试题一(共 15 分)

阅读下列说明，回答问题 1 至问题 4，将解答填入答题纸的对应栏内。

【说明】

某医疗护理机构为老年人或有护理需求者提供专业护理，现欲开发一基于 Web 的医疗管理系统，以改善医疗护理效率。该系统的主要功能如下:

(1)通用信息查询。客户提交通用信息查询请求，查询通用信息表，返回查询结果。

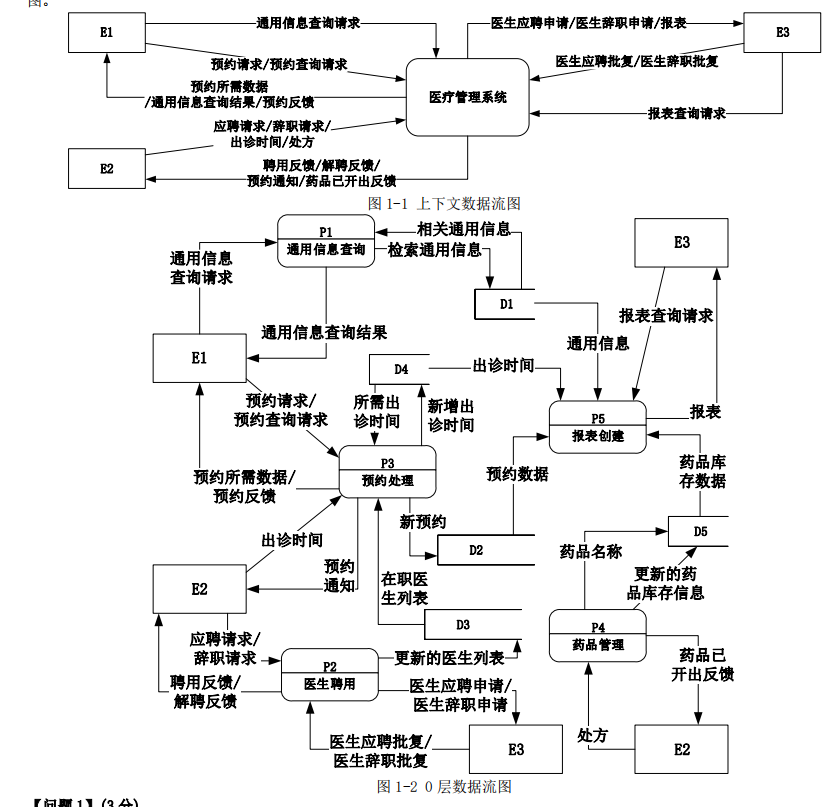
(2)医生聘用。医生提出应聘/辞职申请，交由**主管**进行聘用/解聘审批，更新医生表，并给医生反馈聘用/解聘结果**;删除解聘医生的出诊安排**。

(3)预约处理。医生安排出诊时间，存入医生出诊时间表;根据客户提交的预约查询请求，查询在职医生及其出诊时间等预约所需数据并返回;创建预约，提交预约请求，在预约表中新增预约记录，更新所约医生出诊时间并给医生发送预约通知;给客户反馈预约结果。

(4)药品管理。医生提交处方，根据药品名称从**药品数据**中查询相关药品**库存信息**，开出药品，更新对应药品的库存以及预约表中的治疗信息;给医生发送“药品已开出”反馈

(5)报表创建。根据主管提交的报表查询请求(报表类型和时间段)，从预约数据、通用信息、药品库存数据、医生以及医生出诊时间中进行查询，生成报表返回给主管。

现采用结构化方法对医疗管理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



【问题 1】(3 分)

使用说明中的词语，给出图 1-1 中的实体 El〜E3 的名称。

【问题 2】(5 分)

使用说明中的词语，给出图 1-2 中的数据存储 D1〜D5 的名称。

【问题 3】(4 分)

使用说明和图中术语，补充图 1-2 中缺失的数据流及其起点和终点。

【问题 4】(3 分)

使用说明中的词语，说明“预约处理”可以分解为哪些子加工，并说明建模图 1-1 和图 1-2 时如何保持数据流图平衡。

试题二(共 15 分)

阅读下列说明，回答问题 1 至问题 3,将解答填入答题纸的对应栏内。

【说明】

某海外代购公司为扩展公司业务，需要开发一个信息化管理系统。请根据公司现有业务及需求完成该系统的数据库设计。

【需求描述】

(1)记录公司员工信息。员工信息包括工号、身份证号、姓名、性别和一个手机号，工号唯一标识每位员工，员工分为代购员和配送员。

(2)记录采购的商品信息。商品信息包括商品名称、所在超市名称、采购价格、销售价格和商品介绍，系统内部用商品条码唯一标识每种商品。一种商品只在一家超市代购。

(3)记录顾客信息。顾客信息包括顾客真实姓名、身份证号(清关缴税用)、一个手机号和一个收货地址，系统自动生成唯一的顾客编号。

(4)记录托运公司信息。托运公司信息包括托运公司名称、电话和地址，系统自动生成唯一的托运公司编号。

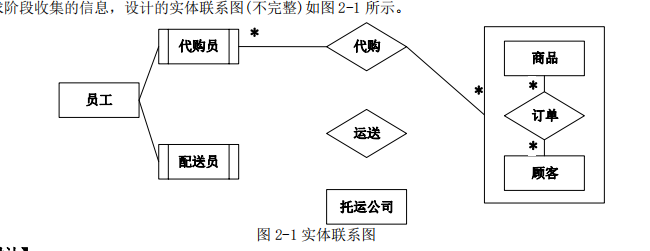
(5)**顾客**登录系统之后，可以下订单购买**商品**。订单支付成功后，系统记录唯一的支付凭证编号，顾客需要在订单里指定运送方式：空运或海运。

(6)代购员根据顾客的**订单**在超市采购对应**商品**，一份订单所含的多个商品可能由多名代购员从不同超市采购。

(7)采购完的商品交由配送员根据顾客订单组合装箱，然后交给托运公司运送。托运公司按顾客订单*核对商品名称和数量*，然后按顾客的*地址*进行运送。

【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)：

员工(工号，身份证号，姓名，性别，手机号)

商品(条码，商品名称，所在超市名称，采购价格，销售价格，商品介绍)

顾客(编号，姓名，身份证号，手机号，收货地址)

托运公司(托运公司编号，托运公司名称，电话，地址)

订单(订单 ID, (a) ，商品数量，运送方式，支付凭证编号)

代购(代购 ID,代购员工号， (b) )

运送(运送 ID,配送员工号，托运公司编号，订单 ID，发运时间)

