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 2016 年 上半年 下午试卷 案例

（考试时间 150 分钟）

试题一 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】

某会议中心提供举办会议的场地设施和各种设备，供公司与各类组织机构租用。场地包括一个大型报告厅、一个小型报告厅以及诸多会议室。这些报告厅和会议室可提供的设备有投影仪、白板、视频播放/回放设备、计算机等。为了加强管理，该中心欲开发一会议预订系统，系统的主要功能如下。

(1）检查可用性。客户提交预订请求后，检查预订表，判定所申请的场地是否在申请日期内可用；如果不可用，返回不可用信息。

(2）临时预订。会议中心管理员收到客户预定请求的通知之后，提交确认。系统生成新临时预订存入预订表，并对新客户创建一条客户信息记录加以保存。根据客户记录给客户发送临时预订确认信息和支付定金要求。

(3）分配设施与设备。根据临时预订或变更预定的设备和设施需求，分配所需设备(均能满足用户要求)和设施，更新相应的表和预订表。

(4）确认预订。管理员收到客户支付定金的通知后，检查确认，更新预订表，根据客户记录给客户发送预订确认信息。

(5）变更预订。客户还可以在支付余款前提交变更预订请求，对变更的预订请求检查可用性，如果可用，分配设施和设各；如果不可用，返回不可用信息。管理员确认变更后，根据客户记录给客户发送确认信息。

(6）要求付款。管理员从预订表中查询距预订的会议时间两周内的预定，根据客户记录给满足条件的客户发送支付余款要求。

(7）支付余款。管理员收到客户余款支付的通知后，检查确认，更新预订表中的已支付余款信息。

现采用结构化方法对会议预定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(不完整)。

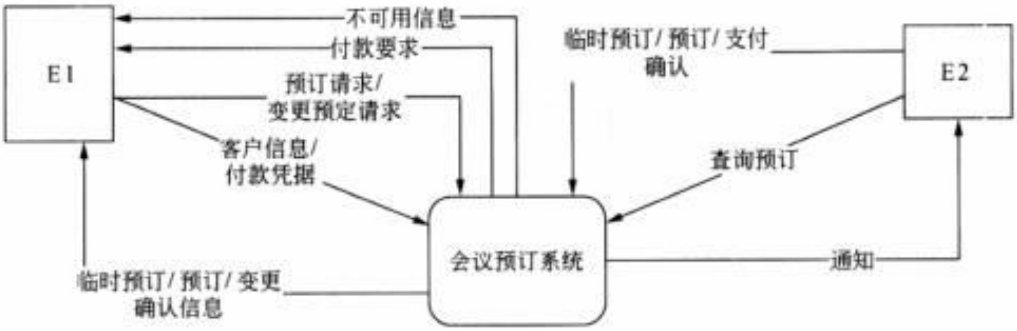


图 1-1　上下文数据流图

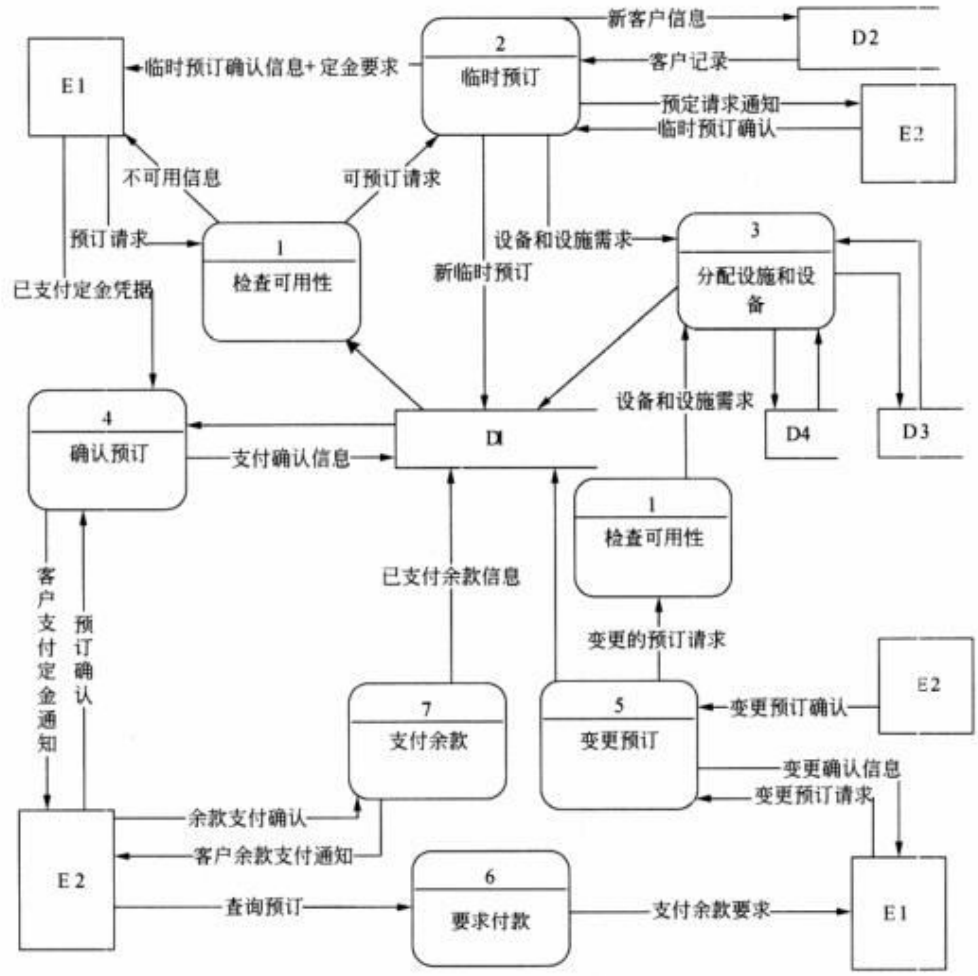


图 1-2　0 层数据流图

问题：1.1

使用说明中的词语，给出图 1-1 中的实体 E1 ～ E2 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

