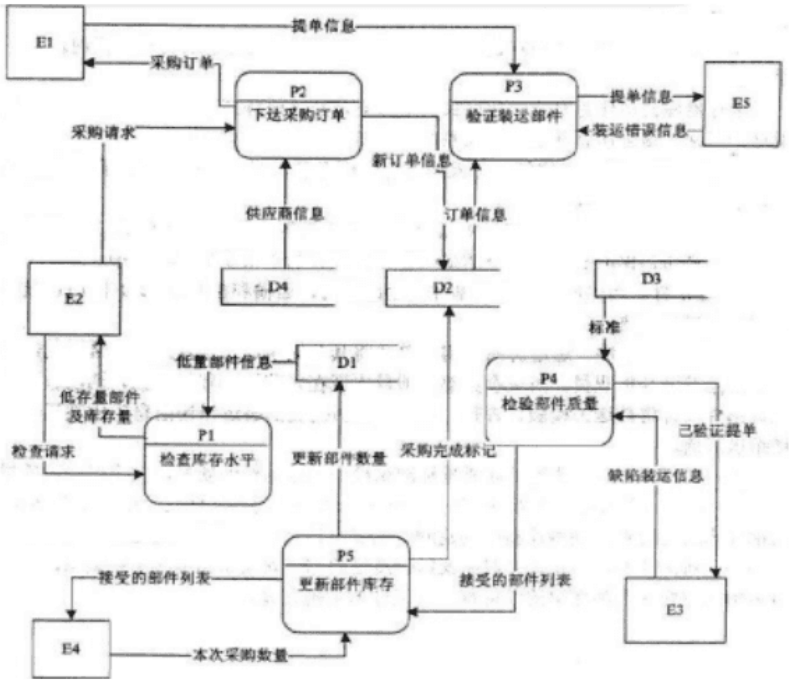


1-2 0 层数据流图

1 使用说明中的词语，给出图 1-1 中的实体 E1~E5（5 分）

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。（4 分）

3 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。（4 分）

4 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（2 分）

试题二（15）

1 至问题 3,将解答填入答题纸的对应栏内。

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(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)。

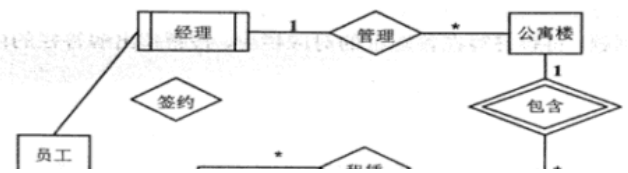
3、对于每个租客，系统需记录姓名、多个联系电话、一个银行账号（方便自动扣房租)、一个紧急联系人的姓名及联系电话。

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，比如：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。



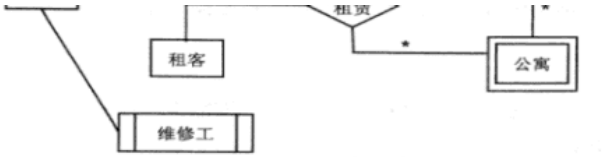


图 2-1 实体联系图

【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):

联系电话(电话号码，租客编号)

租客（租客编号，姓名，银行账户，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，（c），押金）

1 补充图 2-1 中的“签约”联系所关联的实体及联系类型。（4.5 分）

2 补充逻辑结构设计中的（a）、（b）、（c）三处空缺。（4.5 分）

3 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。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 （15 分）

1 至问题 3，将解答填入答题纸的对应栏内。

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到端的玩具定制和销售服务。在系统设计阶段，"创建新订单 (New Order)" 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。

表 3-1 创建新订单 (NewOrder) 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
	1.会员（C1）点击“新的订单”按钮； 2.系统列出所有正在销售的电动玩具清单及价格（C2）

典型事件流	3.会员点击复选框选择所需电动玩具并输入对应数量，点击“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的收货地址和支付方式（C4）； 5.会员点击“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 C6）中； 8.系统将以表格形式显示完整订单信息（C7），同（C8）至会员预先配置的邮 C9）。	
候选事件流	3a	（1）会员点击“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）会员点击单选按钮选择所需要定制的电动点击“结算” 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）若会员点击“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 若会员点击“取消订单”，则该用例终止执

