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 2014 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 （共 15 分）

阅读下列说明和图，回答问题 1 至问题 4，将解答填入答题纸的对应栏内。

【说明】

某巴士维修连锁公司欲开发巴士维修系统，以维护与维修相关的信息。该系统的主要功能如下：

1)记录巴士 ID 和维修问题。巴士到车库进行维修，系统将巴士基本信息和 ID 记录在巴士列表文件中，将待维修机械问题记录在维修记录文件中，并生成维修订单。

2)确定所需部件。根据维修订单确定维修所需部件，并在部件清单中进行标记。

3)完成维修。机械师根据维修记录文件中的待维修机械问题，完成对巴士的维修，登记维修情况；将机械问题维修情况记录在维修记录文件中，将所用部件记录在部件清单中，并将所用部件清单发送给库存管理系统以对部件使用情况进行监控。巴士司机可查看已维修机械问题。

4)记录维修工时。将机械师提供的维修工时记录在人事档案中；将维修总结发送给主管进行绩效考核。

5)计算维修总成本。计算部件清单中实际所用部件、人事档案中所用维修工时的总成本；将维修工时和所用部件成本详细信息给会计进行计费。 现采用结构化方法对巴士维修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。

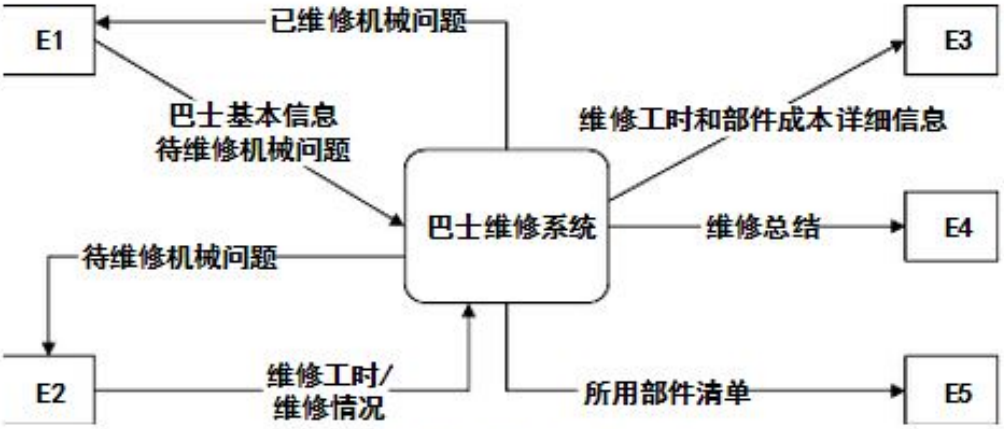


图 1-1 上下文数据流图

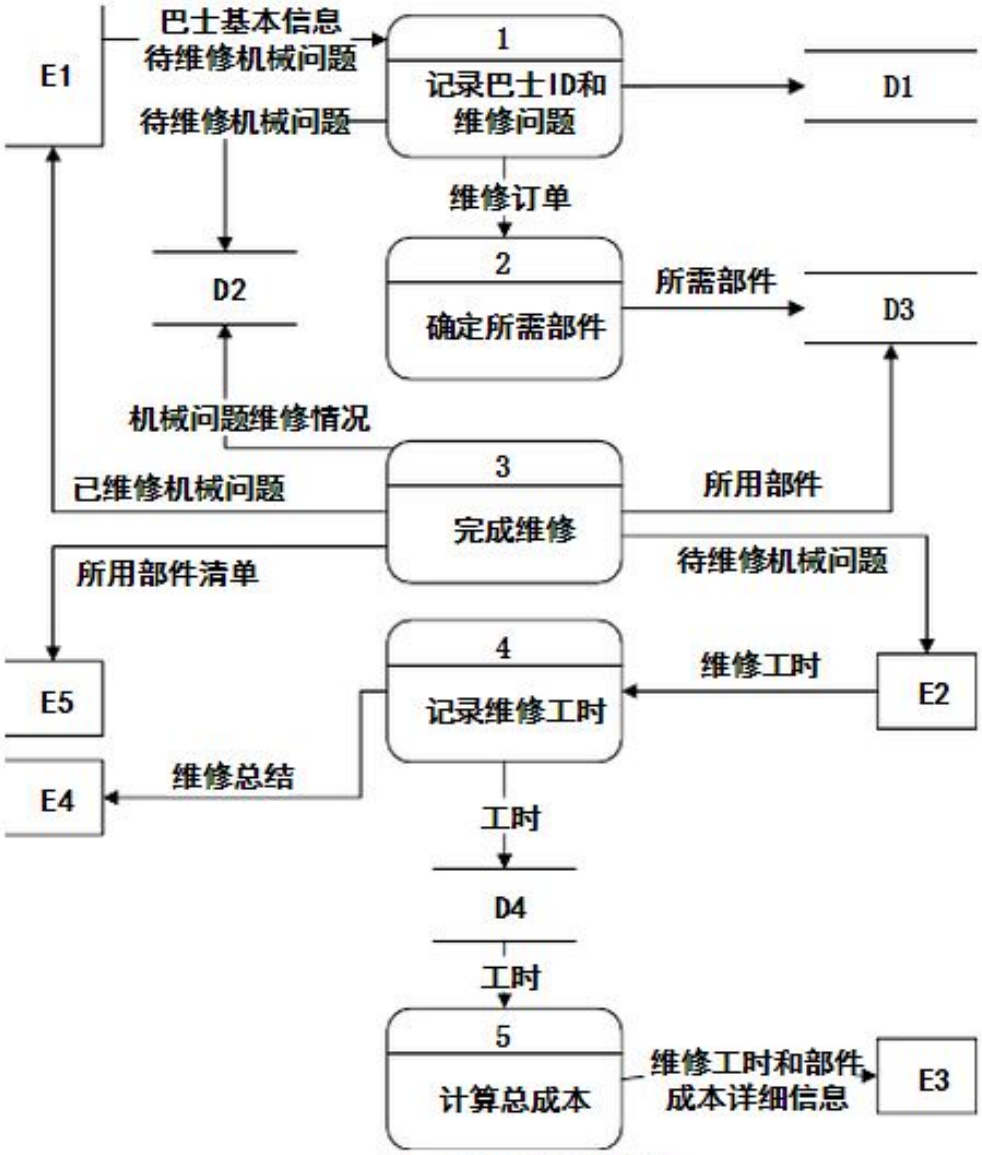


图 1-2 0 层数据流图

问题：1.1
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
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
说明图 1-2 中所存在的问题。
问题：1.4
根据说明和图中术语，采用补充数据流的方式，改正图 1-2 中的问题。要求给出所补充数据流的名称、起点和终点。