根据说明和图中术语，补充图 1-2 之中缺失的数据流及其起点和终点。

问题：1.4
如果发送给客户的确认信息是通过 Email 系统向客户信息中的电子邮件地址进行发送的，那么需要对图 1-1 和 1-2 进行哪些修改？用 150 字以内文字加以说明。

试题二 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】

某销售公司当前的销售业务为商城实体店销售。现该公司拟开展网络销售业务，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）记录公司所有员工的信息。员工信息包括工号、身份证号、姓名、性别、出生日期和电话，并只登记一部电话。

(2）记录所有商品的信息。商品信息包括商品名称、生产厂家、销售价格和商品介绍。系统内部用商品条码唯一区别每种商品。

(3）记录所有顾客的信息。顾客信息包括顾客姓名、身份证号、登录名、登录密码、和电话号码。一位顾客只能提供一个电话号码。系统自动生成唯一的顾客编号。

(4）顾客登录系统之后，在网上商城购买商品。顾客可将选购的商品置入虚拟的购物车内，购物车可长期存放顾客选购的所有商品。顾客可在购物车内选择商品、修改商品数量后生成网购订单。订单生成后，由顾客选择系统提供的备选第三方支付平台进行电子支付，支付成功后系统需要记录唯一的支付凭证编号，然后由商城根据订单进行线下配送。

(5）所有的配送商品均由仓库统一出库。为方便顾客，允许每位顾客在系统中提供多组收货地址、收货人及联系电话。一份订单所含的多个商品可能由多名分检员根据商品所在仓库信息从仓库中进行分拣操作，分拣后的商品交由配送员根据配送单上的收货地址进行配送。

(6）新设计的系统要求记录实体店的每笔销售信息，包括营业员、顾客、所售商品及其数

量。

【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，出生日期，电话)

商品(商品条码，商品名称，生产厂家，销售价格，商品介绍， (a))

顾客(顾客编号，姓名，身份证号，登录名，登录密码，电话)

收货地点(收货 ID，顾客编号，收货地址，收货人，联系电话)

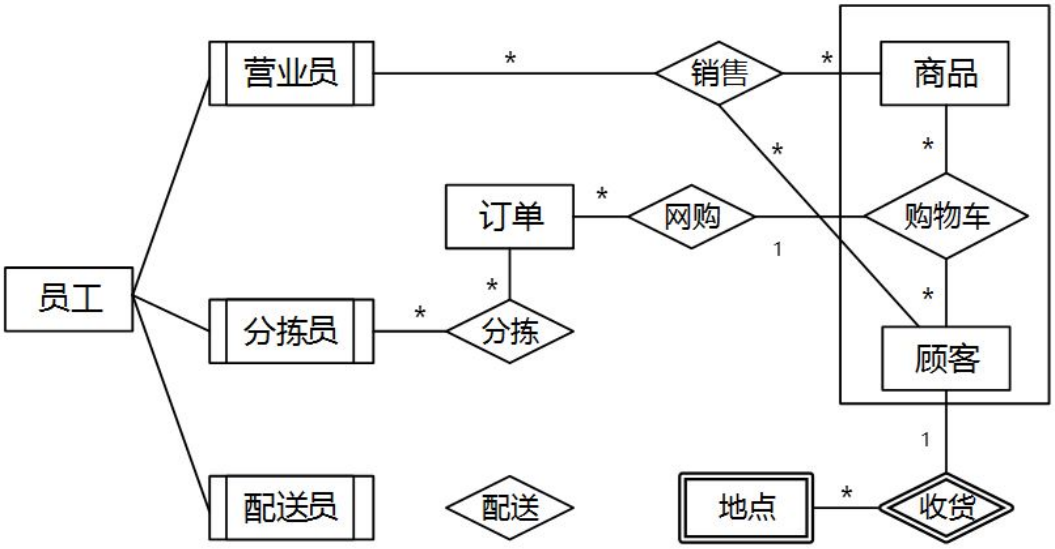
购物车(顾客编号，商品条码，商品数量)

订单(订单 ID，顾客编号，商品条码，商品数量， (b))

分检(分拣 ID，分拣员工号， (c) ，分拣时间)

配送(配送 ID，分拣 ID，配送员工号，收货 ID，配送时间，签收时间，签收快照)

销售(销售 ID，营业员工号，顾客编号，商品条码，商品数量)



问题： 2.1

补充图 2-1 中的“配送”联系所关联的对象及联系类型。

问题： 2.2

补充逻辑结构设计中的(a)、(b)和(c)三处空缺。

问题： 2.3

对于实体店销售，若要增加送货上门服务，由营业员在系统中下订单，与网购的订单进行后续的统一管理。请根据该需求，对图 2-1 进行补充，并修改订单关系模式。

试题三 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】

某软件公司欲设计实现一个虚拟世界仿真系统。系统中的虚拟世界用于模拟现实世界中的不同环境(由用户设置并创建)，用户通过操作仿真系统中的 1~2 个机器人来探索虚拟世界。机器人维护着两个变量 b1 和 b2，用来保存从虚拟世界中读取的字符。

该系统的主要功能描述如下：

(1) 机器人探索虚拟世界(RunRobots)。用户使用编辑器(Editor)编写文件以设置想要模拟的环境，将文件导入系统(LoadFile)从而在仿真系统中建立虚拟世界(SetupWorld)。机器人在虚拟世界中的行为也在文件中进行定义，建立机器人的探索行为程序(SetupProgram)。机器人在虚拟世界中探索时(RunProgram)，有 2 种运行模式：