【问题 1】(3 分)

根据问题描述，补充图 2-1 的实体联系图。

【问题 2】(6 分)

补充逻辑结构设计结果中的(a)、(b)两处空缺。

【问题 3】(6 分)

为方便顾客，允许顾客在系统中**保存**多组收货地址。请根据此需求，增加“顾客地址”弱实体，对图 2-1 进行补充，并修改“运送”关系模式。

试题三(共 15 分)

阅读下列说明，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 ETC(Electronic Toll Collection,不停车收费)系统在高速公路沿线的特定位置上设置一个横跨道路上空的龙门架 (Toll gantry),龙门架下包括 6 条车道(Traffic lanes),每条车道上安装有雷达传感器(Radar sensor)、无线传输器(Radio transceiver)和数码相机(Digital Camera)等用于不停车收费的设备，以完成正常行驶速度下的收费工作。该系统的基本工作过程如下：

(1)每辆汽车上安装有车载器，***驾驶员(Driver)***将一张具有唯一识别码的磁卡插入车载器中。磁卡中还包含有驾驶员账户的当前信用记录。

(2)当汽车通过某条车道时，不停车收费设备识别车载器内的特有编码，判断车型，将收集到的相关信息发送到该路段所属的***区域系统***(Regional center)中，计算通行费用，创建收费交易(Transaction)，从驾驶员的专用账户中扣除通行费用。如果驾驶员账户透支,则记录透支账户交易信息。区域系统再将交易后的账户信息发送到维护驾驶员账户信息的***中心系统***(Central system)。

(3)车载器中的**磁卡**可以使用邮局的付款机进行充值。充值信息会传送至中心系统,以更新驾驶员账户的余额。

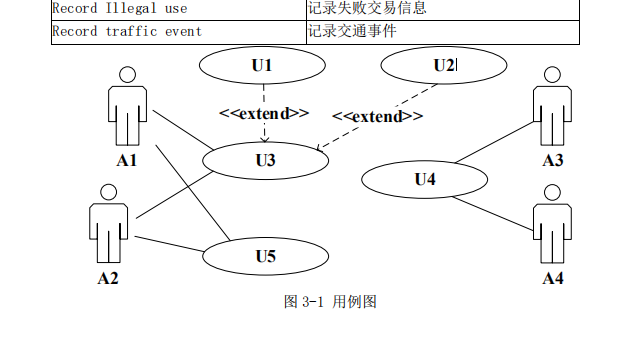
(4)当没有安装**车载器**或者车载器发生故障的车辆通过车道时，车道上的**数码相机**将对车辆进行拍照，并将车辆照片及拍摄时间**发送到区域系统**，记录失败的交易信息；并将该交易信息发送到中心系统。

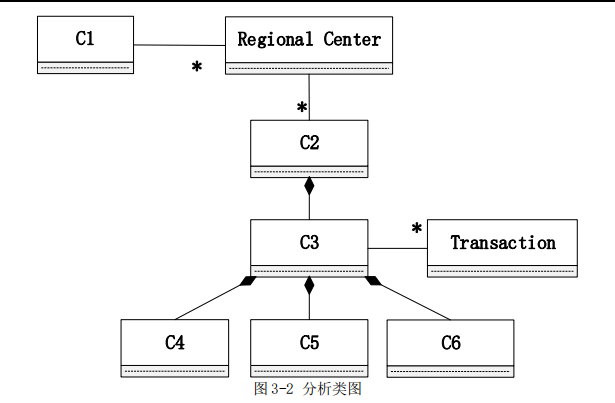
(5)区域系统会获取不停车收费**设备所记录的**交通事件(Traffic events);***交通广播电台(Traffic advice center)***根据这些交通事件进行路况分析并播报路况。

现采用面向对象方法对上述系统进行分析与设计，得到如表 3-1 所示的用例列表以及如图 3-1 所示的用例图和图 3-2所示的分析类图。

表 3-1 用例列表

|  |  |
| --- | --- |
| 用例名称 | 说明 |
| Create transaction | 记录收费交易 |
| Charge card | 磁卡充值 |
| Underpaid transaction | 记录透支账户交易信息 |
| Record Illegal use | 记录失败交易信息 |
| Record traffic event | 记录交通事件 |





【问题 1】(4 分)

根据说明中的描述，给出图 3-1 中 A1〜A4 所对应的参与者名称。

【问题 2】(5 分)

根据说明中的描述及表 3-1，给出图 3-1 中 U1〜U5 所对应的用例名称。

【问题 3】(6 分)

根据说明中的描述，给出图 3-2 中 C1〜C6 所对应的类名。

试题四(共 15 分)

阅读下列说明和 C 代码，回答问题 1 和问题 2,将解答填入答题纸的对应栏内。

【说明】

某公司购买长钢条，将其切割后进行出售。切割钢条的成本可以忽略不计，钢条的长度为整英寸。已知价格表 p，其中pi (i=l,2,...,m)表示长度为 i 英寸的钢条的价格。现要求解使销售收益最大的切割方案。

求解此切割方案的算法基本思想如下：

假设长钢条的长度为 n 英寸，最佳切割方案的最左边切割段长度为 i 英寸，则继续求解剩余长度为 n-i 英寸钢条的最佳切割方案。考虑所有可能的 i,得到的最大收益rn对应的切割方案即为最佳切割方案。rn的递归定义如下：

rn=max1≤i≤𝑛(pi + rn−i)

对此递归式，给出自顶向下和自底向上两种实现方式。

【C 代码】

/\* 常量和变量说明

n:长钢条的长度

p[]:价格数组

\*/

#define LEN 100

int Top\_Down\_Cut\_Rod(int p[], int n) {/\*自顶向下\*/

int r = 0;

int i;

if(n == 0){ return 0; }

for(i = 1; (1) ; i++) {

int tmp = p[i] + Top\_Down\_Cut\_Rod(p,n-i);

r = (r >= tmp) ? r : tmp;

}

return r;

}

int Bottom\_Up\_Cut\_Rod(int p[], int n) {/\*自底向上\*/

int r[LEN] = { 0 };

int temp = 0;

int i, j;

for (j = 1; j <= n; j++) {

temp = 0;

for (i = 1; (2) ; i++) {

temp = (3) ;

}

(4) ;

}

return r[n];

}

【问题 1】(8 分)

根据说明，填充 C 代码中的空(1)~(4)。

【问题 2】(7 分)

