装运通知	P4 或 检验部件质量	E1 或 供应商
原有部件库存量	D1 或 库存	P5 或 更新部件库存
已验证的提单信息	P3 或 验证装运部件	P4 或 检验部件质量

注：表中数据流顺序无关

试题二 答案：　解析：　补充内容如下图中虚线所示。

本题考查数据库概念设计及逻辑设计中 E-R 图向关系模式的转换方法。

此类题目要求考生认真阅读题目中对需求问题的描述，经过分类、聚集、概括等方法，从中确定实体及其联系。题目已经给出了 6 个实体以及部分实体之间的联系，需要根据需求描述，将实体之间联系补充完整。

题目中已经给出了租客与公寓间的租赁关系，由“一份租赁合同通常由一个或多个租客(合租)与该公寓楼的经理签订”可知，需要建立经理和“租客与公寓间的租赁关系”之间的联系，即将联系作为实体，参与下一次联系，使用聚合的方法。因此，最后的结果如参考答案图中虚线部分所示。

(a）业务技能

(b）楼编号

(c）月租金

从需求描述 4 中“系统需记录每个维修工的业务技能”，可知员工的属性信息需要业务技能属性。由需求 1 中“每幢公寓楼有唯一的楼编号和地址”，可知楼编号是唯一的，不会重复，可作为公寓楼的主键属性。需求 5 中说明了合同的属性信息中包含签定日期、开始时间、租期、押金、月租金，模式中还缺少月租金属性。完整的关系模式如下：

ER 图的补充方式不唯一，补充内容如 ER 图一或 ER 图二中虚线所示。

ER 图一

ER 图二

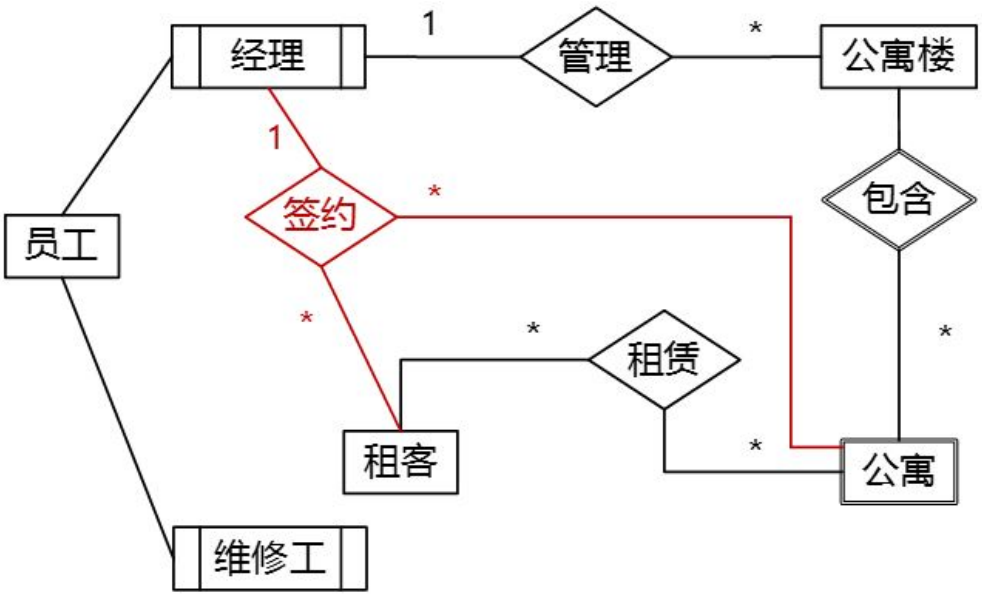
关系模式：维修记录(故障编号，租客编号，楼编号，公寓号，故障描述，员工编号，维修日期，维修内容)