①自动控制(Run)：事先编排好机器人的动作序列(指令(Instruction))，执行指令，使机器人可以连续动作。若干条指令构成机器人的指令集(InstructionSet)。

②单步控制(Step)：自动控制方式的一种特殊形式，只执行指定指令中的一个动作。

(2) 手动控制机器人(ManipulateRobots)。选定 1 个机器人后(SelectRobot)，可以采用手动方式控制它。手动控制有 4 种方式：

①Move：机器人朝着正前方移动一个交叉点。

②Left：机器人原地沿逆时针方向旋转 90 度。

③Read：机器人读取其所在位置的字符，并将这个字符的值赋给 b1；如果这个位置上没有字符，则不改变 b1 的当前值。

④Write：将 b1 中的字符写入机器人当前所在的位置，如果这个位置上已经有字符，该字符的值将会被 b1 的值替代。如果这时 b1 没有值，即在执行 Write 动作之前没有执行过任何 Read 动作，那么需要提示用户相应的错误信息(ShowErrors)。

手动控制与单步控制的区别在于，单步控制时执行的是指令中的动作，只有一种控制方式，即执行下个动作；而手动控制时有 4 种动作。

现采用面向对象方法设计并实现该仿真系统，得到如图 3-1 所示的用例图和图 3-2 所示的初始类图。图 3-2 中的类“Interpreter”和“Parser”用于解析描述虚拟世界的文件以及机器人行为文件中的指令集。

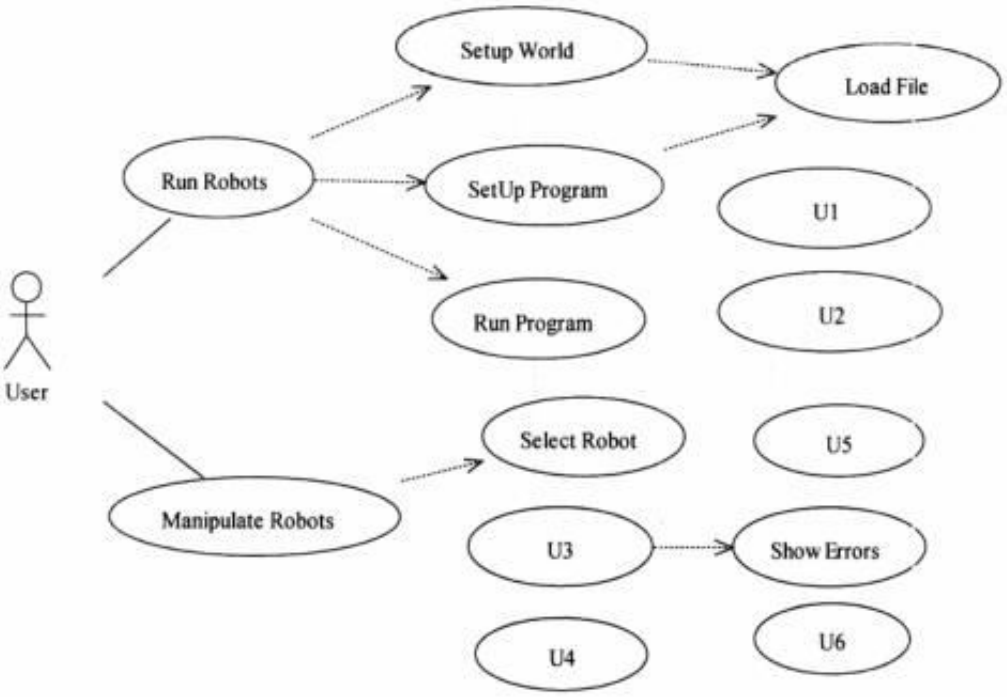


图 3-1　用例图

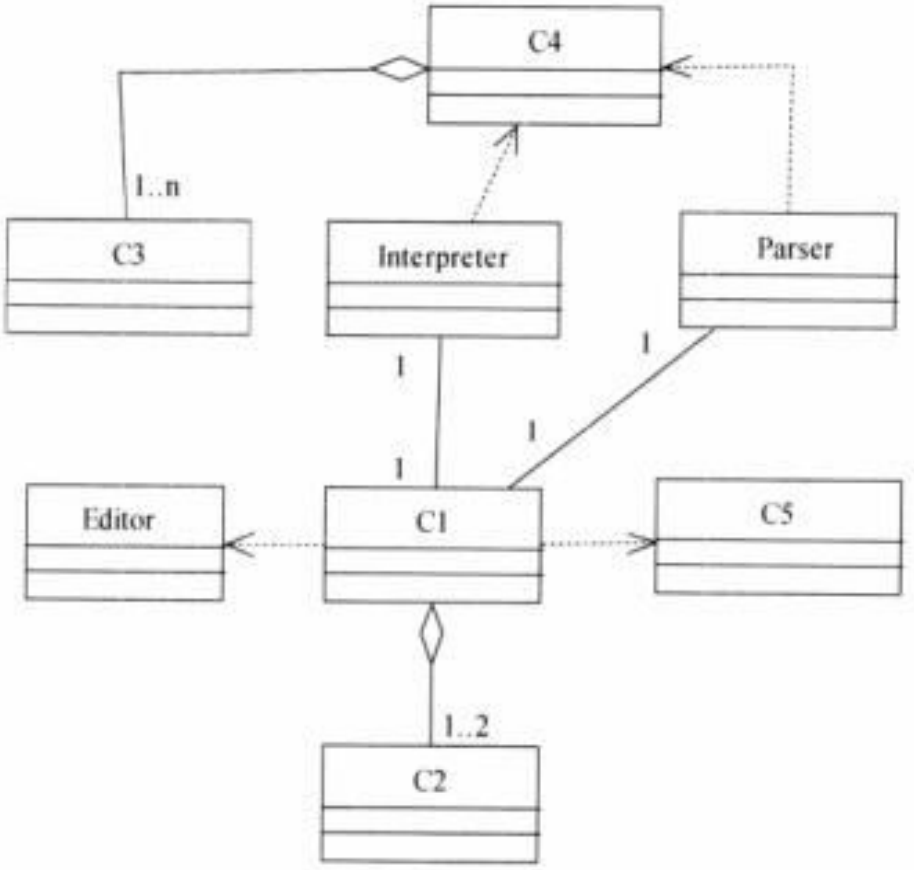


图 3-2　初始类图

问题：3.1
根据说明中的描述，给出图 3-1 中 U1 ～ U6 所对应的用例名。
问题：3.2
图 3-1 中用例 U1 ～ U6 分别与哪个(哪些)用例之间有关系，是何种关系？
问题：3.3
根据说明中的描述，给出图 3-2 中 C1 ～ C5 所对应的类名。