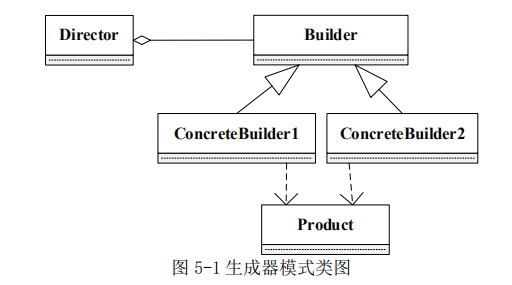
根据说明和 C 代码，算法采用的设计策略为 (5) 。求解rn时，自顶向下方法的时间复杂度为 (6) ；自底向上方法的时间复杂度为 (7) (用 O 表示)。

试题五(共 15 分)

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】

生成器(Builder)模式的意图是将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示。图5-1 所示为其类图。



#include<iostream>

#include<string>

using namespace std;

class Product {

private:

string partA,partB;

public:

Product() {}

void setPartA(const string &s) { partA = s; }

void setPartB(const string &s) { partB = s; }

//其余代码省略

};

class Builder {

public:

(1) ;

virtual void buildPartB() = 0;

(2) ;

};

class ConcreteBuilder1 : public Builder {

private:

Product \*product;

public:

ConcreteBuilder1() { product = new Product(); }

void buildPartA() { (3) ("Component A"); }

void buildPartB() { (4) ("Component B"); }

Product\* getResult() { return product; }

//其余代码省略

};

class ConcreteBuilder2 : public Builder {

/\*代码省略.\*/

};

class Director {

private:

Builder \* builder;

public:

Director(Builder\* pBuilder) { builder = pBuilder; }

void construct() {

(5) ;

//其余代码省略

}

//其余代码省略

};

int main() {

Director\* director1 = new Director(new ConcreteBuilder1());

director1->construct();

delete director1;

return 0;

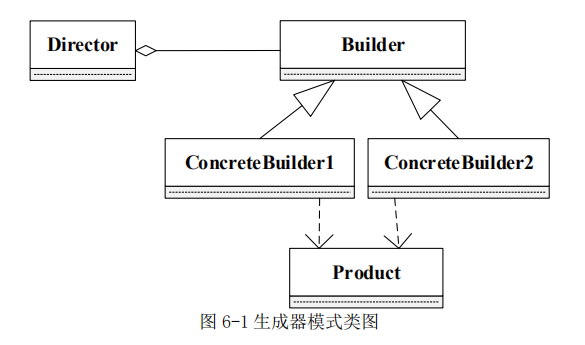
}

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。

【说明】

生成器(Builder)模式的意图是***将一个复杂对象的构建与它的表示分离，使得同样的构建过程可以创建不同的表示***。图6-1 所示为其类图。



【Java 代码】

import java.util.\*;

class Product {

private String partA;

private String partB;

public Product() {};

public void setPartA(String s) {partA = s;}

public void setPartB(String s) {partB = s;}

}

interface Builder {

public (1) ;

public void buildPartB();

public (2) ;

}

class ConcreteBuilder1 implements Builder {

private Product product;

public ConcreteBuilder1() {

product = new Product();

}

public void buildPartA() { (3) ("Component A");}

public void buildPartB() { (4) ("Component B");}

public Product getResult() {

return product;

}

}

class ConcreteBuilder2 implements Builder {

// 代码省略

}

class Director {

private Builder builder;

public Director(Builder builder) {

this.builder = builder;

}

void construct() { (5) ;}

// 代码省略

}

class Test {

public static void main(String[] args) {

Director director1 = new Director(new ConcreteBuilder1());

director1 .construct();

}

}

试题一 参考答案及解析

【参考答案】

【问题1】E1：客户； E2：医生； E3：主管

【问题2】D1：通用信息表；D2：预约表；D3：医生列表；D4：出诊时间表；D5：药品库

【问题 3】

起点 终点 名称

P2 D4 删除解聘医生的出诊时间

P2 D2 删除解聘医生的预约数据

D5 P4 查询相关药品库存信息

P4 D2 更新预约表

D3 P5 医生 或 医生信息

【问题 4】

“预约处理”可以分解为子加工：安排出诊时间、就诊预约。

在建模数据流图时应保持父图(图 1-1)与子图(图 1-2)的平衡，即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，但如果父图的一个输入(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个数据流，那么它们仍然算是平衡的。

**【试题解析】**

本题考察面向结构化软件开发方法学中需求分析阶段使用的数据流图(DFD 图)。作答时，一定要现把题目中“功能部分”大致阅读一遍，画出关键功能性动词，以及以表或库结尾的名词，然后从图中查找题目中的对应句，做题。

【问题 1】需要填写外部实体，外部实体为不属于软件本身但是又与当前软件有交互关系的外部的人、软件、硬件、组织结构、数据库系统等，在作答时需要细心地对每一个阅读到的外部实体(一般为名词)高度重视；

【问题 2】考察数据存储文件，这需要对阅读到的“...文件”或“...表”等能够存储数据的媒介词汇高度重视。

【问题 3】不仅仅通过阅读文字描述来作答，同时也要使用父图与子图的数据守恒原则进行作答。本题的文字描述中，共阐述了(1)～(5)共 5个系统必须要实现的功能模块，这些功能(动词描述)对应数据流图中的加工 P1～P5。由于需要找出图 1-2 中缺少的数据流，故建议边阅读文字描述，边对照图 1-2 以便找出其中缺少的数据流。

从描述"(1)通用信息查询(对应加工 P1)。客户提交通用信息查询请求(可知实体‘客户’有数据流‘通用信息查询请求’流向加工‘通用信息查询’)，查询通用信息表(可知存储文件‘通用信息表’有数据流‘通用信息’流向加工‘通用信息查询’)，返回查询结果(可知加工‘通用信息查询’有数据流‘查询结果’流向实体‘客户’)。"，并对照图 1-2中 P1 加工和 E1 实体处可知 E1 为实体"客户"，D1 为“通用信息表”。此处通过对比，未发现缺少的数据流！

从描述“(2)医生聘用(对应加工 P2)。医生提出应聘/辞职申请(可知实体‘医生’有数据流‘应聘/辞职请求’流向加工 P2)，交由主管进行聘用/解聘审批(可知加工 P2 有数据流“聘用申请”和“解聘申请”流向实体‘主管’，以便主管审批)，更新医生表(可知加工 P2 有数据流流向存储文件‘医生表’，即 D3)，并给医生反馈聘用/解聘结果(可知加工 P2有对应的数据流‘聘用反馈’和‘解聘反馈’流向实体‘医生’);删除解聘医生的出诊安排(可知应该有一条数据流由 P2流向类似‘出诊安排表’的存储文件，此处缺少此数据流，打个标记，待确定‘出诊安排表’类似的文件后再补上)”。阅读到此，可确定 E2 为“医生”，E3 为“主管”，D3 为“医生列表”，且找到一条缺少的数据流。