备注：此联系名称能够合理表达需求即可。

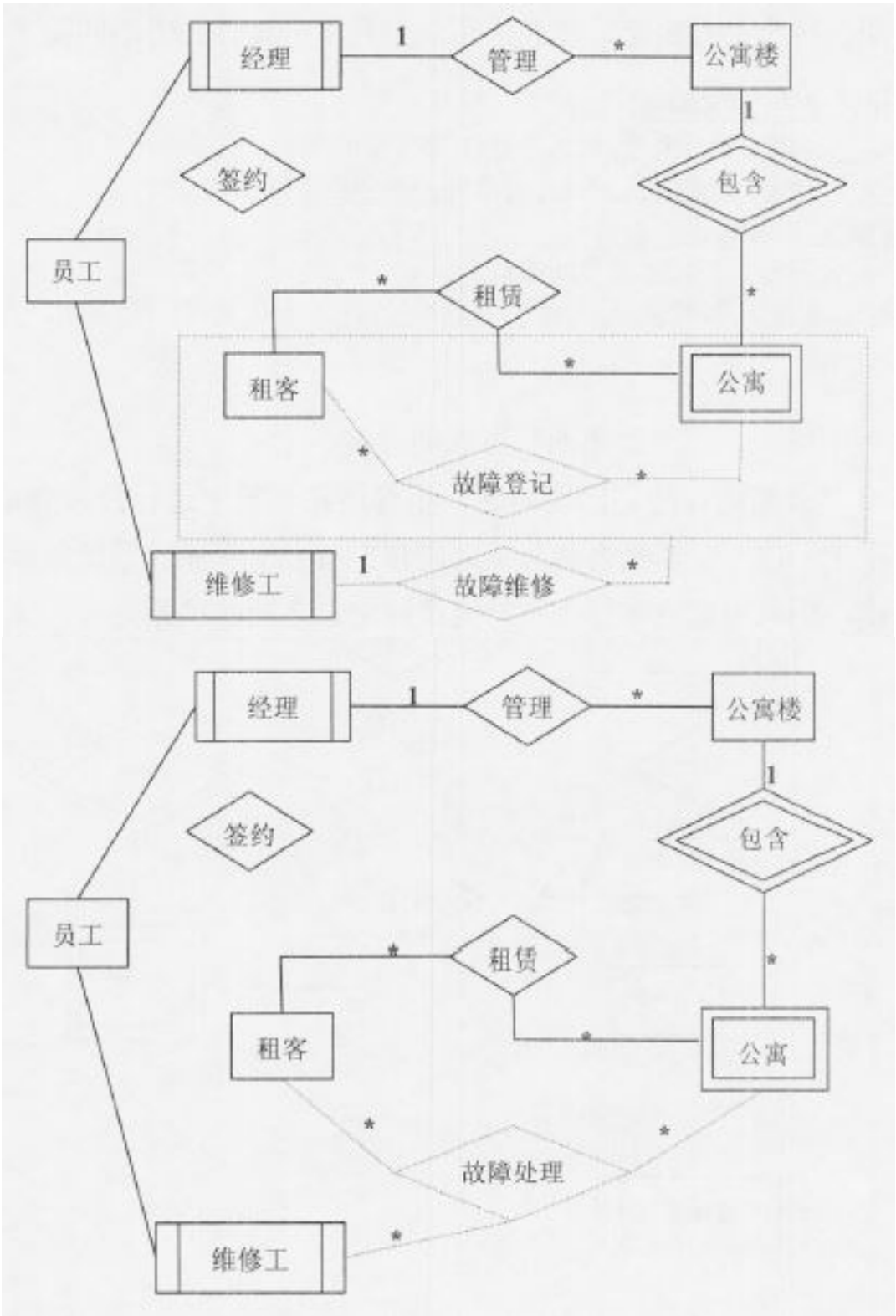
此题 E-R 图不唯一，这里给出两种备选的答案。

答案一：由“公寓内设施如出现问题，租客可在系统中进行故障登记”，但公寓出现问题的次数不止一次，可知租客和公寓之间存在着 m:n 联系。系统故障生成之后会派维修工进行维修，因此可建立维修工和特定故障记录之间的联系。

答案二：也可直接建立租客、公寓和维修工之间的三元联系。



联系电话(电话号码,租客编号)
租客(租客编号,姓名,银行账号,联系人姓名,联系人电话)
员工(员工编号,姓名,联系电话,类别,月工资, 业务技能)
公寓楼(楼编号 ,地址,经理编号)
公寓(楼编号,公寓号,卧室数,卫生间数)
合同(合同编号,租客编号,楼编号,公寓号,经理编号,签订日期,起始日期,租期, 月租金 ,押金)