试题二 （共 15 分）

阅读下列说明和图，回答问题 1 至问题 3, 将解答填入答题纸的对应栏内。

【说明】

某家电销售电子商务公司拟开发一套信息管理系统，以方便对公司的员工、家电销售、家电厂商和客户等进行管理。

【需求分析】（1）系统需要维护电子商务公司的员工信息、客户信息、家电信息和家电厂商信息等。员工信息主要包括：工号、姓名、性别、岗位、身份证号、电话、住址，其中岗位包括部门经理和客服等。客户信息主要包括：客户 ID 、姓名、身份证号、电话、住址、账户余额。家电信息主要包括：家电条码、家电名称、价格、出厂日期、所属厂商。家电厂商信息包括：厂商 ID 、厂商名称、电话、法人代表信息、厂址。（2）电子商务公司根据销售情况，由部门经理向家电厂商订购各类家电。每个家电厂商只能由一名部门经理负责。（3）客户通过浏览电子商务公司网站查询家电信息，与客服沟通获得优惠后，在线购买。

【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)：

客户(客户 ID 、姓名、身份证号、电话、住址、账户余额)

员工(工号、姓名、性别、岗位、身份证号、电话、住址)

家电(家电条码、家电名称、价格、出厂日期、 (1))

家电厂商(厂商 ID、厂商名称、电话、法人代表信息、厂址、 (2))

购买(订购单号、 (3) 、金额)

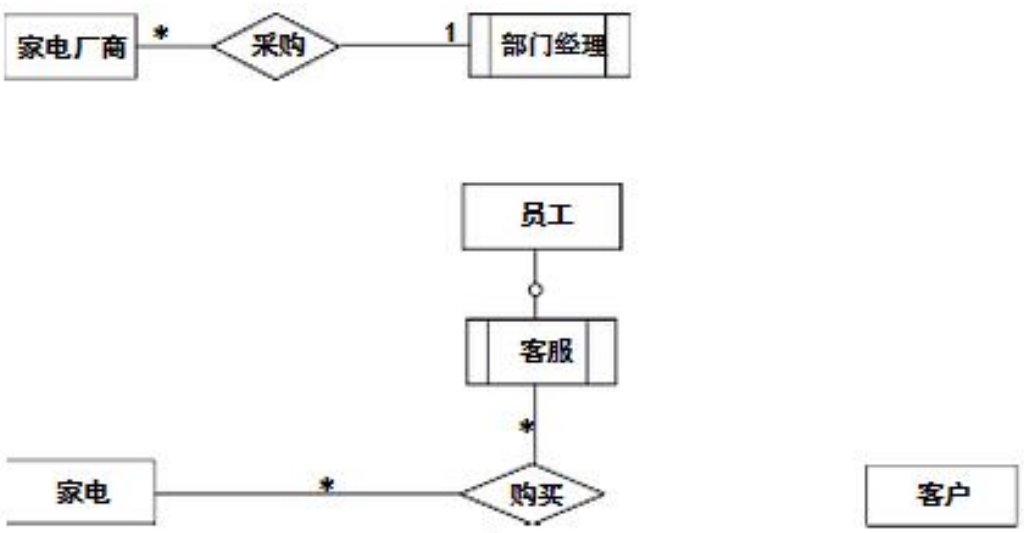


图 2-1 实体联系图

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)～(3)补充完整。用下划线指出“家电”、“家电厂商”和“购买”关系模式的主键。

问题： 2.3

电子商务公司的主营业务是销售各类家电，对账户有余额的客户，还可以联合第三方基金公司提供理财服务，为此设立客户经理岗位。客户通过电子商务公司的客户经理和基金公司的基金经理进行理财。每名客户只有一名客户经理和一名基金经理负责，客户经理和基金经理均可负责多名客户。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 （共 15 分）

阅读下列说明和图，回答问题 1 至问题 3, 将解答填入答题纸的对应栏内。

【说明】

某高校图书馆欲建设一个图书馆管理系统，目前已经完成了需求分析阶段的工作。功能需求均使用用例进行描述，其中用例“借书(CheckOutBooks)”的详细描述如下。

参与者：读者(Patron)。

典型事件流：

1．输入读者 ID；

2．确认该读者能够借阅图书，并记录读者 ID；

3．输入所要借阅的图书 ID；

4．根据图书目录中的图书 ID 确认该书可以借阅，计算归还时间，生成借阅记录；

5．通知读者图书归还时间。

重复步骤 3-5，直到读者结束借阅图书。

备选事件流：

2a. 若读者不能借阅图书，说明读者违反了图书馆的借书制度(例如，没有支付借书费用等)

①告知读者不能借阅，并说明拒绝借阅的原因；

②本用例结束。

4a. 读者要借阅的书无法外借

①告知读者本书无法借阅；

②回到步骤 3 。

说明：图书的归还时间与读者的身份有关。如果读者是教师，图书可以借阅一年；如果是学生，则只能借阅 3 个月。读者 ID 中包含读者身份信息。

现采用面向对象方法开发该系统，得到如图 3-1 所示的系统类模型(部分)；以及如图 3-2 所示的系统操作“check0ut(bookID)(借书)”的通信图(或协作图)。

问题： 3.1
根据说明中的描述，以及图 3-1 和图 3-2 ，给出图 3-1 中 C1 ～ C4 处所对应的类名(类名使用图 3-1 和图 3-2 中给出的英文词汇)。