阅读描述“(3)预约处理(对应加工 P3)。医生安排出诊时间(对应图中 E2 流向 P3 的数据流‘出诊时间’)，存入医生出诊时间表(对应图中 P3 流向 D4 的数据流‘新增出诊时间’，即 D4 为‘出诊时间表’);根据客户提交的预约查询请求 (对应 E1 至 P3 的数据流‘预约查询请求’)，查询在职医生(对应 D3 至 P3 的数据流‘在职医生列表’)及其出诊时间(对应 D4 至 P3 的数据流‘所需出诊时间’)等预约所需数据并返回(对应 P3 至 E1 的数据流‘预约所需数据’);创建预约，提交预约请求(对应 E1 至 P3 的数据流‘预约请求’)，在预约表中新增预约记录(对应 P3 至 D2 的数据流‘新预约’，即 D2为‘预约表’)，更新所约医生出诊时间(对应 P3 至 D4 的数据流‘新增出诊时间’)并给医生发送预约通知(对应 P3 至 E2的数据流‘预约通知’);给客户反馈预约结果(对应 P3 至 E1 的数据流‘预约反馈’)。”。到此得出 D2 为预约表，D3 为 “医生表”，D4 为“出诊时间表”。根据本段的分析加上之前标记缺少的数据流，可分析出缺少的数据流有两条，一条由P2 流向 D4，名称为“删除解聘医生的出诊时间”，另外一条由 P2 流向 D2，名称为“删除解聘医生的预约数据”。

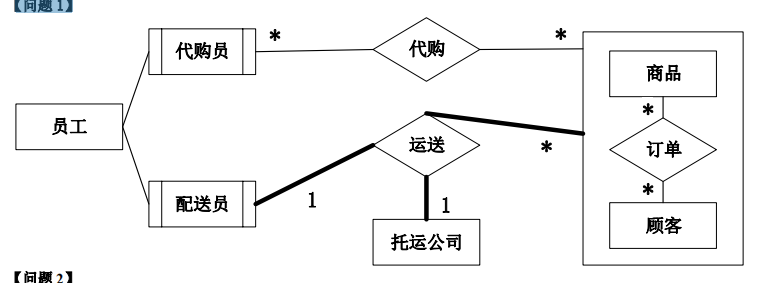
阅读描述“(4)药品管理(对应加工 P4)。医生提交处方(对应 E2 至 P4 的数据流‘处方’)，根据药品名称从药品数据中查询相关药品库存信息(可知存储文件‘药品库’应有数量流‘查询相关药品库存信息’流向 P4，图中未发现)，开出药品，更新对应药品的库存(对应 P4 至 D5 的数据流‘更新的药品库存信息’，知 D5 为‘药品库’)以及预约表中的治疗信息(可分析出 P4 应有数据流‘更新预约表’流向 D2，图中未发现);给医生发送“药品已开出”反馈(对应 P4 至 E2 的数据流‘药品已开出反馈’数据流)。”。到此，可分析出 D5 为“药品库”，且缺少 P4 至 D2 的数据流“更新预约表”和 D5 流向 P4 的数据流“查询相关药品库存信息”。

阅读描述“(5)报表创建(对应加工 P5)。根据主管提交的报表查询请求(报表类型和时间段)(对应 E3 至 P4 的数据流 ‘报表查询请求’)，从预约数据(对应 D2 至 P5 的数据流‘预约数据’)、通用信息(对应 D1 至 P5 的数据流‘通用’)、药品库存数据(对应 D5 至 P5 的数据流‘药品库存数据’)、医生(对应 D3 至 P5 的数据流‘医生信息’，缺少)以及医生出诊时间(对应 D4 至 P5 的数据流‘出诊时间’)中进行查询，生成报表返回给主管(对应 P5 至 E3 的数据流‘报表’)。”。通过本段分析，发现缺少 D3 至 P5 的一条数据流，名称为“医生信息”。通过仔细阅读“(3)预约处理……给客户反馈预约结果”段落，其中“医生安排出诊时间，存入医生出诊时间表;”完全属于医生使用的功能“安排出诊时间”，而“根据客户提交的预约查询请求，查询在职医生及其出诊时间等预约所需数据并返回;创建预约，提交预约请求，在预约表中新增预约记录，更新所约医生出诊时间并给医生发送预约通知;给客户反馈预约结果。”属于客户的“就诊预约”功能，故“预约处理”可以分解为“安排出诊时间”和“就诊预约”两个子加工。在本题的分层数据流图中，图 1-1 是父图，图 1-2 是子图，在建模数据流图时应保持父图与子图的平衡，即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，但如果父图的一个输入（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个数据流，那么它们仍然算是平衡的。

试题二 参考答案及解析

【参考答案】

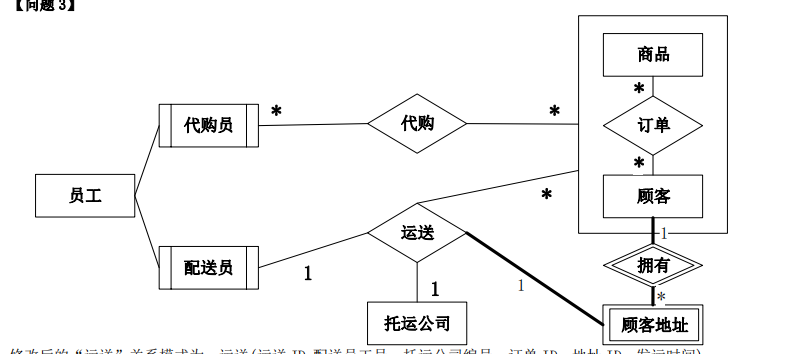
【问题 1】



【问题 2】

(a)运送方式、顾客编号、商品条码 (b)订单 ID

【问题 3】



修改“运送”模式:运送(运送 ID,配送员工号，托运公司编号，订单ID，地址ID，发运时间)。【试题解析】

下午试题二的解题方法，建议为先阅读问题，勾画出关键字，再去边仔细阅读文字描述，边看图、边看关系模式边作答。

根据文字描述“(5)顾客登录系统之后…顾客需要在订单里指定运送方式：空运或海运…”知关系模式“订单”缺少一属性“运送方式”，又由于将 E-R 图转换为关系模式时，E-R 图中多对多的联系“订单”转换的关系二维表中需要将两端的实体“商品”和“顾客”的主键字段加入对应的二维表，故关系模式“订单”中还缺少“顾客编号”和“商品条码”两个属性，以实现关系模式的参照完整性,故空(b)为“运送方式、顾客编号、商品条码”。