试题三 答案：　　解析：　(a)C2 、 C4 、 C7 、 C10 、 C11

(b)C3 、 C5 、 C8

(c)C1 、 C6 、 C9 、 C12

本题考查面向对象设计方法及应用。

面向对象设计是一种工程化软件设计规范，其基本思想包括抽象、封装和可扩展性。类封装了信息和行为，是面向对象技术的重要组成部分，类是具有相同属性、方法和关系的对象集合的总称。在系统中，每个类都具有一定的职责，即指类所担任的任务。一个类可以有多种职责，设计得好的类一般至少有一种职责。类图描述了模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其他类的关系等。状态图是描述一个实体基于事件反映的动态行为，显示了该实体如何根据当前所处的状态对不同的事件作出反应。

面向对象设计是软件设计师必须掌握的专业知识与技能，特别是需要掌握软件类设计、类图和状态图等设计内容。

设计类是面向对象设计中最重要的组成部分，也是最复杂和最耗时的部分。面向对象设计过程中，类可以分为三种类型：实体类、控制类和接口类。其中，实体类映射需求中的每个实体，实体类保存需要存储在永久存储体中的信息，主要负责持久化数据的存储；控制类是用于控制用例工作的类，一般是由动宾结构的短语转化来的名词，主要负责业务逻辑的处理；接口类用于封装在用例内、外流动的信息或数据流，主要负责系统与用户之间的交互。在表 3-1 中， C1（会员）、 C6（订单表）、 C9（邮箱地址）、 C12（支付方式）主要用来存储信息，所以属于实体类； C3（计算总价）、 C5（调用支付系统）、 C8（发送完整订单信息）主要用来处理业务逻辑，所以属于控制类； C2（列出电动玩具清单及价格）、 C4（显示地址和联系方式）、 C7（显示完整订单信息）、 CI0（显示清单和定制属性）、 C11（显示默认支付方式）主要用来与用户交互，所以属于接口类。