试题四 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】

在一块电路板的上下两端分别有 n 个接线柱。根据电路设计，用(i,π(i))表示将上端接线柱 i 与下端接线柱 π(i)相连，称其为该电路板上的第 i 条连线。如图 4-1 所示的 π(i)排列为{8,7,4,2,5,1,9,3,10,6}。对于任何 1≤i π(j)。

在制作电路板时，要求将这 n 条连线分布到若干绝缘层上，在同一层上的连线不相交。现在要确定将哪些连线安排在一层上，使得该层上有尽可能多的连线，即确定连线集 Nets ={(i,π(i))， 1≤i≤n}的最大不相交子集。

【分析问题】

记 N(i,j)={t|(t,π(t))∈Nets,t≤i,π(t)≤j}。 N(i,j)的最大不相交子集为 MNS(i,j)，size(i,j)=|MNS(i,j)|。

经分析，该问题具有最优子结构性质。对规模为 n 的电路布线问题，可以构造如下递归式：

【C 代码】

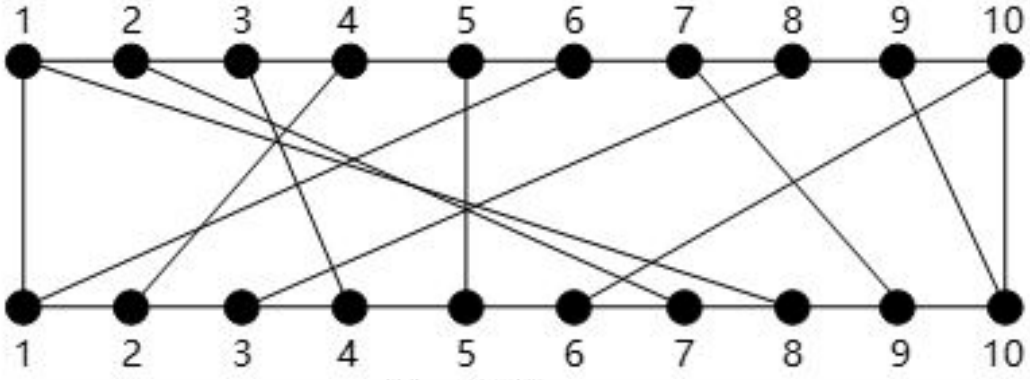
下面是算法的 C 语言实现。

(1) 变量说明

size[i][j]：上下端分别有 i 个和 j 个接线柱的电路板的第一层最大不相交连接数

pi[i]： π(i)，下标从 1 开始

(2) C 程序



（1）当 i = 1 时， $size(1,j)=\begin{cases}0 & j<\pi(1)\\ 1 & \text{其他情况}\end{cases}$

（2）当 i > 1 时， $size(i,j)=\begin{cases}size(i-1,j) & j<\pi(i)\\ \max\{size(i-1,j),size(i-1,\pi(i)-1)+1\} & \text{其他情况}\end{cases}$

```
#include "stdlib.h"
#include <stdio.h>
#define   N   10          /* 问题规模 */
int m = 0;                /* 记录最大连接集合中的接线柱 */
void maxNum(int pi[], int size[N + 1][N + 1], int n)  {  /* 求最大不相交
连接数 */
    int i, j;
    for(j = 0; j < pi[1]; j++)    size[1][j] = 0;     /* 当j < π(1)时 */
    for(j = pi[1]; j <= n; j++)   ____(1)____;        /* 当j >= π(1)时 */
    for(i = 2; i < n; i++)  {
       for(j = 0; j < pi[i]; j++)   ____(2)____;      /* 当j < p[i]时 */
       for(j = pi[i]; j <= n; j++)  {  /* 当j >= c[i]时，考虑两种情况 */

        size[i][j]=size[i-1][j]>=size[i-1][pi[i]-1]+1?size[i-1][j]:
        size[i-1][pi[i]-1]+1;
      }
}
/* 最大连接数 */
   size[n][n] = size[n-1][n] >= size[n-1][pi[n]-1]+1 ? size[n-1][n] :
   size[n-1][pi[n]-1]+1;
}
/* 构造最大不相交连接集合，net[i]表示最大不相交子集中第i条连线的上端接线柱的序号 */
void constructSet(int pi[], int size[N + 1][N + 1], int n, int net[n]) {
   int i, j = n;
   m = 0;
   for(i = n; i > 1; i--)      {     /* 从后往前 */
      if(size[i][j] != size[i-1][j]) {  /* (i, pi[i])是最大不相交子集的一
      条连线 */
         ____(3)____;                 /*将i记录到数组net中，连接线数自增1*/
         j = pi[i]-1;                 /* 更新扩展连线柱区间 */
      }
   }
   if(j >= pi[1])   net[m++] = 1;    /* 当i =1时 */
}
```

问题：4.1

根据以上说明和C代码，填充C代码中的空(1)～(3)。

问题：4.2

根据题干说明和以上C代码，算法采用了(4)算法设计策略。函数maxNum和constructSet的时间复杂度分别为(5)和(6)(用O表示)”

