2014 年上半年软件设计师考试下午真题（参考答案）

- 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。

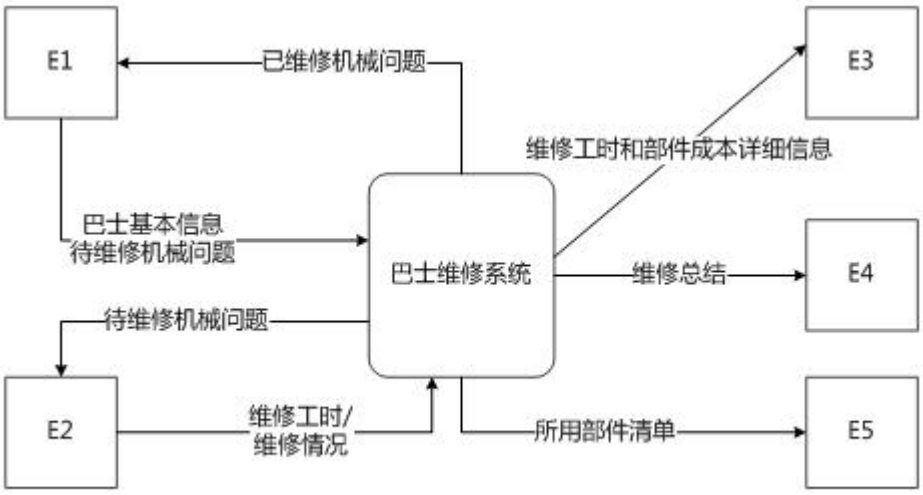

图 1-1 上下文数据流图

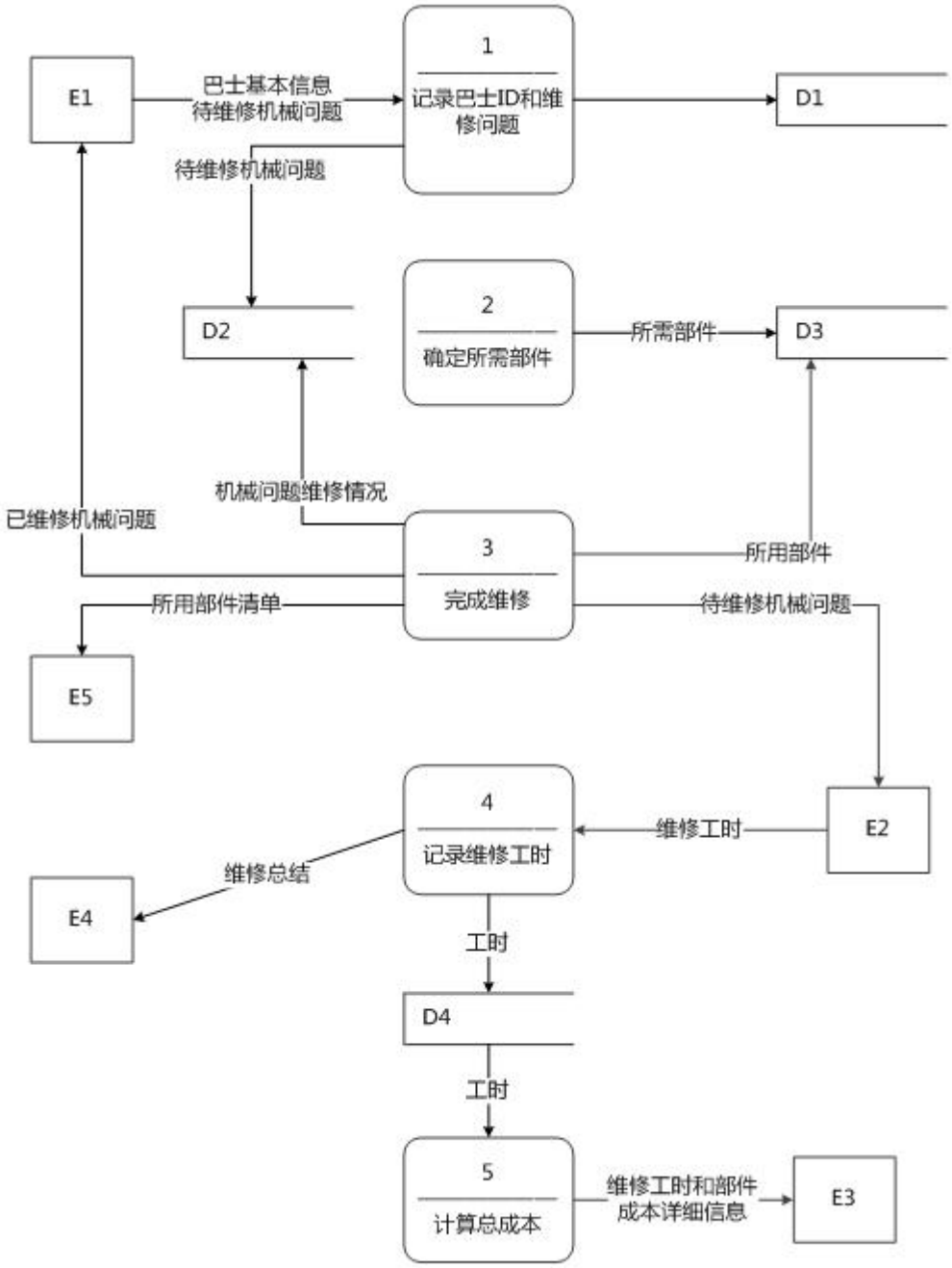

图 1-2 0 层数据流图

【问题 1】（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 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 3】（3 分）

说明图 1-2 中所存在的问题。

【问题 4】（3 分）

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。

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 1-1 所示。

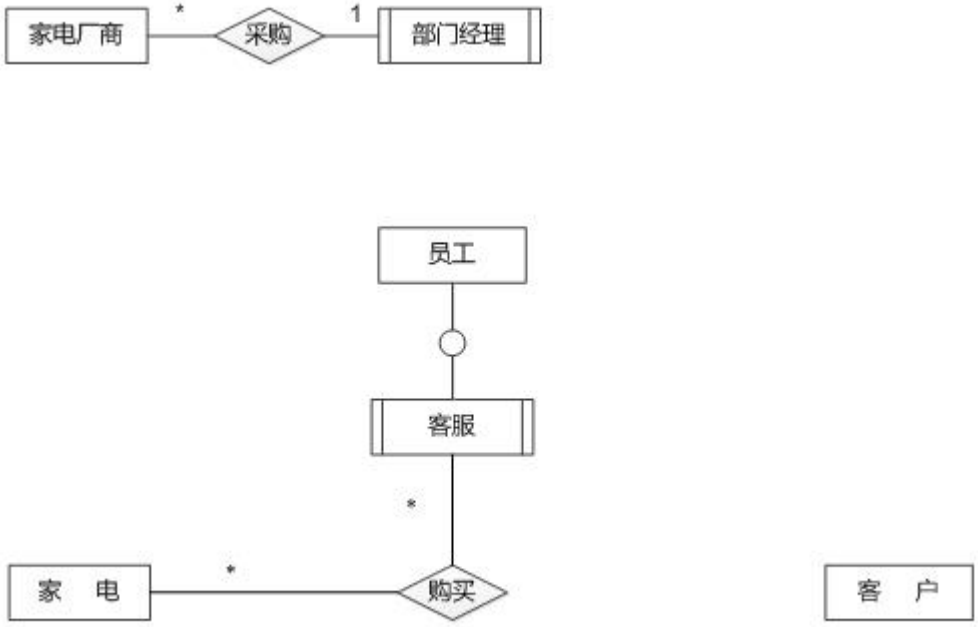

图 1-1 实体联系图

【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〕：

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）

【问题 1】（6 分）

补充图 1-1 中的联系和联系的类型。

【问题 2】（6 分）

根据图 1-1，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 3】（3 分）

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，还可以联合第二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 1-1 进行修改，画出修改后的实体间联系和联系的类型。

●

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（Check Out Books）”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“checkOut(bookID)（借书）”通信图（或协作图）。