3-2 候选设计类分类

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

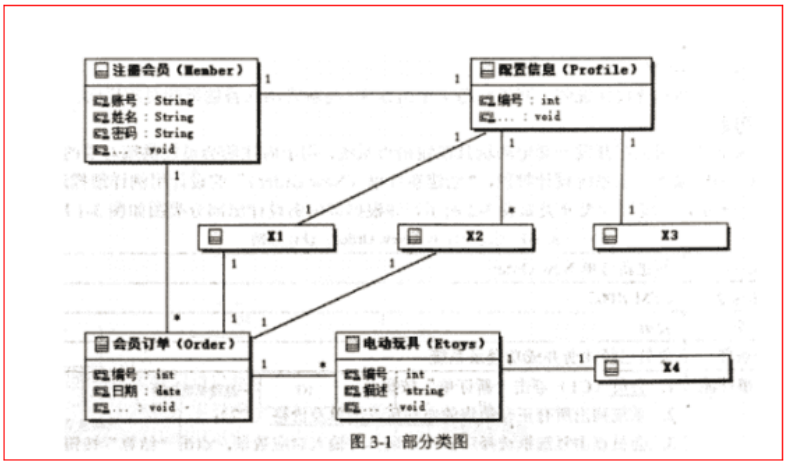


图 3-1 部分类图

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。

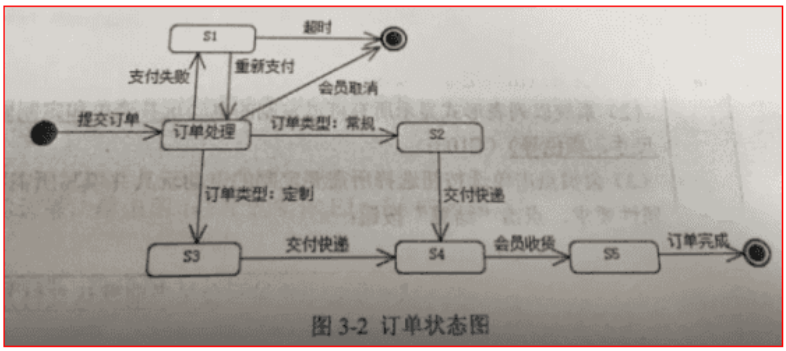


图 3-2 订单状态图

1 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 3-2 中的（a）、（b）和（c）处。(6 分)

2 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。（4 分）

3 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。（5 分）

试题四：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。

假币问题 n 枚硬币，其中有一枚是假币，已知假币的重量较轻。现现只有一个天平，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 )
        }
    }
    }
```

1 根据题干说明，填充 C 代码中的空（1）-（3）

2 根据题干说明和 C 代码，算法采用了（）设计策略。

函数 getCounterfeitCoin 的时间复杂度为（）（用 O 表示）。

3 若输入的硬币数为 30，则最少的比较次数为（），最多的比较次数为（）。

试题五(共 15 分) (请从试题五、试题六中选答一题)

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。

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。

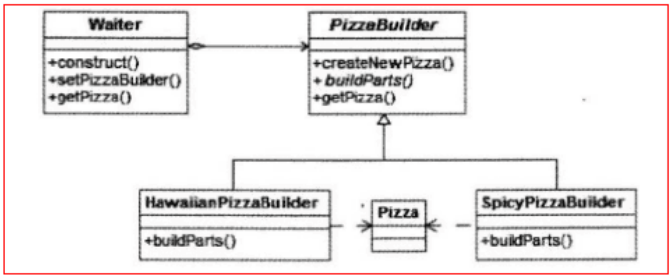


图 5-1 1 类图

【C++代码】

```
#include<iostream>
#include <string>
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:
        void setPizzaBuilder(PizzaBuilder* pizzaBuilder) { /*设置构建器*/
    ( 2 )
}
Pizza* getPizza() { return pizzaBuilder->getPizza(); }
    void construct() { /*构建*/
        pizzaBuilder->createNewPizza();
        ( 3 )
    }
};
int main(){
    Waiter*waiter=new Waiter();
    PizzaBuilder*hawaiian pizzabuilder=new
    HawaiianPizzaBuilder()
    ( 4 );
    ( 5 );
    cout<< "pizza: "<< waiter->getPizza()->getParts()<< endl;
}
```

程序的输出结果为:

```
pizza: cross + mild + ham&pineapple
```

试题六(共 15 分)

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。

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。

```
    public Pizza getPizza() { return pizzaBuilder.getPizza(); }
    public void construct() { /*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

试题一答案解析：

1：E1 供应商　E2 采购部门　E3 检验员　E4 库管员　E5 职员

2：　D1 库存表 D2　　采购订单表　D3 质量标准表　D4 供应商表

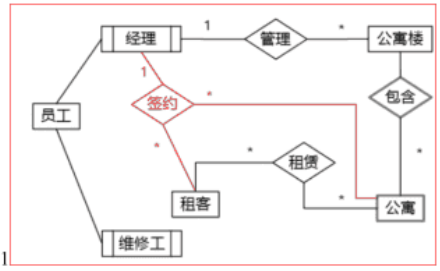
3：　P3（验证装运部件）-----E1（客户）　　P4（校验部件质量）-----E1（客户）