问题：4.3

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】

某软件系统中，已设计并实现了用于显示地址信息的类Address(如图5-1所示)，现要求提供基于Dutch语言的地址信息显示接口。为了实现该要求并考虑到以后可能还会出现新的语言的接口，决定采用适配器(Adapter)模式实现该要求，得到如图5-1所示的类图。

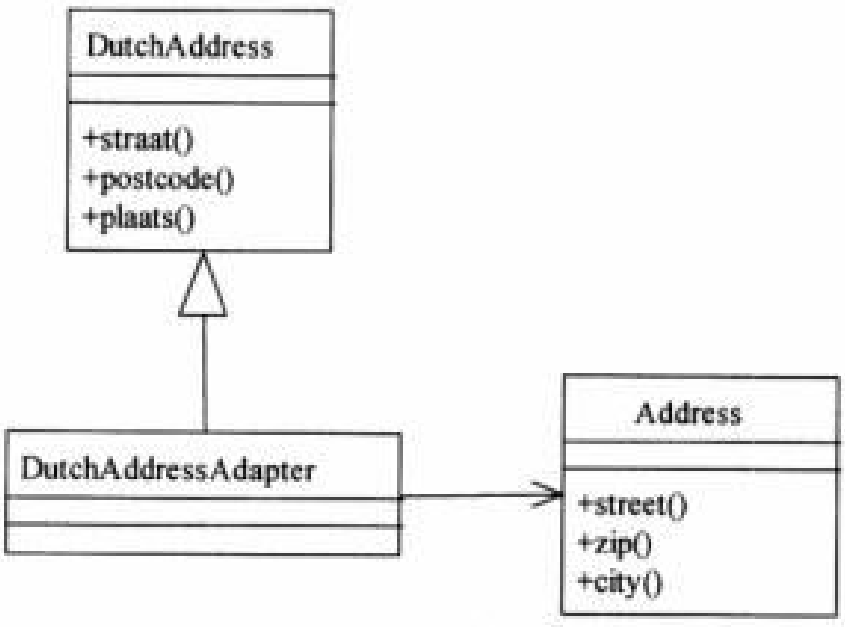


图5-1 适配器模式类图

问题：5.1

【C++代码】

```
#include <iostream>
using namespace std;

class Address {
public:
   void street()    {  /* 实现代码省略 */    }
   void zip()       {   /* 实现代码省略 */    }
   void city()      {   /* 实现代码省略 */    }
// 其他成员省略
};

class DutchAddress {
public:
   virtual void straat() = 0;
   virtual void postcode() = 0;
   virtual void plaats() = 0;
// 其他成员省略
};

class DutchAddressAdapter : public DutchAddress{
private:
   ____(1)____;
public:
    DutchAddressAdapter(Address *addr) {
        address = addr;
    }
    void straat() {
        ____(2)____;
```

```
    }
    void postcode() {
        ____(3)____;
    }
    void plaats() {
        ____(4)____;
    }
// 其他成员省略
};

void testDutch(DutchAddress *addr) {
        addr->straat();
        addr->postcode();
        addr->plaats();
}

int main() {
    Address *addr = new Address();
    ____(5)____;
    cout << "\n The DutchAddress\n" << endl;
    testDutch(addrAdapter);
    return 0;
}
```

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】

某软件系统中，已设计并实现了用于显示地址信息的类 Address(如图 6-1 所示)，现要求提供基于 Dutch 语言的地址信息显示接口。为了实现该要求并考虑到以后可能还会出现新的语言的接口，决定采用适配器(Adapter)模式实现该要求，得到如图 6-1 所示的类图。

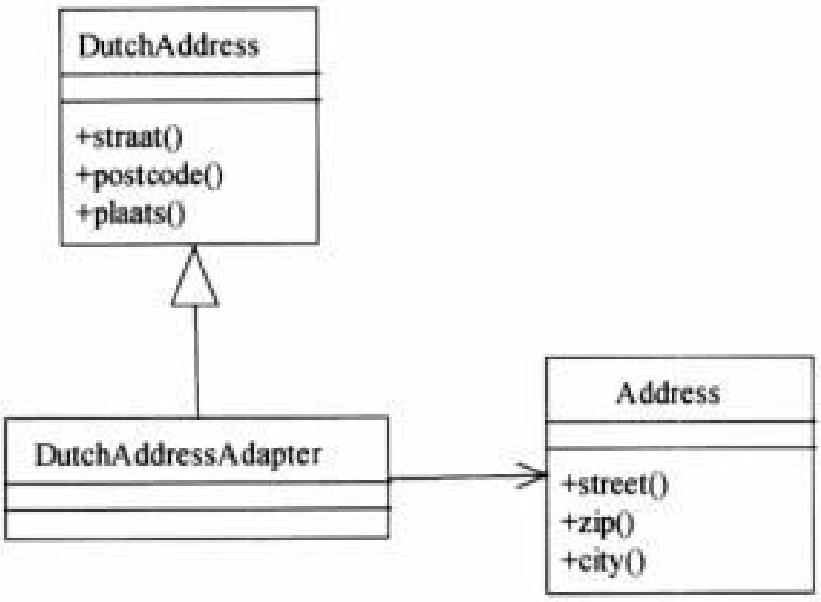


图 6-1　适配器模式类图

问题：6.1

【Java 代码】

```
import java.util.*;

class Address {
    public void street()    {    //  实现代码省略        }
    public void zip()       {    //  实现代码省略        }
    public void city()      {    //  实现代码省略        }
// 其他成员省略
}

class DutchAddress {
```

```
    public void straat()    { // 实现代码省略        }
    public void postcode()  { // 实现代码省略        }
    public void plaats()    { // 实现代码省略        }
// 其他成员省略
}

class DutchAddressAdapter extends DutchAddress {
    private ____(1)____;

    public DutchAddressAdapter(Address addr) {
        address = addr;
    }

    public void straat() {
        ____(2)____;
    }

    public void postcode() {
        ____(3)____;
    }

    public void plaats() {
        ____(4)____;
    }
// 其他成员省略
}

class Test {
    public static void main(String[] args) {
        Address addr = new Address();
        ____(5)____;
        System.out.println("\n The DutchAddress\n");
        testDutch(addrAdapter);
    }

   static void testDutch(DutchAddress addr) {
       addr.straat();
       addr.postcode();
       addr.plaats();
   }
}
```