根据文字描述“(6)代购员根据顾客的订单在超市采购对应商品，一份订单所含的多个商品可能由多名代购员从不同超市采购。”知 E-R 图中“订单”与“代购员”之间是多对多的联系，此联系即为“代购”，将联系“代购”转换为关系模式时，为了实现参照完整性，需要将两端的主键字段加入对应的关系模式，故关系模式“代购”中缺少订单的主键“订单ID”，空(a)为“订单 ID”。

阅读文字描述“(7)采购完的商品交由配送员根据顾客订单组合装箱(说明一个订单的所有商品整体打包运送，一个订单一个包一个配送员足够)，然后交给托运公司运送。托运公司按顾客订单核对商品名称和数量(说明一个订单一个包对应一个托运公司)，然后按顾客的地址进行运送。”可知，运送与实体“托运公司”、子实体“配送员”、“订单”之间有关联关系，配送员、托运公司、订单三者之间的关系为“1:1：\*”，补充后的图 2-1 请参加【问题 1】的参考答案。

为方便顾客，允许顾客在系统中保存多组收货地址，即属性“收货地址”变成了实体“顾客”的多值属性，为了避免数据冗余而带来的数据表的存储异常问题，可新增一弱实体“顾客地址”，其对强实体“顾客”有存在依赖关系，弱实体用双边框矩形表示(在 E-R 中的属性为“地址编号，收货地址”)，其与强实体之间的联系用双线菱形表示，同时为了保证每个商品订单运送地址的唯一性，需要在 E-R 图中补充联系“运送”与“收货地址”的连接关系。补充后的图 2-1 参见

【问题 3】的参考答案。通过分析，不难看出，关系模式“运送”应加入一字段“地址编号”，修改后的关系模式为“运送(运送 ID,配送员工号，托运公司编号，订单 ID，地址 ID，发运时间)”。

**试题三 参考答案及解析**

【参考答案】

【问题 1】A1：Central system；A2：Driver；A3：Traffic advice center：A4：Traffic advice center。 (其中 A1 与 A2 可交换，A3 与 A4 可交换)

【问题 2】U1：Underpaid transaction；U2：Record Illegal use；U3：Create transaction；

U4：Record traffic event；U5：Charge card； (其中 U1 与 U2 可交换)

【问题 3】C1：Central system；C2：Toll gantry；C3：Traffic lanes；C4：Radar sensor；

C5：Radio transceiver；C6：Digital Camera （注意：C4、C5、C6 可交换）

**【试题解析】**

本题考查面向对象分析与设计方法，UML 是面向对象分析与设计过程中最常用的建模语言，关于面向对象分析与设计、及 UML 相关知识请参见比特培训相关课程。本题的作答方式与试题一、试题二类似，比特建议先阅读问题，带着问题去阅读文字描述，然后边阅读、边对照 UML 图、边思考、边作答的方式能精准地定位答案。

文字描述“…在高速公路沿线的特定位置上设置一个横跨道路上空的龙门架(Toll gantry),龙门架下包括 6 条车道(Traffic lanes)(可知龙门架是整体类，车道是局部类，它们之间是组合的关联关系),每条车道上安装有雷达传感器(Radar sensor)、无线传输器(Radio transceiver)和数码相机(Digital Camera)等用于不停车收费的设备(可知车道是整体类，雷达传感器、无限传输器、数码相机是局部类，它们之间是组合关系。)，以完成正常行驶速度下的收费工作。”，在对比图 3-2 知 C2 为龙门架“Toll gantry”，C3 为车道“Traffic lanes”，C4 为雷达传感器“Radar sensor”，C5 为无线传输器“Radio transceiver”，C6 为数码相机“Digital Camera”，其中 C4、C5、C6 可交换。对于【问题 2】，虽然本题并未明确要求使用英文类名作答，但由于图 3-2 中已有的类名为英文，故本题的答案建议写英文类名，当然，中文类名也不错。

从文字描述“(2)当汽车通过某条车道时，不停车收费设备识别车载器内的特有编码，判断车型，将收集到的相关信息发送到该路段所属的区域系统(Regional center)中，计算通行费用，创建收费交易(Transaction)…”可看出，车道“Traffic lanes”类(C3)与交易“Transaction”类是有关联关系的，这与图 3-2 中 C3 与类 Transaction 的“1：\*”的关系相符；同时也能看出一个区域系统(Regional center)覆盖多个路段，而一个路段有多条车道(Traffic lanes)，一个车道上方又安装有对应的龙门架，这与图 3-2 中类“Regional center”和类“Toll gantry”的“1：\*”的关联关系相符。

“…计算通行费用，创建收费交易(Transaction)，从驾驶员的专用账户中扣除通行费用。如果驾驶员账户透支,则记录透支账户交易信息。…”中“如果”两字表明系统在执行用例“记录收费交易(Create transaction)”时，用例“记录透支账户交易信息(Underpaid transaction)”有可能执行，也可能不执行。这说明用例“Create transaction”是被扩展用例，而“Underpaid transaction”是扩展用例，再对照图 3-1 可知 U3 为“Create transaction”，U1 为“Underpaid transaction”，注意，在用例的扩展关系中，箭头指向“被扩展”的用例。

通过 “…区域系统再将交易后的账户信息发送到维护驾驶员账户信息的中心系统(Central system)…”可知区域系统(Regional center)与中心系统(Central system)是多对 1 的关联关系，这与图 3-2 中 C1 处对比知 C1 为类“Central system”。

通过以上分析，用例“记录收费交易(Create transaction)”需要从从驾驶员的专用账户中扣除通行费用，而驾驶员账户信息由中心系统(Central system)进行维护，故图 3-1 中与用例 U3“记录收费交易(Create transaction)”相关联的参与者 A1 为“Central system”。

文字描述“(3)车载器中的磁卡可以使用邮局的付款机进行充值。充值信息会传送至中心系统,以更新驾驶员账户的余额。”表明“中心系统(Central system)”与用例“此卡充值(Charge card)”有关联关系，而充值应该是驾驶员(Driver)自行完成，故图 3-1 中 A2 应为参与者“驾驶员(Driver)”，用例 U5 为“此卡充值(Charge card)”。A1 与 A2 的答案可交换。文字描述“(4)当没有安装车载器或者车载器发生故障的车辆通过车道时，车道上的数码相机将对车辆进行拍照，并将车辆照片及拍摄时间发送到区域系统，记录失败的交易信息；并将该交易信息发送到中心系统。”也表明 ETC 系统在记录交易信息时可能会出现失败的状况，故用例“记录失败交易信息(Record Illegal use)”应对应图 3-1 中的 U2。U1 与 U2的答案可交换。