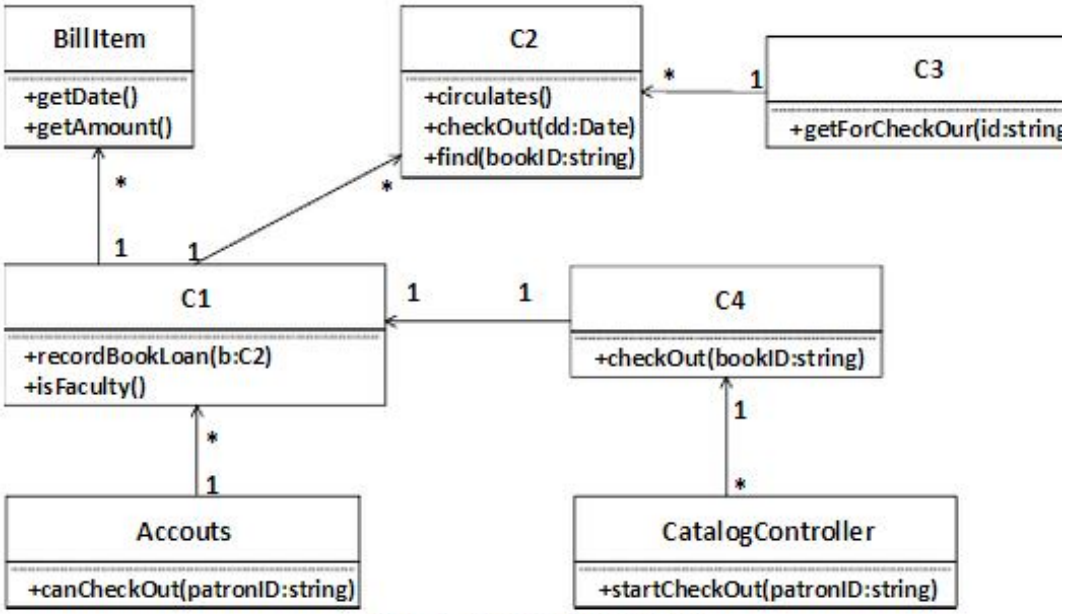


图 3-1 系统类模型（部分）

图 3-1 系统类模型（部分）

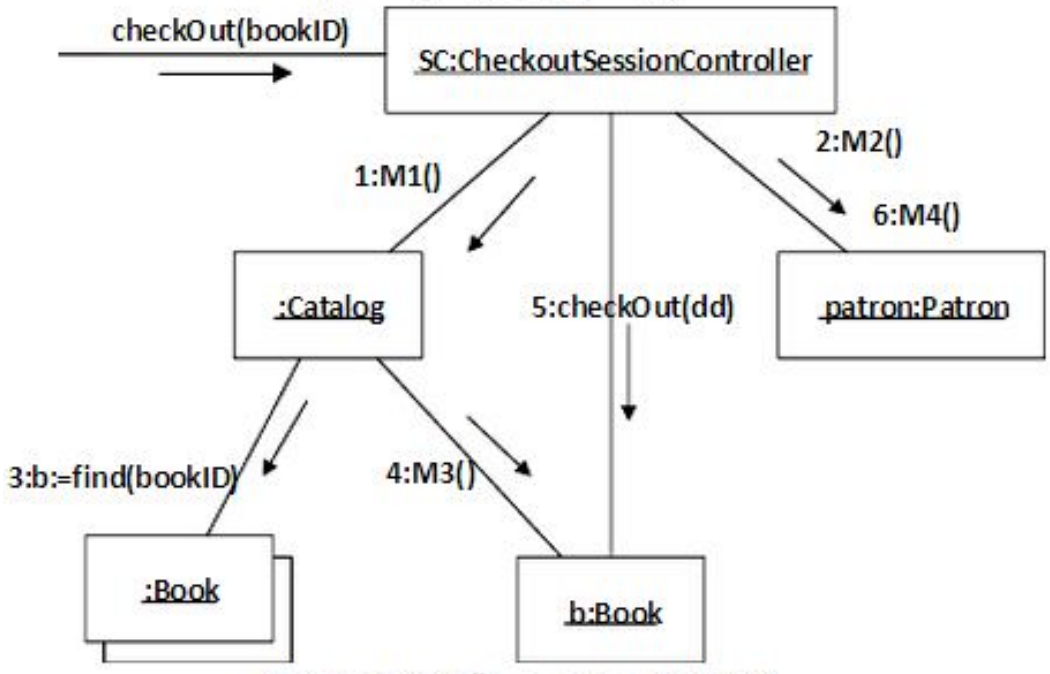


图 3-2 系统操作 checkOut 的通信图

问题： 3.2

根据说明中的描述，以及图 3-1 和图 3-2，给出图 3-2 中 M1 ～ M4 处所对应的方法名(方法名使用图 3-1 和图 3-2 中给出的英文词汇)。

问题： 3.3

用例“借书”的备选事件流 4a 中，根据借书制度来判定读者能否借阅图书。若图书馆的借书制度会不断地扩充，并需要根据图书馆的实际运行情况来调整具体使用哪些制度。为满足这一要求，在原有类设计的基础上，可以采用何种设计模式？简要说明原因。

试题四 （共 15 分）

阅读下列说明和 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。

【说明】

采用归并排序对 n 个元素进行递增排序时，首先将 n 个元素的数组分成各含 n/2 个元素的两个子数组，然后用归并排序对两个子数组进行递归排序，最后合并两个已经排好序的子数组得到排序结果。 下面的 C 代码是对上述归并算法的实现，其中的常量和变量说明如下：

arr：待排序数组

p， q， r：一个子数组的位置为从 p 到 q，另一个子数组的位置为从 q+l 到 r

begin， end:待排序数组的起止位置

left， right:临时存放待合并的两个子数组

n1， n2:两个子数组的长度

i， j， k：循环变量

mid:临时变量

【C 代码】

```
#include<stdio.h>
#include<stdlib.h>
#define MAX 65536
void merge(int arr[],int p,int q,int r) {
     int *left,*right;
     int nl,n2,i,j,k;
     nl=q-p+1;
     n2=r-q;
     if((left=(int*)malloc((nl+1)*sizeof(int)))==NULL){
        perror("malloc error");
        exit(1);
      }
     if((right=(int*)malloc((n2+1)*sizeof(int)))==NULL){
        perror("malloc error");
         exit(1);
      }
      for(i=0;i<n1;i++){
         left[i] = arr[p+i];
       }
      left[i] = MAX;
      for(i=0;i<n2;i++){
        right[i] = arr[q+i+1];
      }
          right[i]= MAX;
          i=0; j=0;
          for (k=p;__(1)__k++ ) {
           if(left[i]> right[j]{
              __(2)__
            j++;
           }else{
            arr[k]= left[i];
            i++;
          }
     }
  }
  void mergeSort(int arr[],int begin,int end){
      int mid;
      if ( __(3)__ ){
      mid=(begin+ end)/2;
      mergeSort(arr,begin,mid);
      __(4)__
      merge(arr,begin,mid,end);
      }
  }
```

问题： 4.1

根据以上说明和 C 代码，填充 C 代码中的空(1)～(4)。

问题： 4.2

根据题干说明和以上 c 代码，算法采用了 (5) 算法设计策略。

分析时间复杂度时，列出其递归式为（6），解得渐进时间复杂度为（7）（用 0 符号表示）。空间复杂度为(8)（用 0 符号表示）。

问题： 4.3

两个长度分别为 n1 和 n2 的已排好序的子数组进行归并，根据上述 C 代码，则元素之间比较次数为（9）。

试题五 （共 15 分）

【说明】

某实验室欲建立一个实验室环境监测系统，能够显示实验室的温度、湿度以及洁净度等环境数据。当获取到最新的环境测量数据时，显示的环境数据能够更新。 现在采用观察者（Observer）模式来开发该系统。观察者模式的类图如图 5-1 所示。