试题一 答案：　解析：　E1：客户

E2：管理员

本题考查采用结构化方法进行系统分析与设计，主要考查数据流图(DFD)的应用，是比较传统的题目，考点与往年类似，要求考生细心分析题目中所描述的内容。

面向数据流建模是目前仍然被广泛使用的结构化分析与设计的方法之一，而 DFD 是面向数据流建模的重要工具，是一种便于用户理解、分析系统数据程的图形化建模工具，是系统逻辑模型的重要组成部分。 DFD 将系统建模成“输入一加工(处理)一输出”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，并采用分层的方式加以表示。

上下文 DFD(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(处理)，然后根据系统从哪些外部实体接收数据流，以及系统将数据流发送到哪些外部实体，建模出的上下文图中只有唯一的一个加工和一些外部实体，以及这两者之间的输入输出数据流。0 层 DFD 在上下文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输出数据流，使得所有上下文 DFD 中的输入数据流，经过这些加工之后变换成上下文 DFD 的输出数据流。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，注意要在绘制下层数据流图时要保持父图与子图平衡。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。

本题考查上下文 DFD，要求确定外部实体。在上下文 DFD 中，系统名称作为唯一加工的名称，外部实体和该唯一加工之间有输入输出数据流。通过考查系统的主要功能，不难发现，系统中涉及到客户和会议中心管理员，没有提到其他与系统交互的外部实体。根据描述(1)“客户提交预订请求后”，(2)“会议中心管理员收到客户预定请求的通知之后，提交确认”、“根据客户记录给客户发送临时预订确认信息和支付定金要求”等信息，对照图 1-1，从而即可确定 E1 为“客户”实体， E2 为“管理员”实体。

D1：预订表

D2：客户表

D3：场地表(设施表或场地设施表)

D4：设备表

注： D3 和 D4 可互换

本题要求确定图 1-2 所示的 0 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。根据(1)“客户提交预订请求后，检查预订表”，(2)“系统生成新临时预订存入预订表，并对新客户创建一条客户信息记录加以保存”，可知D1为预订表、D2为客户表；根据“会议中心提供举办会议的场地设施和各种设备”，(3)“根据临时预订或变更预定的设备和设施需求，分配所需设备(均能满足用户要求)和设施，更新相应的表和预订表”，“分配设施和设备”可知D3为和D4分别为场地(设施)表和设备表。

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，数量不同，考查图1-1中从加工“会议预订系统”输出至E1的数据流，有“临时预订/预订/变更确认信息”，而图1-2中从加工输出至E1的数据流“临时预订确认信息”和“变更预订确认信息”，但缺少了其中一条数据流“预订确认信息”。

另外，图1-1中有“付款凭据”，图1-2中没有“付款凭据”，而只有“已支付定金凭据”，没有针对说明(7)中“管理员收到客户余款支付的通知后”中的“支付余款凭据”。上述两条数据流的遗失，使父图和子图数据流没有达到平衡。所以需要确定这两条条数据流或者其分解的数据流的起点或终点。

考查说明中的功能，先考查“确认预定”，功能(4)中“给客户发送预订确认信息”，对照图1-2，加工4没有到实体E1客户的“预订确认信息”数据流；功能(7)中“管理员收到客户余款支付的通知后”，对照图1-2，加工7没有从实体E1客户输入的数据流“余款支付凭据”。图中“余款支付凭据”数据流是上下文数据流图中数据流“支付凭据”的分解，与另一条分解出的数据流“已支付定金凭据”对照，改名为“已支付余款凭据”。

下面再仔细核对说明和图1-2之间是否还有遗失的数据流。

不难发现，功能(4)中“根据客户记录给客户发送预订确认信息”，而图1-2中加工4从D1预订表中读取预订信息，并没有读取客户信息，所以，此处遗失了数据流“客户记录”，起点是D2客户表，终点是加工4确认预订；功能(5)中“管理员确认变更后，根据客户记录给客户发送确认信息”，而图1-2中加工5并没有所根据的“客户记录”输入数据流，所以，此处遗失了数据流“客户记录”，起点是D2客户表，终点是加工5变更预订；功能(6)中“根据客户记录给满足条件的客户发送支付余款要求”，而图1-2中加工6并没有所根据的“客户记录”输入数据流，所以，此处遗失了数据流“客户记录”，起点是D2客户表，终点是加工6要求预订。

继续核对说明和图1-2，不难发现，功能(6)中“管理员从预订表中查询距预订的会议时间两周内的预定”，而图1-2中没有从D1预订表到加工6的输入流，所以，此处遗失了数据流“距预订会议时间两周内的预订”，其起点是D1预订表，终点是加工6要求付款。

将Email系统作为外部实体，并将发送给客户(E1)的确认信息数据流的终点全部改为Email系统(或具体说明确认信息数据流：临时预订确认信息、预订确认信息、变更确认信

息，终点均改为 Email 系统)。

DFD 中，外部实体可以是用户，也可以是与本系统交互的其他系统。如果某功能交互的是外部系统(在本题中是 Email 系统)，则本系统需要将发送给客户的确认信息发送给 Email 系统。然后由第三方 Email 系统向客户发送邮件，此时第三方 Email 系统即为外部实体，而非本系统内部加工，因此需要对图 1-1 和图 1-2 进行修改，添加外部实体“Email 系统”，并将数据流确认信息的终点全部改为 Email 系统。即将数据流“临时预订确认信息”、“预订确认信息”、“变更确认信息”数据流的终点改为新的外部实体“Email 系统”。