X1 ：收货地址

X2 ：支付方式

X3 ：邮箱地址

X4 ：定制属性

根据创建新订单的用例描述，所设计的系统部分类图如下图所示。所以 X1　X4 处分别填入的类名：收货地址、支付方式、邮箱地址和定制属性。

S1 ：挂起

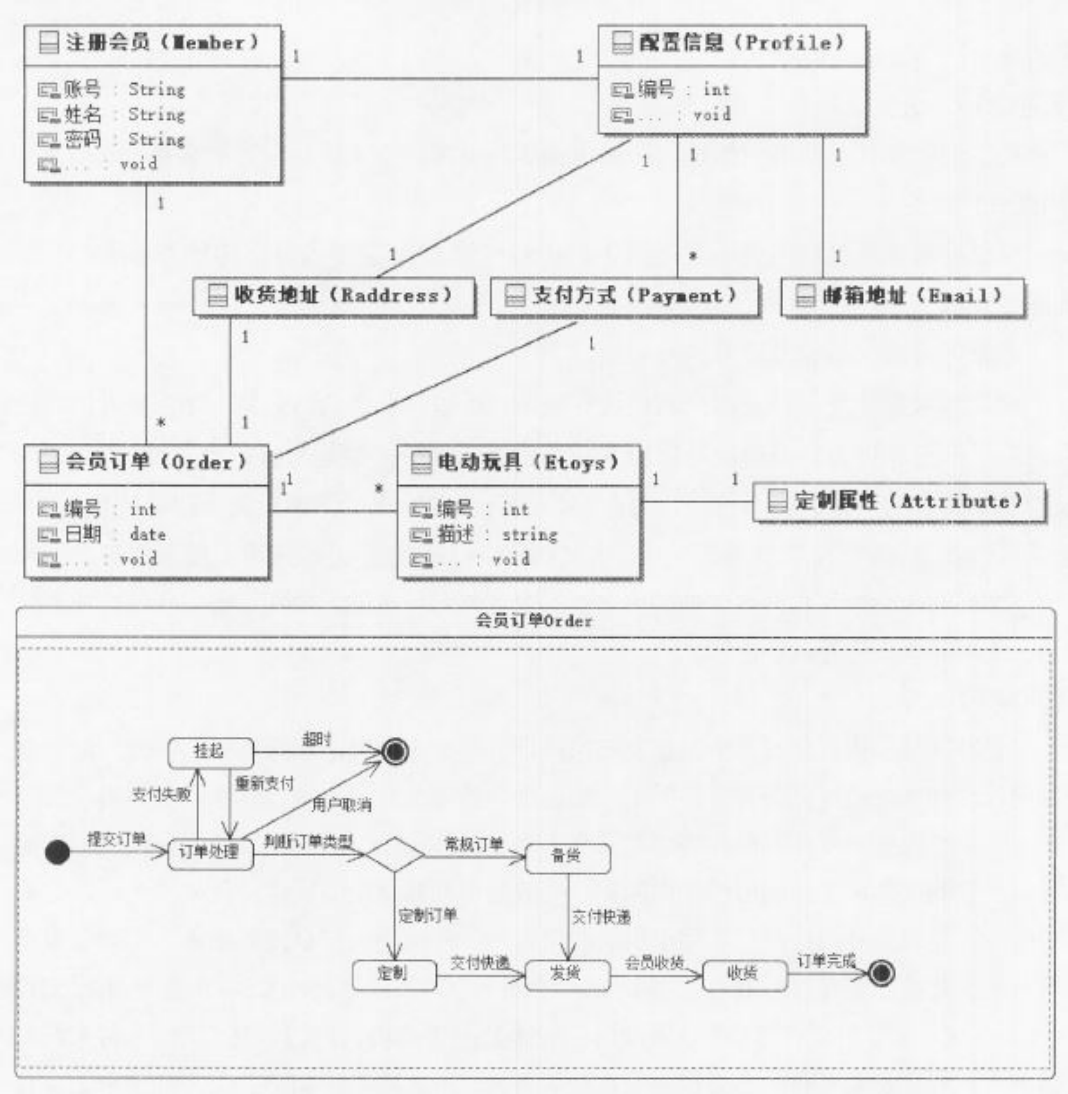
S2 ：备货

S3 ：定制

S4 ：发货

S5 ：收货

根据订单处理过程的描述，所设计的系统状态图如下图所示。所以 S1　S5 处分别填入的状态名：挂起、备货、定制、发货和收货。



试题四 答案：　解析：　(1) first+(last-first) / 2+1 或(last+first) / 2+1 或(last+first+1) / 2 或等价形式

(2) firstSum

(3) first+(last-first) /2 或(last+first) /2 或等价形式

本题考查算法设计与分析的相关知识。

此类题目要求考生掌握常见的算法设计策略，包括分治法、动态规划、贪心法、回溯法和分支限界法，需掌握这些算法设计策略求解问题的特点，并熟悉一些典型的实例。 在做题过程中，认真阅读题目对问题和求解方法的描述。

题干已经描述了假币问题的求解算法的基本思路。这个题目应该比较简单。已经知道假币的重量较轻。

C 代码中，空(1) 所在的代码块是在偶数块硬币的前提下，计算前半部分硬币的重量，因此该空格应填 first+(last-first)/2+1 。空(2) 是判断前半部分 firstSum 轻还是后半部分 lastSum 轻，从而确定是在前半部分还是后半部分递归调用，因此该空格应填 firstSum

(4)分治法

(5) O(lgn)

这是一个典型的分治算法，算法时间复杂度为 T(n)=T(n/2)+0(1) =0(lgn)。

(6) 2

(7) 4

对于 30 枚硬币的情况，首先需要分成两个 15 枚比较轻重(1 次比较)。

最好的情况是，再进行一次比较时，前 7 枚和后 7 枚一样重(1 次比较)，此时假币在中间，即第 8 枚。因此，最少经过 2 次比较即可确定假币。

最坏的情况是，前 7 枚和后 7 枚不一样重，假设前 7 枚更轻(1 次比较)。此时取出前 7 枚硬币，继续比较前 3 枚和后 3 枚，还是不一样重，假设前 3 枚更轻(1 次比较)。 剩下 3 枚硬币，再经过一次比较即可确定假币(1 次比较)。因此最多经过 4 次比较即可确定假币。

试题五 答案：　解析： (1) virtual void buildParts() =0

(2) this->pizzaBuilder=pizzaBuilder

(3) pizzaBuilder->buildParts()

(4) waiter->setPizzaBuilder(hawaiian_pizzabuilder)

(5) waiter->construct()