文字描述“(5)区域系统会获取不停车收费设备所记录的交通事件(Traffic events);交通广播电台(Traffic advice center)根据这些交通事件进行路况分析并播报路况。”表明用例“记录交通事件(Record traffic event)”与两个参与者有关联关系，一个是“交通广播电台(Traffic advice center)”，另一个是“区域系统(Traffic advice center)”，再对照图 3-1 可知，U4 为“Record traffic event”，A3 为“Traffic advice center”，A4 为“Traffic advice center”，其中 A3 与 A4 可交换。对于【问题 1】中每个参与者的答案，比特建议写英文名称，当然写中文名称也不错，因为题目并未规定，但是对于【问题 2】中的用例名一定要写英文名称，因为表 3-1 已经明确了用例名为英文名称。

要想正确地解答本题，需要掌握面向对象的分析与设计思想和 UML 建模语言知识，同时还需要有丰富的解题经验。对于本题，有解题经验的朋友会发现，表 3-1 中用例名为英文名称，图 3-2 中已经有名称的类也是英文名称可得出的结论是用例图 3-1 和类图 3-2 中需要填入的内容也应该是英文名称，那么答案就应该缩小到本题文字描述中具有英文名称的关键字部分，这就一下子就缩小了答案的搜索范围呢！！

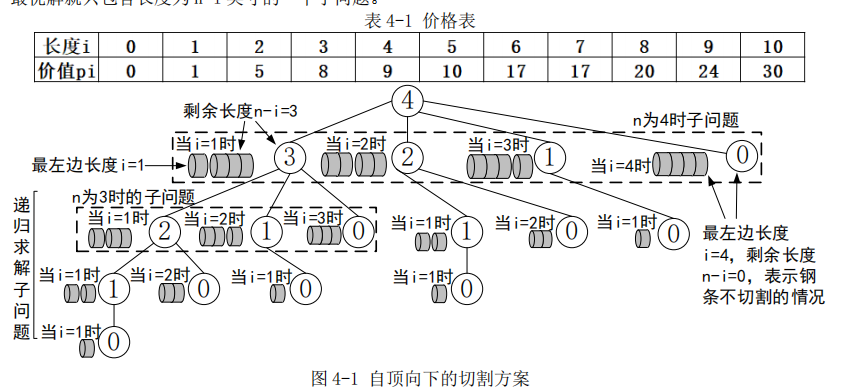
**试题四 【参考答案】**

(1)i<=n (2)i<=j (3)temp<p[i]+r[j-i]?p[i]+r[j-i]:temp (4)r[j]=temp; (5)动态规划 (6)O(2𝑛) (7)O(𝑛2)

【试题解析】

本题要解决钢条如何切割从而获取最大价值的问题，这是一个最优化问题。假设钢条切割价值类似表 4-1 所示，对于长度为 n 英寸的钢条，用rn表示最大收益，则rn=max{r1+rn−1, r2+rn−2,…,rn−1+r1, pn}，其中pn表示不切割钢条的情况，其他式子表示切成两段，分别长 i 英寸和 n-i 英寸，接着求解这两段的最优切割收益。为了求得原问题的解，先求性质完全一样但规模更小的子问题，即完成首次切割后，将切割出的两部分分别求解后再组合子问题的最优解，在所有可能的两段切割方案中选取组合收益最大的构成原问题的最优解，据此分析本题的钢条切割问题有最优子结构性质，这适合使用动态规划算法。

根据本题题意，将钢条从左边切割下长度为 i 英寸的一段，然后只对剩下的右边长度为 n-i英寸的另一段进行再切割 (递归求解)，对左边的一段则不再进行切割，所以rn=max1≤i≤𝑛(pi + rn−i)，即左边一段的价值加上右边的最大收益，这种情况下，最优解就只包含长度为 n-i 英寸的一个子问题。



如图4-1为长度为4英寸的钢条自顶向下的切割方案树。将长度为4英寸的钢条切成两段的情况有四种：

1.当i=1时，即最左边切割段长度为1英寸，右边段长度为3英寸(n-i=4-1=3)，获取的收益为p1+r4−1，而r4−1的最大收益还需要递归求解;

2.当i=2时，即最左边切割段长度为2英寸，右边段长度为2英寸(n-i=4-2=2)，获取的收益为p2+r4−2，而r4−2的最大收益还需要递归求解;

3.当i=3时，即最左边切割段长度为3英寸，右边段长度为1英寸(n-i=4-3=1)，获取的收益为p3+r4−3，而r4−3的最大收益还需要递归求解;

4.当i=4时，即最左边切割段长度为4英寸，右边段长度为0英寸(n-i=4-4=0)，获取的收益为p4，这种情况表示钢条不切割，获取的收益即p4，当然，这种场景也无须再递归求解。

以上的四种情况需要一个循环来实现，且能看出i的取值范围是1～4，比较4种情况的收益获得最大收益(这与空(1)处的循环体代码对应)。当钢条的长度为n英寸时，则i的取值范围是1～n，这与试题空(1)处的代码对应，故空(1)填写“i<=n”。

从图4-1可看出，算法从最长的长度n开始依次将规模大的问题划分为规模(n-i)较小的子问题，然后自顶向下，依次往深度方向递归求解子问题的解，图中的叶子节点个数即为算法的时间复杂度，长度为 4 英寸的叶子节点有24−1=8，则长度为n英寸的叶子节点个数为2𝑛−1个，故空(6)为“O(2𝑛)”。从图中易看出存在相同子问题的重复求解现象，如根为2的子树就有两棵(表示长度为2英寸的同一子问题求解了两次)，根为1的子树有4棵(表示长度为1英寸的同一子问题求解了4次)，要避免相同子问题的重复求解需要使用动态规划算法。

试题中Bottom\_Up\_Cut\_Rod(int p[], int n)函数使用数组r[LEN]保存了每个子问题的收益值，消除了同一子问题的重复求解过程。此函数的求解过程如下：

1.当j=1 时，求长度为1英寸的钢条切割最优收益值，只有1种切割方案，即不切割,最左边长度为 i=1英寸，剩余的右边段长度为 j-i=1-1=0 英寸，获得的收益为p1+𝑟𝑗−𝑖=p1+𝑟0 =1+0=1，保存到 r[j]即r[1]中，值为 1；