数 据 流	起 点	终 点
已支付余款凭据	E1 或 客户	7 或 支付余款
距预订会议时间两周内的预订	D1 或 预订表	6 或 要求付款
预订确认信息	4 或 确认预订	E1 或 客户
客户记录	D2 或 客户表	6 或 要求付款
客户记录	D2 或 客户表	5 或 变更预定
客户记录	D2 或 客户表	4 或 确认预定

注：上述 6 条数据流无顺序要求。

试题二 答案：　解析：　补充内容如图中虚线所示：

本题考查数据库概念结构设计和逻辑结构设计。

此类题目要求考生认真阅读题目中的需求描述，配合已给出的 E-R 图，理解概念结构设计中设计者对实体及联系的划分和组织方法，结合需求描述完成 E-R 图中空缺部分，并使用 E-R 图向关系模式的转换方法，完成逻辑结构设计。

根据所给 E-R 图，结合需求描述，购物车作为顾客和商品之间的联系，而订单由顾客从购物车中选择商品生成，因此将购物车这一联系当作实体，与订单实体产生联系。将联系当作实体参与另一联系，称为聚合，通常当后一联系与此联系相关时，采用这种设计方法。顾客可以从购物车中生成多个订单，一个订单只能从一个购物车里提取商品，属于一对多联系。

根据需求描述中的“分拣后的商品交由配送员根据配送单上的收货地址进行配送。”可以知道，配送是与分拣联系相关的联系，同样的，将分拣联系进行聚合，参与配送联系，同时参与配送联系的还有配送员和地点，为多对多对多联系，语义为配送员根据分拣结果按照收货地点进行配送，与需求相符。

(a)所在仓库

(b)支付凭证

(c)订单 ID

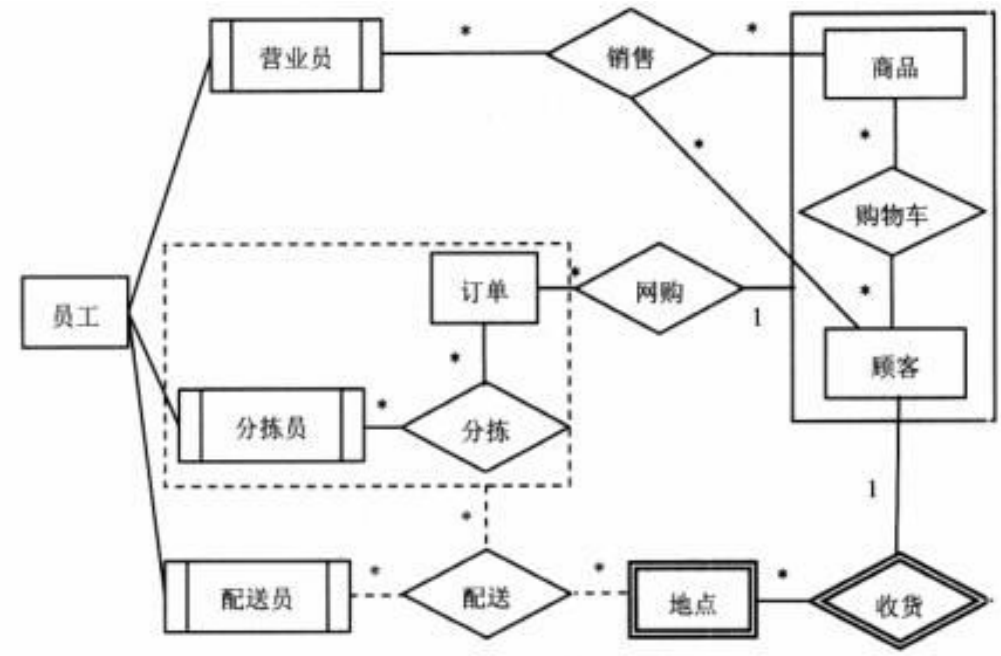
本小题考核 E-R 图向关系模式的转换。由于 E-R 图中没有画出实体及联系的属性，需要根据需求描述进行补充。根据需求中的“一种商品只能放在一个仓库中”和“一份订单所含的多个商品可能由多名分拣员根据商品的所在仓库信息从仓库中进行分拣操作”，可以确定“所在仓库”作为商品实体的属性，转入商品关系中。