本题考查生成器(Builder)模式的基本概念和应用。

生成器模式是创建型设计模式中的一种。创建型模式抽象了实例化过程，帮助一个系统独立于如何创建、组合和表示它的那些对象。一个类创建型模式使用继承改变被实例化的类，而一个对象创建型模式将实例化委托给另一个对象。

生成器模式是对象创建型模式，其意图是将一个复杂对象的构建与其表示分离，使得同样的构建过程可以创建不同的表示。此模式的结构图如下图所示。

其中：

• Builder 为创建一个 Product 对象的各个部件指定抽象接口。

• ConcreteBuilder 实现 Builder 的接口以构造和装配该产品的各个部件，定义并明确它所创建的表示，提供一个检索产品的接口。

• Director 构造一个使用 Builder 接口的对象。

• Product 表示被构造的复杂对象。 ConcreteBuilder 创建该产品的内部表不并定义它的装配过程。包含定义组成组件的类，包括将这些组件装配成最终产品的接口。客户以及 Builder 和 Director 的交互过程如下图所示。

本题中图 5-1 类图中的类与上图中类之间存在着对应关系。 Waiter 与 Director 对应；PizzaBuilder 与 Builder 对应； HawaiianPizzaBuilder 和 SpicyPizzaBuilder 与 ConcreteBuilder 对应，均作为 PizzaBuilder 的子类，并且，由于本题中检索产品的接口简单一致，所以也定义在父类 PizzaBuilder 中，而没有分别定义 HawaiianPizzaBuilder 和 SpicyPizzaBuilder 中； Pizza 与 Product 对应。其对应的协作为，客户创建 Waiter 对象并用想要的具体 PizzaBuilder 对象配置 Waiter 中定义的 PizzaBuilder 对象。其中，PizzaBuilder 为创建一个 Pizza 对象的各个部件指定抽象接口，由具体的构建器子类实现。然后使 PizzaBuilder 进行 Pizza 对象的构建。

在 PizzaBuilder 中定义创建 Pizza 对象的各个部件制定抽象接口，由于该操作的具体实现在子类 HawaiianPizzaBuilder 和 SpicyPizzaBuilder 中，所以此处定义为纯虚函数，即：

```
virtual void buildParts () =0;
```

客户(main() 函数)创建 Waiter 对象，并用 PizzaBuilder 对象进行配置，即在 main() 函数中创建一个 PizzaBuilder 的具体子类的对象并进行 Builder 的配置，即：

```
PizzaBuilder* hawaiian_pizzabuilder=new HawaiianPizzaBuilder() ;
waiter->setPizzaBuilder(hawaiian_pizzabuilder);
```

然后开始 Pizza 的构建，即:

```
waiter->construct() ;
```

上图协作中最后的获取结果在本题中用输出表示，因为创建了 HawaiianPizzaBuilder 对象，所以输出为 cross+mild+ham&pineapple 。

在 Waiter 中，定义函数 setPizzaBuilder() 和 construct() 。其中， setPizzaBuilder() 使用客户提供的具体 PizzaBuilder 对象来设置 PizzaBuilder 对象，其引用名称为 pizzaBuilder ，对象中属性的名称和方法参数的名称相同时，采用 this 关键字加以区分，