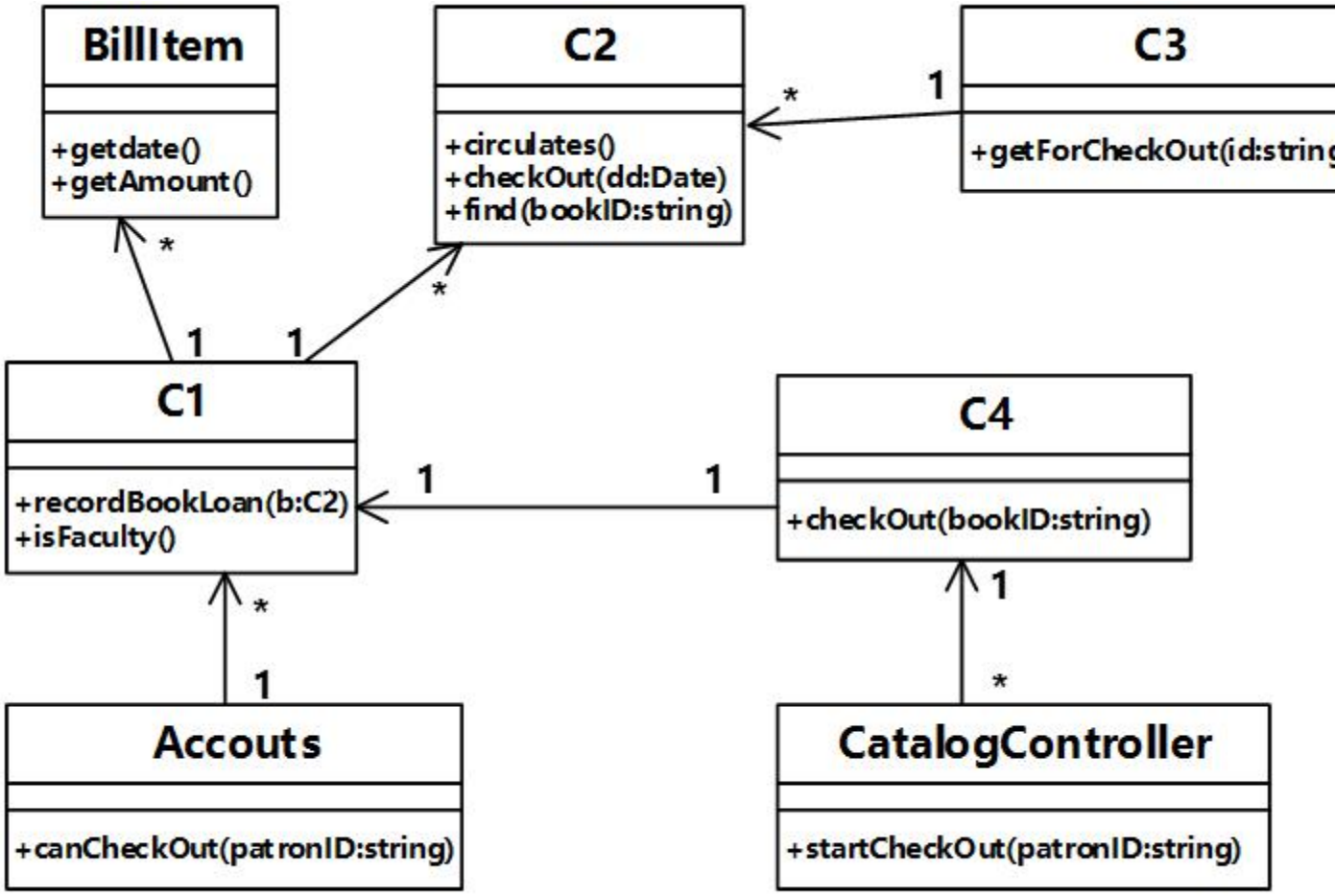
BillItem
+getdate()
+getAmount()
C2
+circulates()
+checkOut(dd:Date)
+find(bookID:string)
C3
+getForCheckOut(id:string
C1
+recordBookLoan(b:C2)
+isFaculty()
C4
+checkOut(bookID:string)
Accouts
+canCheckOut(patronID:string)
CatalogController
+startCheckOut(patronID:string)
*
1
1
*
1
1
*
1
1
*
1
*