　　P3（验证装运部件）------ P4（校验部件质量）　　P1（检查库存水平）-----D1（库存表）

4：父图中某个加工的输入输出数据流必须与其子图的输入输出数据流在数量上和内容上保持一致，即数据不会凭空产生，也不能凭空消失。父图的一个输入（或输出）数据流应对应子图中几个输入（或输出）数据流，而子图中组成的这些数据流的数据项全体正好是父图中的这个数据流。

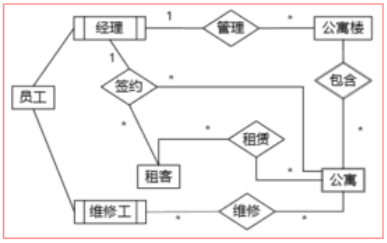
试题二答案解析：

1



2　a）业务技能 b）楼编号 c）月租金

3



新增维修关系，维修工维修公寓，关系模式为维修情况
维修情况（故障编号，员工编号，楼编号，公寓号，维修日期，维修内容）

试题三答案解析：
a）：C4、C5、C7、C8、C10、C11
b）：C3
c）：C1、C2、C6、C9、C12

一、实体类

实体类是用于对必须存储的信息和相关行为建模的类。 实体对象（实体类的实例）用于保存和更新一些现象的有关信息， 例如： 事件、 人员或者一些现实生活中的对象。实体类通常都是永久性的， 它们所具有的属性和关系是长期需要的， 有时甚至在系统的整个生存期都需要。

二、边界类

边界类是系统内部与系统外部的业务主角之间进行交互建模的类。 边界类依赖于系统外部的环境， 比如业务主角的操作习惯、 外部的条件的限制等。 它或者是系统为业务主角操作提供的一个 GUI， 或者系统与其他的系统之间进行一个交互的接口，所以当外部的 GUI 变化时，或者是通信协议有变化时，只需要修改边界类就可以了， 不用再去修改控制类和实体类。 业务主角通过它来与控制对象交互，实现用例的任务。边界类调用用例内的控制类对象，进行相关的操作。一个系统可能会有多种边界类：
用户界面类 - 帮助与系统用户进行通信的类
系统接口类 - 帮助与其他系统进行通信的类
设备接口类 - 为用来监测外部事件的设备（如传感器）提供接口的类

三、控制类

控制类用于对一个或几个用例所特有的控制行为进行建模， 它描述的用例的业务逻辑的实现， 控制类的设计与用例实现有着很大的关系。 在有些情况下， 一个用例可能对应多个控制类对象， 或在一个控制类对象中对应着对个用例。 它们之间没有固定的对应关系， 而是根据具体情况进行分析判断， 控制类有效将业务逻辑独立于实体数据和边界控制， 专注于处理业务逻辑， 控制类会将特有的操作和实体类分离， 者有利于实体类的统一化和提高复用性。 当业务主角通过边界类来执行用例的时候， 产生一个控制类对象，在用例被执行完后， 控制类对象会被销毁。控制类的特点： 独立于环境、 和用例的实现关联、 使用关联实体类或操作实体类对象、 专注于业务逻辑的实现。 当然如果用例的逻辑较为简单， 可以直接利用边界类来操作实体类， 而不必再使用控制类。 或者用例的逻辑较为固定， 业务逻辑固定不会改变。也可以直接在边界类实现该逻辑。

2 X1：收货地址 X2：支付方式 X3：邮箱地址 X4：电动玩具定制属性
3 S1：订单挂起 S2：订单备货 S3：订单定制 S4：订单发货 S5：订单收货

试题四答案

1. 1）first+(last-first)/2 或(first+last)/2 （2）firstSum<lastSum 3）first+(last-first)/2 或(first+last)/2

2 .4）分治法 5）O（nlogn）

3.6）2 （7）4

试题分析：若输入 30 个硬币，找假硬币的比较过程为：

第 1 次： 15 比 15，此时能发现假币在 15 个的范围内。

第 2 次： 7 比 7，此时，如果天平两端重量相同，则中间的硬币为假币，此时可找到假币，这是最理想的状态。

第 3 次： 3 比 3，此时若平衡，则能找出假币，不平衡，则能确定假币为 3 个中的 1 个。

第 4 次： 1 比 1， 到这一步无论是否平衡都能找出假币， 此时为最多比较次数。

试题五答案

```
1 virtual void buildParts()
2 this->pizzaBuilder=pizzaBuilder
3 pizzaBuilder->builderParts()
4 waiter->setPizzaBuilder(hawaiian_pizzabuilder)
5 waiter->construct()
```

试题六答案

```
1）abstract void buildParts();
2）this.pizzaBuilder=pizzaBuilder
3）pizzaBuilder.buildParts()
4）waiter.setPizzaBuilder(hawaiian_pizzabuilder)
5）waiter.construct()
```