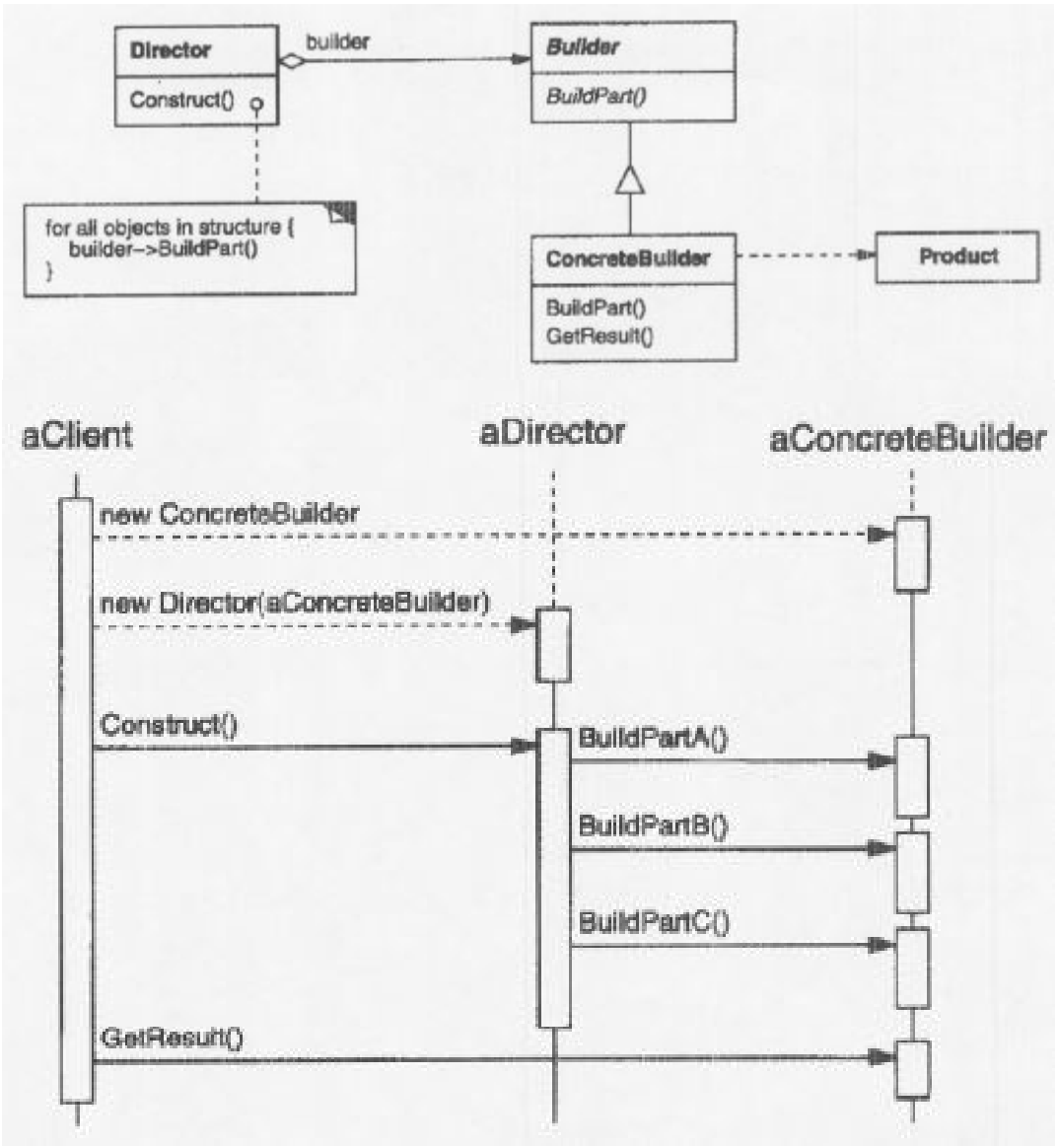
即：

```
voidsetPizzaBuilder (PizzaBuilder* pizzaBuilder) { /*设置构建器*/
this->pizzaBuilder=pizzaBuilder;
}
```

construct() 函数使用所设置的 PizzaBuilder 对象创建 Pizza 及其部件，即：

```
voidconstruct () { /*构建*/
pizzaBuilder->createNewPizza() ;
pizzaBuilder->buildParts() ;
}
```

综上所述，空(1) 为定义创建 Pizza 的对象的各个部件制定抽象接口，即纯虚函数的定义：virtualvoidbuildParts() =0 ；空(2)为 PizzaBuilder 对象的设置，即：this->pizzaBuilder=pizzaBuilder ；空(3) 为具体 PizzaBuilder 对象进行 Pizza 的构建，即：pizzaBuilder->buildParts() ；空(4) 和空(5) 为客户程序用具体的 PizzaBuilder 对象设置 Waiter 中的 PizzaBuilder,并通知开始构建，即 waiter->setPizzaBuilder(hawaiian _ pizzabuilder)和 waiter->construct() 。



试题六 答案：　解析： (1) abstract void buildParts()

(2) this.pizzaBuilder=pizzaBuilder

(3) pizzaBuilder.buildParts()

(4) waiter.setPizzaBuilder(hawaiian_pizzabuilder)

(5) waiter.construct()