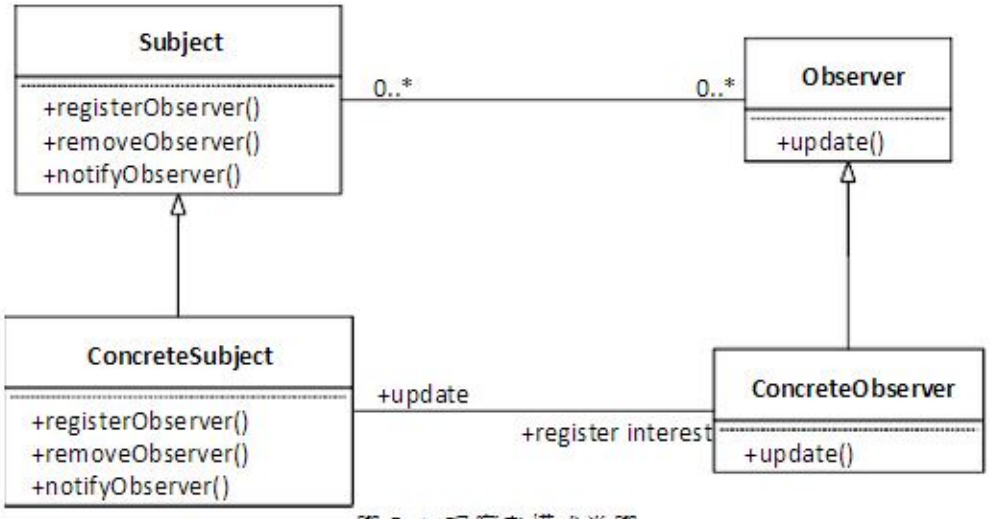


图 5-1 观察者模式类图

【C++代码】

```
#include <iostream>
#include <vector>
using namespace std;
class Observer {
public:
  virtual void update(float temp, float humidity, float cleanness)=0;
};
class Subject {
public:
  virtual void registerObserver(Observer *o)=0;  //注册对主题感兴趣的观察者
  virtual void removeObserver(Observer *o)=0;  //删除观察者
  virtual void notifyObservers()=0; //当主题发生变化时通知观察者
};
class EnvironmentData:public   (1)   {
private:
   vector<Observer *> observers;
    float temperature,humidity,cleanness;
 public:
    void registerObserver(Observer* o) {      observers.push_back(o);   }
    void removeObserver(Observer* o) {   /*代码省略 */  }
    void notifyObservers(){
    for(vector<Observer*>::const_iterator it = observers.begin();it!=observers.e
     {     (2)      }
}
void measurementsChanged(){     (3)      }
void setMeasurements(float temperature,float humidity,float cleanness) {
    this->temperature = temperature;
    this->humidity = humidity;
    this->cleanness = cleanness;
     (4)
```

```
    }
};
class CurrentConditionsDisplay:public   (5)   {
private:
     float temperature,humidity,cleanness;
     Subject* envData;
public:
    CurrentConditionsDisplay(Subject* envData) {
        this->envData = envData;
        (6)
    }
    void update(float temperature,float humidity,float cleanness) {
    this->temperature = temperature;
    this->humidity = humidity;
    this->cleanness = cleanness;
    display();
}
void display() {   /*代码省略 */ }
};
int main() {
    EnvironmentData* envData = new EnvironmentData();
    CurrentConditionsDisplay* currentDisplay = new CurrentConditionsDisplay(envData);
    envData->setMeasurements(80,65,30.4f);
    return 0;
}
```