图3-1 系统类模型

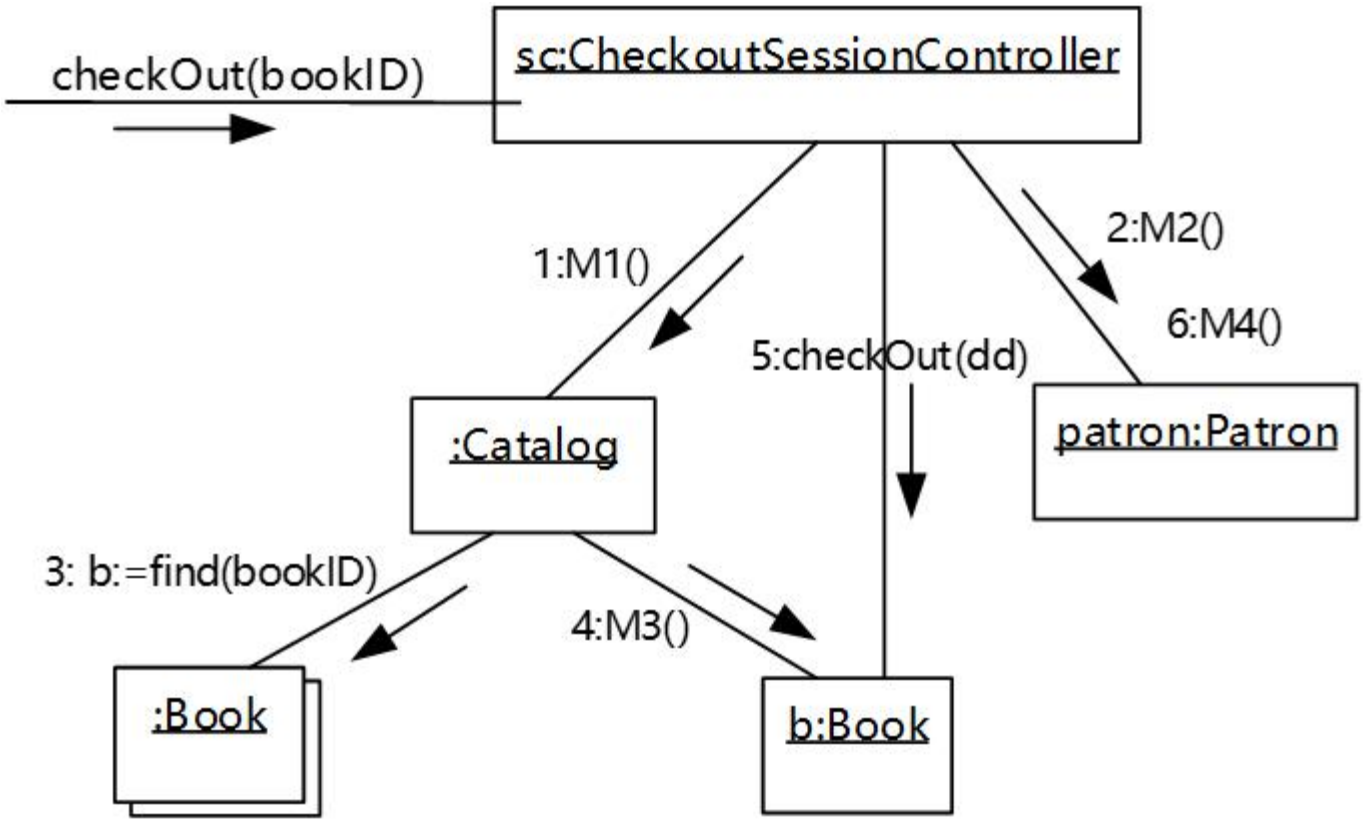

图3-2 系统操作 checkOut的通信图

【问题 1】（8 分）

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）。

【问题 2】（4 分）

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 分）

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。

- 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：一个子数组的位置从 p 到 q，另一个子数组的位置从 q+1 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】

```
#inciude<stdio.h>
#inciude<stdlib.h>
#define MAX 65536
void merge(int arr[],int p,int q,int r) {
   int *left, *right;
   int n1,n2,i,j,k;
   n1=q-p+1;
   n2=r-q;
   if((left=(int*)malloc((n1+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 1】

根据以上说明和 C 代码，填充 1-4。