2.当 j=2 时，有两种切割方案(求解规模为 2 的子问题)：

(1)最左边长度为 i=1 英寸，剩余的右边段长度为 j-i=2-1=1 英寸，获得的收益为p1+𝑟𝑗−𝑖 =p1+𝑟1 =1+1=2；

(2)最左边长度为 i=2=j 英寸，剩余的右边段长度为 j-i=2-2=0英寸(不切割)，获得的收益为p2 + 𝑟𝑗−𝑖=p2+𝑟2−2 = 5+0=5。

对比两种方案的收益值，取最大的一个值 5 存入r[j]，即r[2]中。

3.当 j=3 时，有三种切割方案(求解规模为 3 的子问题)：

(1)最左边长度为 i=1 英寸，剩余的右边段长度为 j-i=3-1=2 英寸，获得的收益为p1+r𝑗−𝑖 =p1+r3−1=p1+r2=1+5=6；

(2)最左边长度为 i=2 英寸，剩余的右边段长度为 j-i=3-2=1 英寸，获得的收益为p2+r𝑗−𝑖 =p2+r3−2=p2+r1=5+1=6；

(3)最左边长度为 i=3=j 英寸，剩余的右边段长度为 j-i=3-3=0 英寸(不切割)，获得的收益为p3+r𝑗−𝑖=p3+r3−3=p3+r0=8+0=8；

对比三种方案的收益值，取最大的一个值 8 存入 r[j]，即 r[3]中。

使用如上的算法步骤可依次推导出规模为 n-1 问题的解，最后求出规模为 n 的大问题的最优值，最优值保存在 r[n]中，这种顺序即是自底向上的求解过程。

根据以上的分析，j 的取值范围是 1～n(表示问题的规模)，而 i 表示每个规模为 j 的子问题对应的最左边段长度，i的取值范围为 1～j。以上的步骤需要双重循环来实现，对应空(2)、(3)、(4)处的代码，空(2)为“i<=j”，空(2)处的循环是求规模为 j 的子问题的最优值，需要比较每个切割方案(i=1、i=2、i=3、…、i=j)的收益并将最大的值保存到临时变量 temp 中，故空(3)填“temp<p[i]+r[j-i]?p[i]+r[j-i]:temp”，即实现rn=max1≤i≤𝑛 (pi + rn−i)，规模为j的子问题最优值需要保存到数组元素 r[j]中，故空(4)填写“r[j]=temp”。

Bottom\_Up\_Cut\_Rod(int p[], int n)函数的运行时间主要花费在循环体中的双重循环处，当 j=1 时，i 的范围是 1；当 j=2 时，i 的范围是 1～2；当 j=3 时，i 的取值范围是 1～3；依次类推，当 j=n 时，i 的取值范围是 1～n，故循环的次数为 1+2+3+4+…+n=(1+n)\*n/2，故空(7)填写“O(𝑛2)”。

以上的算法步骤中，由于每个子问题的最优值都保存在数组 r 中，故求规模更大的子问题最优值时，对于需要用到的较小规模子问题的最优值无须再次求解，只需直接从 r 数组中读取即可，如求解规模为 3 且左边段 i=1 时的收益值为p1+r𝑗−𝑖=p1+r3−1=p1+r2，其中r2为规模为 2 的子问题的最优值(在之前的步骤中已经求出并保存起来了)，这里只需要直接读取值参与运算即可，这既是动态规划策略思想，故空(5)填写“动态规划”。

本题对应的完整程序如下：

#include<stdio.h>

#define LEN 100

int Top\_Down\_Cut\_Rod(int p[], int n) {/\*自顶向下\*/

int r = 0;

int i;

if (n == 0) { return 0; }

for (i = 1; i <= n; i++) {// (1)

int tmp = p[i] + Top\_Down\_Cut\_Rod(p, n - i);

r = (r >= tmp) ? r : tmp;

}

return r;

}

int Bottom\_Up\_Cut\_Rod(int p[], int n) {/\*自底向上\*/

int r[LEN] = { 0 };

int temp = 0;

int i, j;

for (j = 1; j <= n; j++) {//(2)

temp = 0;

for (i = 1; i <= j; i++) {

temp = temp < p[i] + r[j - i] ? p[i] + r[j - i] : temp; //(3)

}

r[j] = temp; //(4)

}

return r[n];

}

void main()

{

int p[11] = { 0,1,5,8,9,10,17,17,20,24,30 }; //切割价值表

int r; //保存最优值

//r = Top\_Down\_Cut\_Rod(price, 4);

r = Bottom\_Up\_Cut\_Rod(price, 4); //求长度为 4 英寸的钢条切割最优值

printf("最大切割价值为：%d", r);

getchar();

}

输出结果：最大切割价值为：10

●试题五 参考答案及解析

【参考答案】

（1）virtual void buildPartA()=0; （2）virtual Product\* getResult()=0;（3）product->setPartA

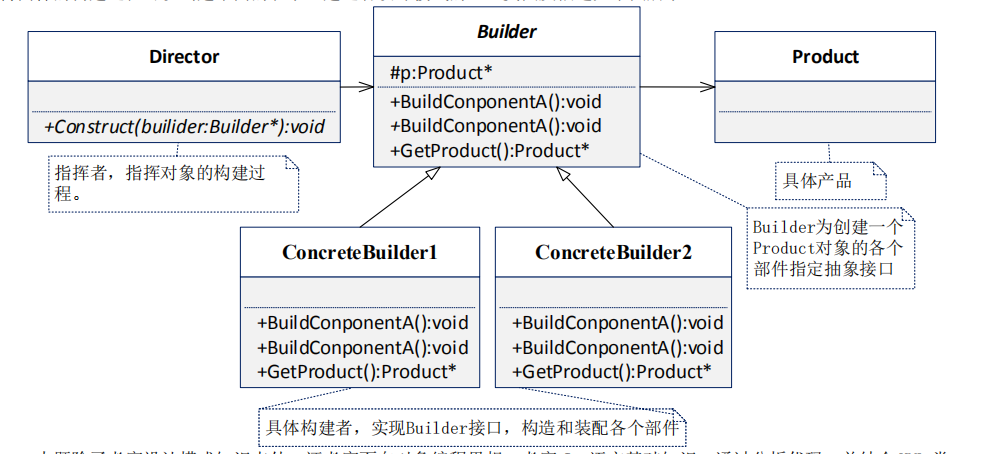
（4）product->setPartB（5）builder->buildPartA() 或 builder->buildPartB()