问题： 5.1

阅读说明和 Java 代码，将应填入（n）处的字句写在答题纸的对应栏内。

试题六 （共 15 分）

【说明】

某实验室欲建立一个实验室环境监测系统，能够显示实验室的温度、湿度以及洁净度等环境数据。当获取到战新的环境测量数据时，显示的环境数据能够更新。

现在采用观察者(Observer)模式来开发该系统。观察者模式的类图如图 6-1 所示。

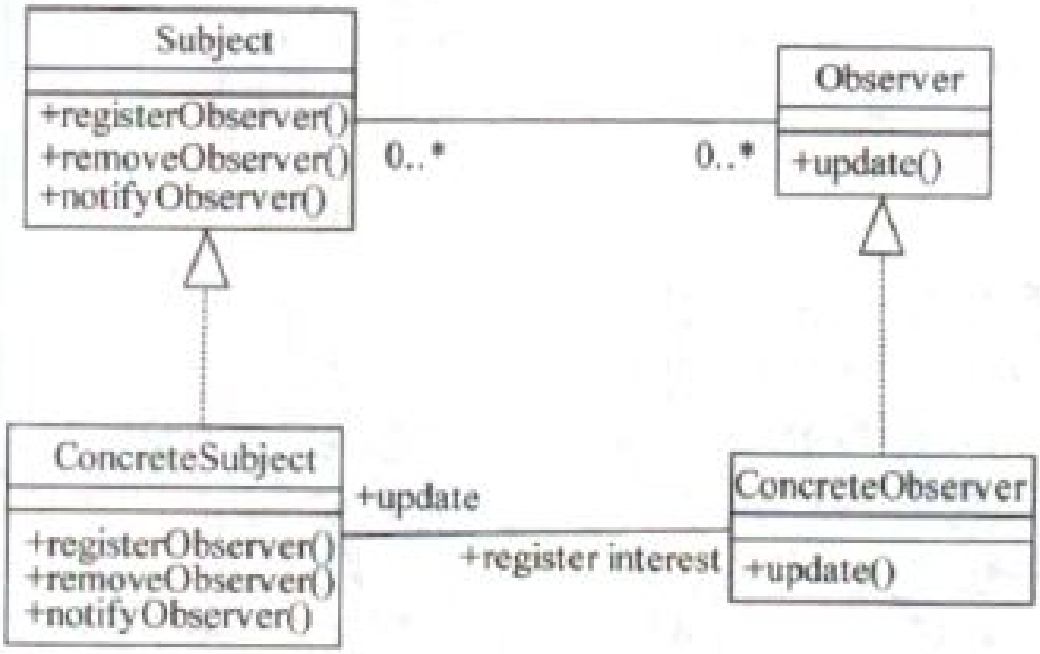


图 6-1 观察者模式类图

【Java 代码】

```
import java.util.*;
interface Observer {
  public void update(float temp, float humidity, float cleanness);
}
interface Subject {
  public void registerObserver(Observer o); //注册对主题感兴趣的观察者
  public void removeObserver(Observer o);   //删除观察者
  public void notifyObservers();            //当主题发生变化时通知观察者
}
class EnvironmentData implements __(1)__{
  private ArrayList observers;
  private float temperature, humidity, cleanness;
  public EnvironmentData() {    observers = new ArrayList(); }
  public void registerObserver(Observer o) {  observers.add(o);  }
  public void removeObserver(Observer o) {  /* 代码省略 */  }
  public void notifyObservers() {
    for (int i = 0; i < observers.size(); i++) {
      Observer observer = (Observer)observers.get(i);
      __(2)__;
    }
  }
```

```
  public void measurementsChanged() { ___(3)___; }
public void setMeasurements(float temperature, float humidity, float
cleanness) {
    this.temperature = temperature;
    this.humidity = humidity;
    this.cleanness = cleanness;
    ___(4)___;
  }
}
class CurrentConditionsDisplay implements ___(5)___{
    private float temperature;
    private float humidity;
    private float cleanness;
    private Subject envData;
    public CurrentConditionsDisplay(Subject envData) {
      this.envData = envData;
      ___(6)___;
}
    public void update(float temperature, float humidity, float cleanness) {
      this.temperature = temperature;
      this.humidity = humidity;
      this.cleanness = cleanness;
      display();
}
    public void display() { /* 代码省略 */ }
}
class EnvironmentMonitor{
  public static void main(String[] args) {
    EnvironmentData envData = new EnvironmentData();
    CurrentConditionsDisplay currentDisplay = new CurrentConditions
    Display(envData);
    envData.setMeasurements(80, 65, 30.4f);
  }
}
```

问题： 6.1

阅读下列说明和 Java 代码，将座填入(n)处的字句写在答题纸的对应栏内。

试题一 答案： 解析：

答案

E1 ： 巴士司机

E2 ： 机械师

E3 ： 会计

E4：主管

E5：库存管理系数

【试题解析】

本题考查的是 DFD 的应用，属于比较传统的题目，考查点也与往年类似。

本问题考查的是顶层 DFD 。顶层 DFD 通常用来确定系统边界，其中只包含一个唯一的加工(即待开发的系统)、外部实体以及外部实体与系统之间的输入输出数据流。题目要求填充的正是外部实体。

从题干说明 1)没有明确说明由巴士到车库后由谁提供待维修问题，图 1-1 中的 E1 ，考察说明中 3)中最后一句说明“巴士司机可查看已维修机械问题”可以看出，从系统到巴士司机有输出数据流“已维修机械问题”，可知 E1 为巴士司机。从 2)中“机械师根据维修记录文件中的待维修机械问题，完成对巴士的维修，登记维修情况”；再看说明 4)中机械师提供维修工时，可以看出，从 E2 到系统有输入数据流“维修工时”、输出数据流“待维修机械问题”，可知 E2 为机械师，还将维修总结发送给主管，即系统到 E4 有输出数据流“维系总结”，可知 E4 为主管。从说明 5)将维修工时和所用部件成本详细信息给会计，从系统到 E3 有输出数据流“维修工时和所用部件成本详细信息”，可知 E3 为会计。说明 3)中将所用部件清单发送给库存管理系统以对部件使用情况进行监控，及系统到 E5 有输出数据流“所用部件清单”，可知 E5 为库存管理系统。

答案

D1：巴士列表文件

D2：维修记录文件

D3：部件清单

D4：人事档案

【试题解析】

本问题考查 0 层数据流图中的数据存储。系统中的主要功能与图 1-2 中的处理一一对应，1)对应处理“ 记录巴士 ID 和维修问题”，将巴士 ID 记录在巴士列表文件中，可知 D1 为巴士列表文件。说明 2)对应处理 “ 确定所需部件 ”，将维修所需部件在部件清单中进行标记，可知以 D3 为部件清单。说明 1)中将待维修机械问题记录在维修记录文件中，可知 D2 为维修记录文件。说明 4)对应处理“ 记录维修工时“，描述了将机械师提供的维修工时记录在人事档案中，可以判定 D4 是人事档案。

答案

图 1-2 中处理 3 只有输出数据流，没有输入数据流。 D2 和 D3 是黑洞，只有输入的数据流，没有输出的数据流。父图与子图不平衡，图 1-2 中没有图 1-1 中的数据流“维 修情况”。

【试题分析】

本问题考查 0 层数据流图中的数据流。分析图 1-2, 可以发现，处理 3 只有输出数据流没有输入数据流， D2 和 D3 只有输入数据流，而没有输出流，造成黑洞。另外，对 照图 1-2 和图 1-1 ，发现图 1-1 中从 E2 输入的数据流维修工时/维修情况，在图 1-2 中只有维修工时，造成父图与子图不平衡。

答案

【试题分析】

针对【问题 3 】分析图 1-2 中存在的问题，题目要求以补充数据流的方式解决，进一步分析说明，说明 3)对应处理“完成维修”，机械师根据维修记录文件中的待维修机械问题完成对巴士的维修，可知处理完成维修需要从维修记录文件读取待维修问题，补充一条从 D2 到处理 3 的数据流“待维修机械问题”。说明 5)对应处理“计算维修总成本”，需要计算部件清单中实际所用部件，补充从部件清单到计算总成本的数据流“实际所用部件”。说明 3)中机械师要登记维修情况，判定图 1-2 中缺少了 E2 到处理 3 的数据流“维修情况”。到此为止所有缺失的数据流都补齐了，也解决了【问题 3 】中的平衡问题、处理只有输出数据流没有输入数据流的问题， D2 和 D3 也既有输入数据流，又有输出数据流。

数据流名称	起　点	终　点
待维修机械问题	D2 或维修记录文件	3 或完成维修
实际所用部件	D3 或部件清单	5 或计算总成本
维修情况	E2 或机械师	3 或完成维修

试题二 答案： 解析：

答案

【试题分析】

本题考查数据库设汁，属于比较传统的题目，考查点也与往年类似。

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，一个家电厂商可以供应多台家电，而一台家电只能对应一个家电厂商，因此“家电厂商”和“家电”之间存在“供应”联系，联系的类型为一对多(1:*，或 l:m)。

根据题目的需求描述可知，“员工”和“部门经理”之间存在一个包含关系。
根据题目的需求描述可知，“客户”、“客服”和“家电’’之间存在“购买”联系，联系的类型为多对多对多(*:*:*，或 m:n:o)。