【问题 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，列出其递归式位（6），解出渐进时间复杂度为（7）（用 O 符号表示）。空间复杂度为（8）（用 O 符号表示）。

【问题 3】

两个长度分别为 n1 和 n2 的已经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）。

- 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。

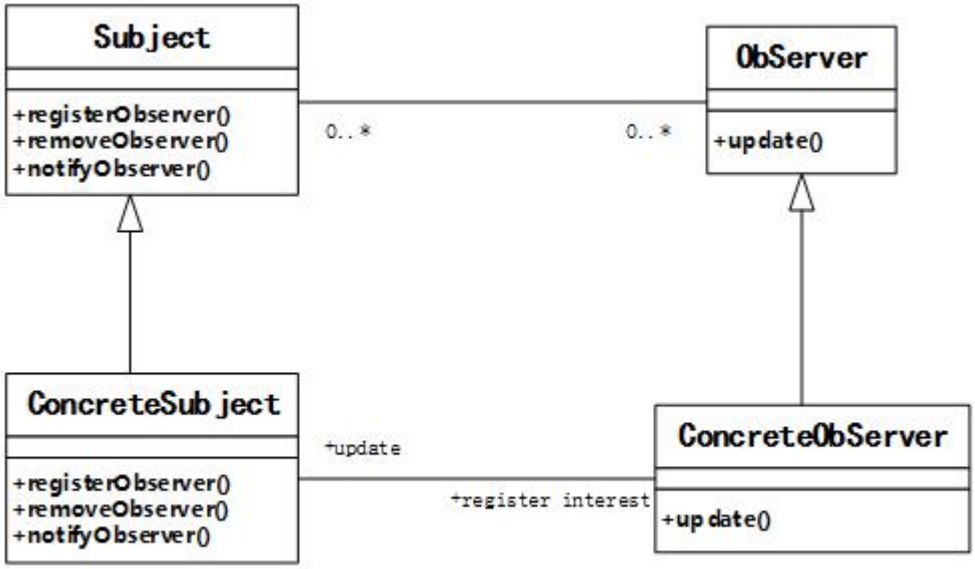

图 5-1 观察者模式类图

【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.end(); it++)
{  （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;
}
```

- 阅读下列说明和 Java 代码，将应填入（n）处的字句卸载答题纸的对应栏内.

【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 6-1 所示。

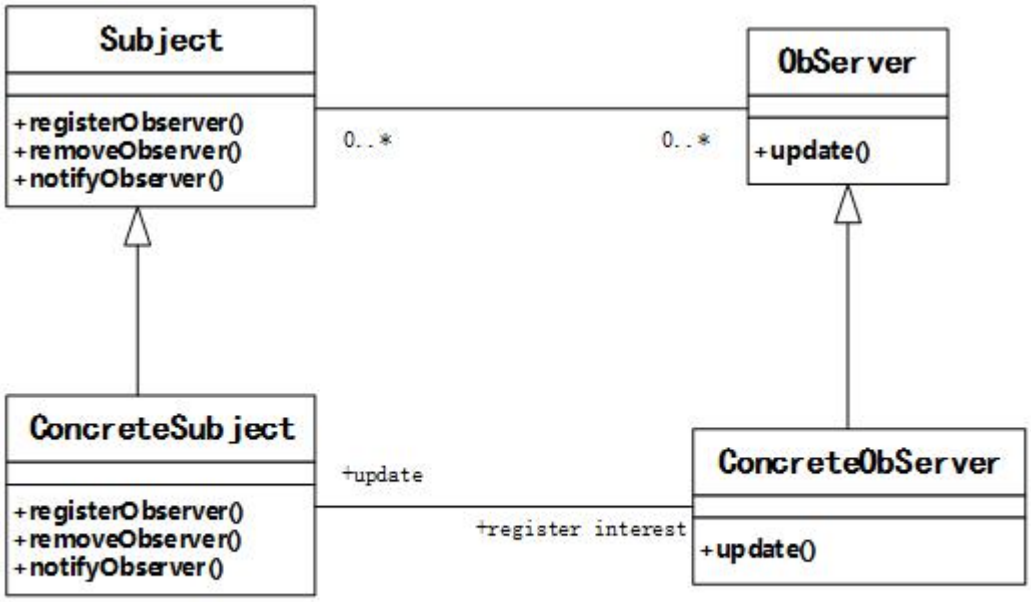

图 6-1 观察者模式类图

【Java 代码】

```
    import jav
（6）A．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 CnrrentConditionsDisplay(envData);
      envDat
（7）A．setMeasurements(80, 65, 30.4f);
   }
}
```