（注意，其中(1)与(2)可交换）

【试题解析】

本题考察建造者设计模式。建造者模式(Builder)又叫生成器模式，其将一个复杂对象的构建过程与它的表示分离，使

得同样的构建过程可以创建不同的表示。建造者设计模式的 UML 类图及描述如下图所示。



本题除了考察设计模式知识点外，还考察面向对象编程思想，考察 C++语言基础知识。通过分析代码，并结合 UML 类图 5-1 知 Builder 为两个派生类 ConcreteBuilder1 和 ConcreteBuilder2 的共同父类型，派生类中共同的成分(包括数据成员、函数成员，甚至是相同的函数头或函数体中相同的部分代码)都需提取到父类型中去。观察父类型中有一纯虚函数 buildPartB()(派生类中都已经实现，但函数体不一样)，则 buildPartA()在父类型中也应该被定义为纯虚函数，故空(1)填写 “virtual void buildPartA()=0”。同理，派生类中 getResult()也应该在父类型被定义为纯虚函数，空(2)填写“virtual Product\* getResult()=0”。

对于空(3)与空(4)，则使用比特培训课堂邓帅经常提到的“帅哥拥有富婆”的故事轻松搞定。帅哥“ConcreteBuilder1”类为啥要拥有富婆“product”(Product 类对象)作为他的数据成员呢？就是因为帅哥“ConcreteBuilder1”类要实现功能 buildPartA()但是又不想自己写代码实现，而让其拥有的富婆“product”替其实现，故空(3)应该填写“product->setPartA”，犹如你拥有了富婆，你就可花富婆的钱一样，否则帅哥找富婆干么？世间哪有那么多纯洁的爱情？！。不开玩笑，让我们严谨地分析！因为 ConcreteBuilder1 类的成员对象 product 已经实现了构建组件 A 的功能(setPartA 函数)，

故 ConcreteBuilder1 实现 buildPartA()时可调用其成员对象的功能函数 setPartA 来实现，避免同样的功能重复编码现象，这种是通过组合方式实现功能复用的场景。同理，空(4)应该填写“product->setPartB”。

在构建者设计模式中，为了减轻客户端的压力(可理解为本题的 main 函数)，让客户端无须关心对象的构建过程和构建细节，降低客户端和构建者之间的耦合度，需要引入指挥者类 Director，让指挥者类负责指挥对象的统一构建。指挥者类中的 construct()函数主要负责指挥完成对象的构建过程，而指挥者 Director拥有了一个 “builder”。

要实现的功能 construct()应该由其拥有的成员“builder”来实现，因为“builder”已经完成了 partA 和 partB的具体实现，故空(5)应为“builder->buildPartA()”或“builder->buildPartB()”。

●试题六 参考答案及解析

【参考答案】

（1）abstract void buildPartA() 或 void buildPartA()

（2）abstract Product getResult() 或 Product getResult()

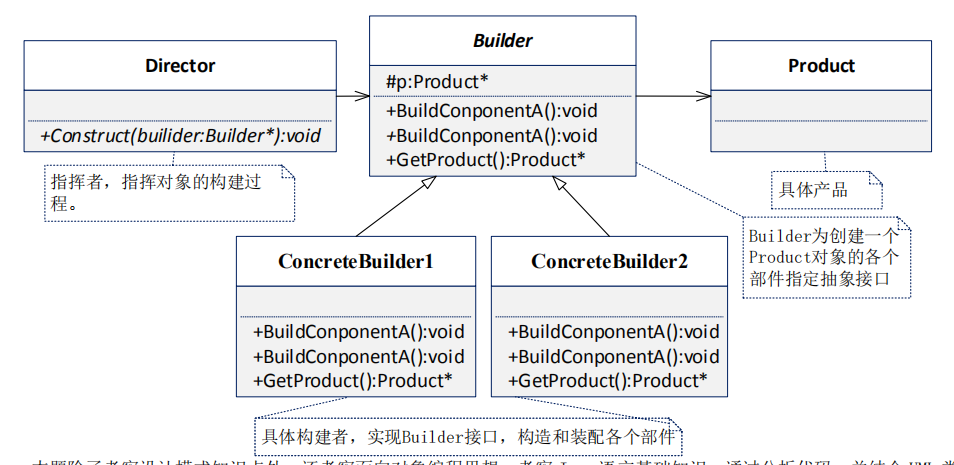
（3）product.setPartA （4）product.setPartB

（5）builder.buildPartA() 或 builder.buildPartB()

（注意，其中(1)与(2)可交换）

【试题解析】

本题考察建造者设计模式。建造者模式(Builder)又叫生成器模式，其将一个复杂对象的构建过程与它的表示分离，使得同样的构建过程可以创建不同的表示。建造者设计模式的 UML 类图及描述如下图所示。



本题除了考察设计模式知识点外，还考察面向对象编程思想，考察 Java 语言基础知识。通过分析代码，并结合 UML 类

图 6-1 知 Builder 为两个派生类 ConcreteBuilder1 和 ConcreteBuilder2 的共同父类型，派生类中共同的成分(包括数据成员、函数成员，甚至是相同的函数头或函数体中相同的部分代码)都需提取到父类型中去。观察父类型中有一抽象函数“public void buildPartB()”(派生类中都已经实现，但函数体不一样)，则 buildPartA()在父类型中也应该被定义为抽象函数，故空(1)填写“abstract void buildPartA()”，由于接口中 public 和 abstract 两关键字都可省略，故也可写成答案 “void buildPartA()”。同理，派生类中 getResult()也应该在父类型被定义为抽象函数，空(2)填写“abstract Product getResult()”或“Product getResult()”。

对于空(3)与空(4)， “ConcreteBuilde1”类为啥要拥有 “product”(Product 类对象)作为他的数据成员呢？就是因为 “ConcreteBuilder1”类要实现功能 buildPartA()但是又不想自己写代码实现，而让其拥有 “product”替其实现，故空(3)应该填写“product.setPartA”，因为 ConcreteBuilder1 类的成员对象 product 已经实现了构建组件 A 的功能(setPartA 函数)，故 ConcreteBuilder1 实现 buildPartA()时可调用其成员对象的功能函数 setPartA 来实现，避免同样的功能重复编码现象，这种是通过组合方式实现功能复用的场景。同理，空(4)应该填写“product.setPartB”。

在构建者设计模式中，为了减轻客户端的压力(可理解为本题的 Test 类)，让客户端无须关心对象的构建过程和构建细节，降低客户端和构建者之间的耦合度，需要引入指挥者类 Director，让指挥者类负责指挥对象的统一构建。指挥者类中的 construct()函数主要负责指挥完成对象的构建过程，而指挥者 Director拥有了一个 “builder”，故要实现的功能 construct()应该由其拥有的成员“builder”来实现，因为“builder”已经完成了 partA 和 partB的具体实现，故空(5)应为“builder.buildPartA()”或“builder.buildPartB()”。