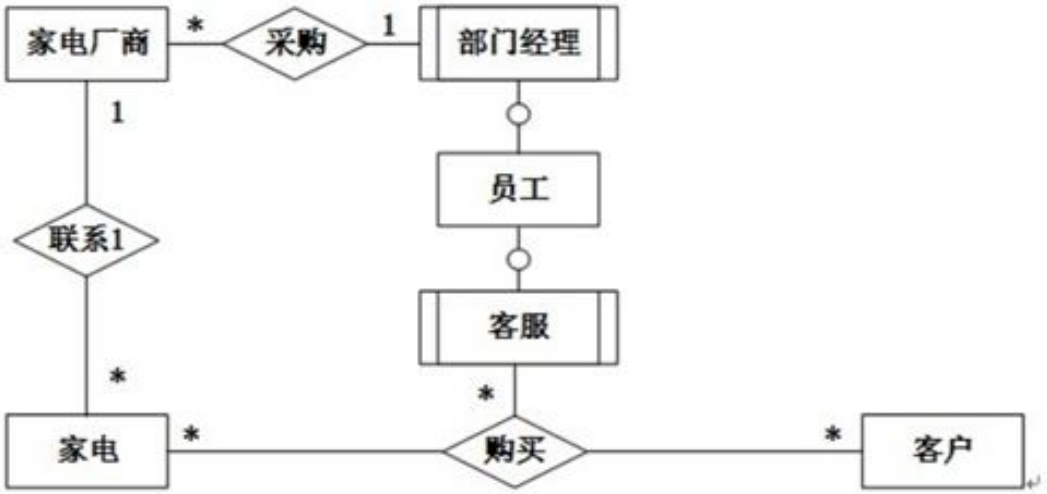
答案

(1）厂商 ID
(2）部门经理工号 或 经理工号 或 员工工号
(3）家电条码，客户 ID，客服工号

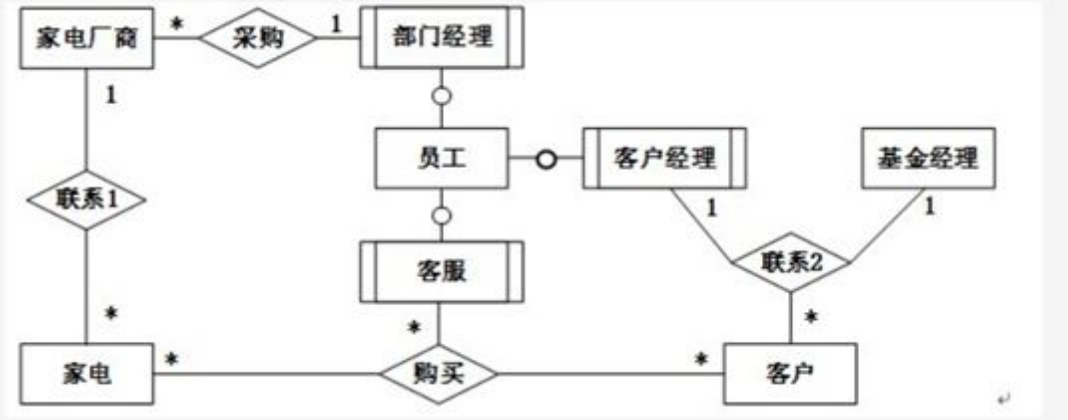
【试题分析】
本问题考查数据库的逻辑结构设计，题目要求补充完整各关系模式，并给出各关系模式的主键。
根据实体联系图和需求描述，“家电”和“家电厂商”存在多对一的关系，在家电关系中尚要记录家电厂商的主键，也就是“厂商 ID”。所以，对于“家电”关系模式，需补充属性“厂商 ID”。“家电条码”为“家电”关系的主键。
根据实体联系图和需求描述，“家电厂商”和“部门经理”之间存在多对一的关系，在家电厂商关系中需要记录部门经理的主键，也就是“部门经理工号”(或“经理工号”、或“员工工号”)。“厂商 ID”为“家电厂商”的主键。
根据实体联系图和需求描述，“客户”、“客服”和“家电”之间的多对多对多的“购买”联系。因为是多对多对多联系，所以“购买”联系需要单独作为一个关系，这个关系需要 id 录“客户”、“客服”和“家电”的主键。所以，对于“购买”关系模式，需补充属性“客户 ID” “客服工号”和“家电条码”。“订购单号”为“购买”的主键。

答案

【试题分析】
本问题考查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，需要新增“客户经理”，包含于“员工”。
根据问题描述，客户只由一名客户经理和一名基金经理负责，客户经理和基金经理均可负责多名客户，所以“客户”、“客户经理”和“基金经理”之间存在一个“理财”联系，联系的类型为多对 1 对 1(*:1:1， 或 m:1:1)。



关系模式	主键
家电	家电条码
家电厂商	厂商 ID
购买	订购单号



试题三 答案： 解析：

答案

C1： Patron

C2： Book

C3： Catalog

C4： CheckoutSessioncontroller

【试题分析】

本题属于经典的考题，主要考查面向对象分析方法以及 UML 类图和通信图的相关知识。

说明中给出了一个具体用例的详细描述，给出了其中的一个系统操作“checkoutbookID)(借书)”的通信图，需要考生利用通信图中的信息来补充类图中缺失的部分。 通信图(communicationdiagram)强调收发消息的对象的结构组织，在早期的版本中也被称作协作图。通信图强调参加交互的对象的组织。产生一张通信图，首先要将参加交互的对象作为图的顶点，然后把连接这些对象的链表示为阁的弧，最后用对象发送和接收的消息来修饰这些链。这就提供了在协作对象的结构组织的语境中观察控制流的一个清晰的可视化轨迹。

消息 checkOut(booklD)的接收者是类 CheckoutSessionContrailer 的对象，说明类 CheckoutSessionController 中应该包含这一方法，否则无法响应该条消息。由图 3-1 可知， C4 处所代表的类应该是 CheckoutSessionController 。

消息 find(booklD)的接收者是类 Book ，同理，由图 3-1 可知， C2 处对应的类应该是 Book 。

根据用例描述，图书信息是包含在图书目录中，所以 C3 处对应的类应该是 Catalog，C1 处对应的就应该是 Patron 了。

答案

M1:getforcheckout

M2:isFaculty

M3:circulates

M4:recordBookLoan

【试题分析】

图 3-1 填充完整之后，图 3-2 的空缺就比较容易填写了。在通信图中，对象之间传递的消息就对应着接收对象中的方法。 M1 对应的就是类 Catalog 中的方法，由图 3-1 可 知， M1 对应的是 getForCheckOut 。