本题考查生成器(Builder)模式的基本概念和应用。

生成器模式是创建型设计模式中的一种。创建型模式抽象了实例化过程，帮助一个系统独立于如何创建、组合和表示它的那些对象。一个类创建型模式使用继承改变被实例化的

类，而一个对象创建型模式将实例化委托给另一个对象。
生成器模式是对象创建型模式，其意图是将一个复杂对象的构建与其表示分离，使得同样的构建过程可以创建不同的表示。此模式的结构图如下图所示。

其中：
Builder 为创建一个 Product 对象的各个部件指定抽象接口。
ConcreteBuilder 实现 Builder 的接口以构造和装配该产品的各个部件，定义并明确它所创建的表示，提供一个检索产品的接口。
Director 构造一个使用 Builder 接口的对象。
Product 表示被构造的复杂对象。 ConcreteBuilder 创建该产品的内部表不并定义它的装配过程。包含定义组成组件的类，包括将这些组件装配成最终产品的接口。
客户以及 Builder 和 Director 的交互过程如下图所示。

本题中图 6-1 类图中的类与上图中类之间存在着对应关系。 Waiter 与 Director 对应；PizzaBuilder 与 Builder 对应； HawaiianPizzaBuilder 和 SpicyPizzaBuilder 与 ConcreteBuilder 对应，均作为 PizzaBuilder 的子类，并且，由于本题中检索产品的接口简单一致，所以也定义在父类 PizzaBuilder 中，而没有分别定义 HawaiianPizzaBuilder 和 SpicyPizzaBuilder 中； Pizza 与 Product 对应。其对应的协作为，客户创建 Waiter 对象并用想要的具体 PizzaBuilder 对象配置 Waiter 中的定义的 PizzaBuilder 对象。其中，PizzaBuilder 为创建一个 Pizza 对象的各个部件指定抽象接口，由具体的构建器子类实现。然后使 PizzaBuilder 进行 Pizza 对象的构建。
在 PizzaBuilder 中定义创建 Pizza 对象的各个部件制定抽象接口，由于该操作的具体实现在子类 HawaiianPizzaBuilder 和 SpicyPizzaBuilder 中，所以此处定义为抽象方法，即：
public abstract void buildParts() ;
客户(FastFoodOrdering)创建 Waiter 对象，并用 PizzaBuilder 对象进行配置，即在 main () 方法中创建一个 PizzaBuilder 的具体子类的对象并进行 Builder 的配置，即：
PizzaBuilder hawaiian_pizzabuilder=new HawaiianPizzaBuilder() ;
waiter.setPizzaBuilder(hawaiian_pizzabuilder);
然后开始 Pizza 的构建，即：
waiter.construct () ;
上图协作中最后的获取结果在本题中用输出表示，因为创建了 HawaiianPizzaBuilder 对象，所以输出为 cross+mild+ham&pineapple 。
在 Waiter 中，定义方法 setPizzaBuilder() 和 construct() 。其中， setPizzaBuilder() 使用客户提供的具体 PizzaBuilder 对象来设置 PizzaBuilder 对象，其引用名称为

pizzaBuilder，对象中属性的名称和方法参数的名称相同时，采用 this 关键字加以区分，即：

```
voidsetPizzaBuilder (PizzaBuilder* pizzaBuilder) { /*设置构建器*/
this.pizzaBuilder=pizzaBuilder;
}
```

construct() 方法使用所设置的 PizzaBuilder 对象创建 Pizza 及其部件，即：

```
voidconstruct () { /*构建*/
pizzaBuilder.createNewPizza() ;
pizzaBuilder.buildParts() ;
}
```

综上所述，空(1) 为定义创建 Pizza 的对象的各个部件制定抽象接口，即抽象方法的定义：abstractvoidbuildParts() ;空(2) 为 PizzaBuilder 对象的设置，即： this.pizzaBuilder=pizzaBuilder ；空(3) 为具体 PizzaBuilder 对象进行 Pizza 的构建，即：pizzaBuilder.buildParts() ；空(4) 为客户程序用具体的 PizzaBuilder 对象设置 Waiter 中的 PizzaBuilder,即： waiter.setPizzaBuilder(hawaiianjpizzabuilder),空(5) 通知开始构建，即： waiter. construct() 。