M3 对应的应该是类 Book 中的方法。由图 3-1 可知， Book 中有 3 个方法， find 和 checkout 已经出现在通信图上了，所以 M3 应该是 circulates 。

M2 和 M4 是类 Patron 中的方法。 Patron 中有 2 个方法。通信图中的消息是有序号的，这个序号表示了消息的时间顺序，也就是说发送 M2 的时间要早于消息 M4,因此必须区分类

Patron 中两个方法使用的先后顺序。在用例描述中特别指出：图书的归还时间与读者的身份有关。计算还书及借书费用时，需先确定读者的身份，因此方法 isFaculty 应该先被调用，所以 M2 对应 isFaculty， M4 对应 recordBookLoan 。

答案

策略模式

策略模式定义了一系列算法，并将每个算法封装起来，而且使它们可以相互替换。 策略模式让算法独立于使用它们的客户而变化。适用于需要在不同情况下使用不同的策略(算法)，或者策略还可能在未来用其他方式来实现。

【试题解析】

本题在设计类时使用到了策略模式。

策略模式定义了一系列的算法，把它们一个个封装起来，并且使它们可以相互替换。此模式使得算法可以独立于使用它们的客户而变化。策略模式的结构如下图所示。

其中：

• Strategy（策略)定义所有支持的算法的公共接口。 Context 使用这个接口来调用某 ConcreteStrategy 定义的算法。

• ConcreteStrategy（具体策略)以 Strategy 接口实现某具体算法。

• Context（上下文)用一个 ConcreteStrategy 对象来配置；维护一个对 Strategy 对象的引用；可定义一个接口来让 Strategy 访问它的数据。

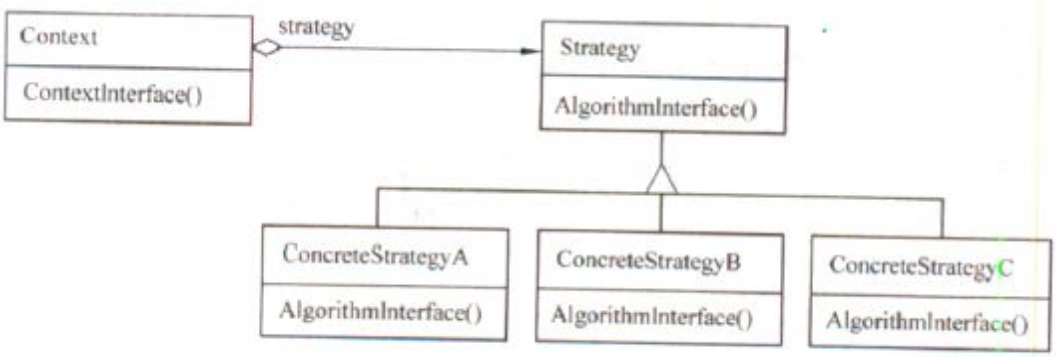
Strategy 模式适用于：

•许多相关的类仅仅是行为有异。“策略”提供了一种用多个行为中的一个行为来配置一个类的方法。

•需要使用一个算法的不同变体。例如，定义一些反应不同空间的空间/时间权衡的算法。当这些变体实现为一个算法的类层次时，可以使用策略模式。

•算法使用客户不应该知道的数据。可使用策略模式以避免暴露复杂的、与算法相关的数据结构。

• 一个类定义了多种行为，并且这些行为在这个类的操作中以多个条件语句的形式出现，将相关的条件分支移入它们各自的 Strategy 类中，以代替这些条件语句。



试题四 答案： 解析：

答案

1)k≤r 或 k

2)arr[k]=right[j]

3) begin

4)mergeSort(arr,mid+1,end)

【试题分析】

本题考查算法设计、分析和C程序实现的知识，属于传统题目，考查点也与往年类似。

归并排序是一种经典的排序算法，基本思想：把n个元素构成的数组分成两个n/2 个元素构成的子数组，再进一步划分，一直到每个子数组仅包含1个元素，此时再把两 两有序的数组合并成更大的有序数组，一直到整个数组有序为止。

本问题考查算法的实现。C程序中有两个函数， merge函数将两个有序数组合并成一个更大的有序数组。归并过程是首先将两个有序的子数组的元素分别放到left 和right 数组中，然后依次比较这两个数组中的元素，从小到大把元素放到arr 数组的特定元素中。包含空格(1) 的for 循环中，给出了将left 和right 元素放入arr 中，放入的位置是从p 到r,因此，空格(1) 填写k <= r 。在比较left[i]和right[j]元素时，若left[i]> right[j]， 则应该把right[j]的值放入arr[k]中，因此空格(2) 填写arr[k] = right[j]。mergeSort 函数进行数组的排序。若数组元素个数大于1，则继续划分，因此空格(3) 填写begin

答案

5) 分治

6) T(n)=2T(N/2)+O(n)或 T(n)=2T(n/2)+f(n) (f(n)为线性函数)

7) O(nlogn)

8) O(n)

【试题分析】

本问题考查算法的设计策略和时间复杂度，归并排序算法是一个典型的分治算法。每次将一个规模为 n 的问题变成两个规模为 n/2 的子问题，划分时直接从中间分开，因此划分采用 O(1) 的时间，然后是递归求解两个子问题，根据 merge 函数代码，合并的时间是线性时间 O(n)。因此递归式为 T(n)=2T(n/2)+f(n)，f(n)为线性函数。用主方法求解，得到时间复杂度为 O(nlgn)。由于在归并过程中，需要 left 和 right 两个辅助数组，其规模为待排序的数组长度，即 O(n)。

答案

(9) nl+n2

【试题分析】

本问题考查对算法的进一步分析。元素之间的比较次数就是 merge 函数的最后一个 for 循环体执行的次数，由于 k 从 p 到 r，故循环体执行次数为 r-p+1 次，即 n1+n2 次。

试题五 答案： 解析：

答案

1)Subject

2)(*it)->update(remperature， humidity， cleanness)

3) notify() bservesers

4) mesasurementsChanged()

5) Observer

6) envData-> regisiterObserver(this)

【试题分析】

本题考察观察者(Observer)模式的概念及应用。

观察者模式定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依

赖于它的对象都得到通知并被自动更新。

Observer 模式适用于：

①当一个抽象模型有两个方面，其中一个方面依赖于另一个方面。将这两者封装在独立地对象中以使它们可以各自独立地改变和复用。

②当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象有待改变时。

③当一个对象必须通知其他对象，而它又不能假定其他对象是谁，即：不希望这些对象是紧耦合的。

观察者模式的结构如下图所示，其中：

Subject（主题）知道它的观察者，可以有任意多个观察者观察同一个目标；提供注册和删除观察者对象的接口。

Observer（观察者）为那些在目标发生改变时需获得通知的对象定义一个更新接口。

ConcreteSubject（具体主题）将有关状态存入一个 ConcreteObserver 对象；当它的状态发生改变时，向它的各个观察者发出通知。

ConcreteObserver（具体观察者）维护一个指向 ConcreteSubject 对象的引用；存储有关状态，这些状态应与主题的状态保持一致；实现 Observer 的更新接口以使自身状态与主题的状态保持一致。

在本题的说明中，给出了观察者模式的结构图，答题时需要首先确定程序中的类与观察者模式结构的对应关系，也就是要找到哪些是 ConcreteSubject，哪些是 ConcreteObserver。由程序上下文可以判断出，类 EnvironmentData 对应的是 ConcreteSubject，类 CurrentConditionsDisplay 对应的是 ConcreteObserver。根据类图，它们分别为 Subject 和 Observer 的子类。由此可以，空(1) 和空(5) 应分别填写 Subject 和 Observer。

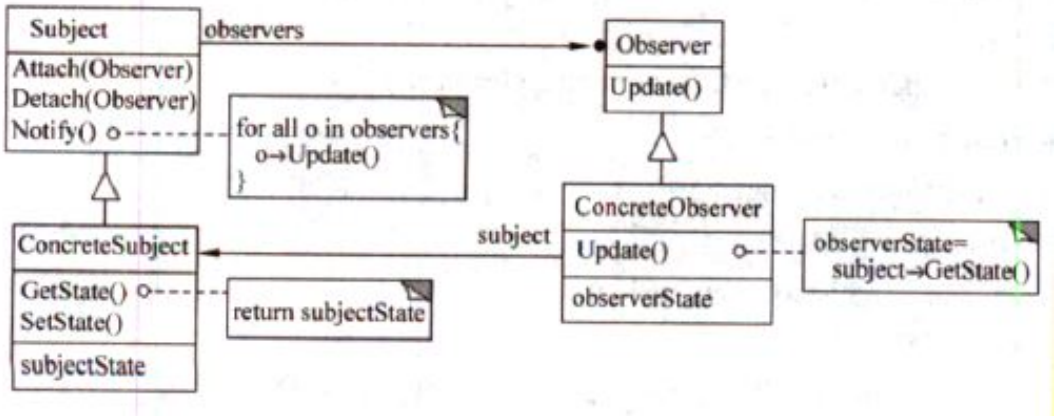
空(2) 要求给出方法 notifyObservers 的实现，其功能是在主题发生变化时通知观察者。通知的实现是通过向对该主题感兴趣的所有观察者发送 update 消息实现的。对主题 EnvironmentData 感兴趣的观察者由向量 observers 表示，所以在 notifyObservers 方法中，就是对向量 observers 中的每个成员发送 update 消息，因此空(2) 应填写 (*it)->update(temperature, humidity, cleanness)。

方法 measurementsChanged 表示主题发生了变化，这时应该通知对应的观察者，所以空(3) 处应填写 notifyObservers()。

方法 setMeasurements 用于设置发生变化后的主题内容，所以空(4) 处应填写 measurementsChanged()。

CurrentConditionsDisplay 是对主题 EnvironmentData 感兴趣的一个观察者，要能够获得

主题的变化，需要首先将自己注册为该主题的观察者，这个注册行为在其构造函数中完成。因此空(6) 处应填写 envData->registerObserver(this)。



试题六 答案： 解析：

答案

(1) Subject

(2) observer.update(temperature, humidity, cleanness)

(3) notifyObservers()

(4) measurementsChanged()

(5) Observer

(6) envData.registerObserver(this)

【试题分析】

本题考查观察者(Observer)模式的概念及应用。

观察者模式定义对象间的一种一对多的依赖关系，当一个对象的状态发生改变时，所有依赖于它的对象都得到通知并被自动更新。

Observer 模式适用于：

①当一个抽象模型有两个方面，其中一个方面依赖于另一个方面。将这两者封装在独立的对象中以使它们可以各自独立地改变和复用。

②当对一个对象的改变需要同时改变其他对象，而不知道具体有多少对象有待改变时。

③当一个对象必须通知其他对象，而它又不能假定其他对象是谁，即：不希望这些对象是紧耦合的。

观察者模式的结构如下图所示，其中：

Subject（主题)知道它的观察者，可以有任意多个观察者观察同一个目标；提供注册和删除观察者对象的接口。

Observer（观察者)为那些在目标发生改变时需获得通知的对象定义一个更新接口。

ConcreteSubject（具体主题)将有关状态存入一个 ConcreteObserver 对象；.当它的状态发生改变时，向它的各个观察者发出通知。

ConcreteObserver（具体观察者)维护一个指向 ConcreteSubject 对象的引用；存储有关状态，这些状态应与主题的状态保持一致；实现 Observer 的更新接口以使自身状态与主题的状态保持一致。

在本题的说明中，给出了观察者模式的结构图，答题时需要首先确定程序中的类与观察者模式结构的对应关系，也就是要找到哪些是 ConcreteSubject,哪些是 ConcreteObserver。由程序上下文可以判断出，类 EnvironmentData 对应的是 ConcreteSubject,类 CurrentConditionsDisplay 对应的是 ConcreteObserver。根据类图，它们分别为 Subject 和 Observer 的子类。由此可以，空(1）和空(5）应分别填写 Subject 和 Observer。

空(2）要求给出方法 notifyObservers 的实现，其功能是在主题发生变化时通知观察者。通知的实现是通过向对该主题感兴趣的所有观察者发送 update 消息实现的。对主题 EnvironmentData 感兴趣的观察者由向量 observers 表示，所以在 notifyObservers 方法中，就是对向量 observers 中的每个成员发送 update 消息，因此空(2）应填写 observer.update(temperature， humidity, cleanness)。

方法 measurementsChanged 表示主题发生了变化，这时应该通知对应的观察者，所以空(3)处应填写 notifyObservers()。

方法 setMeasurements 用于设置发生变化后的主题内容，所以空(4）处应填写 measurementsChanged()。

CurrentConditionsDisplay 是对主题 EnvironmentData 感兴趣的一个观察者，要能够获得主题的变化，需要首先将自己注册为该主题的观察者，这个注册行为在其构造函数中完成。因此空(6）处应填写 envData.registerObserver(this)。