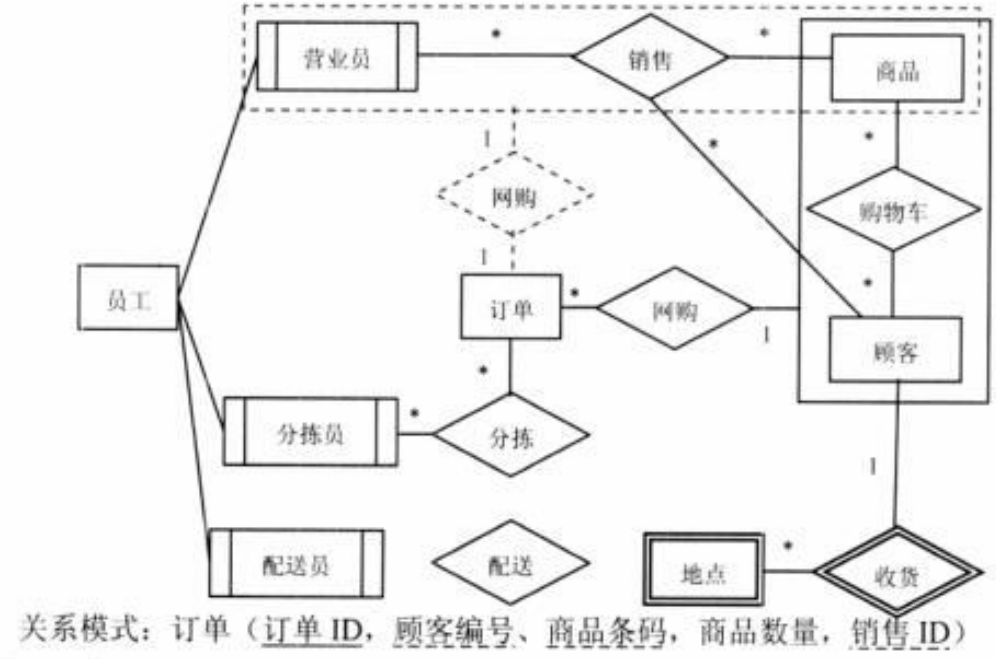
订单关系由 E-R 图中的订单实体和一对多联系网购合并而成，取一方的主码，即购物车这一联系的主码，为参与该联系的实体的主码商品条码和顾客编号，加上网购联系的属性数量，并入到订单实体转成的关系模式中。订单 ID 为订单实体的标识符，订单实体的其他属性需要通过需求描述中获取。根据需求“订单生成后，由顾客选择系统提供的备选第三方支付平台进行电子支付，支付成功后系统需要记录唯一的支付凭证编号”，支付凭证编号应为订单的属性，转入订单关系中。

E-R 图中的分拣联系为分拣员与订单之间的多对多联系，转换成独立的分拣关系模式，应包含分拣员实体的标识符分拣员工号和订单实体的标识符订单 ID，及分拣联系的属性分拣时间。

补充内容如图中虚线所示：

实体店的订单是营业员根据销售结果生成的，将销售联系聚合成实体，与订单产生联系。一笔销售对应一个订单，一个订单对应一笔销售，为一对一联系。转换为关系模式时，将此联系归入订单关系，即取销售的标识符销售 ID 加入到订单关系模式中。





关系模式：订单（订单 ID，顾客编号、商品条码，商品数量，销售 ID）

试题三 答案：　　解析：　U1:Run

U2:Step

U3:Write

U4:Read

U5:Left

U6:Move

注：U1 和 U2 可以互换；U4　U6 可以互换。

本题属于经典的考题，主要考查面向对象分析方法与设计的基本概念。在建模方面，本题涉及到了 UML 的类图和用例图。本题的考点比较常规，题目难度不大。

在图 3-1 的用例图中，需要确定 6 个用例。在面向对象方法中，用例及用例图是描述功能需求的工具，每个用例表示一个单一的功能单元。通过对【说明】中功能描述的阅读，可以将未出现在图 3-1 中的功能单元列举出来：Run 、Step 、Move 、Left 、Read 和 Write 。下面就是要判断这 6 个用例在图中的对应关系了。由图 3-1 可知，U3 中包含了 ShowErrors 的功能，所以 U3 只能对应用例“Write”。其余的没有严格的顺序要求，但是在回答【问题 2 】时要根据所填写的用例来判断用例之间的关系。这里我们按照下列顺序填写：U1->Run ；U2->Step ；U3->Write ；U4->Read ；U5->Left ；U6->Move 。

U1　U2 与 RunProgram 有关系；是泛化关系

U3　U6 与 ManipulateRobots 有关系；是泛化关系

图 3-1 中没有将用例之间的关系完整地给出来，因此需要根据【说明】中的功能描述判定 U1　U6 与其他用例之间的关系。根据【说明】中的描述可知，Run 和 Step 是 RunProgram 的两种具体方式，所以这 3 个用例之间是有关系的，在 UML 用例图中，这种关联通常采用泛化关系描述。同理，U3　U6 用例是用例 ManipulateRobots 的 4 种具体实现方法，因此这 5 个用例之间也是泛化关系。

C1:World

C2:Robot/Robots

C3:Instruction

C4:InstructionSet

C5:Error/Errors

本题要求将类图中缺失的 5 个类补充完整。在解答此类题目时，首先考虑类图中的特殊关系，如继承关系、聚集或组合关系等，这是比较好的突破口。另外应关注类之间的多重度。在图 3-2 中出现了两个聚集关系：C1 和 C2 之间以及 C3 和 C4 之间。我们先考虑 C1 和 C2 这一对，因为这两个类之间的多重度是一个具体的范围 1..2 。【说明】中有一句话：“用户通过操作仿真系统中的 1　2 个机器人来探索虚拟世界”，也就是说在虚拟世界中包含着 1-2 个机器人，由此可以推断 C2 对应的是机器人 Robot/Robots ，Cl 代表的就是整个虚拟世界 World 。

下面我们来看 C3 和 C4 这一对聚集关系。C4 和 Interpreter 、Parser 有关联，而这两个类与文件及机器人指令集的解析有关，由此可以推断，C3 、C4 这两个类也应该跟解析功

能相关。由【说明】可知，系统中有两类需要解析的事物：虚拟世界文件和机器人指令集，而机器人指令集是由若干条指令构成的，这里就出现了一个聚集结构。因此 C3 应该对应 Instruction， C4 对应的是 InstructionSet 。
对于最后一个类，将功能需求与用例图再回顾一遍，发现在类图中还缺少关于错误信息的描述，因此 C5 所对应的就是类 Error 。

试题四 答案： 解析： (1) size[1][j]=1
(2) size[i][j]=size[i-1][j]
(3) net[m++]=i 或其等价形式

一般不要求考生设计问题的求解算法，但要求考生能够理解题目给出的算法设计思路，并补充 C 程序。如本题中的空(1)，可以根据题干中递归式第一部分和 C 代码中的注释，得到答案 size[1][j]=1 。

空(2)则根据阅读题干中递归式第二部分和 C 代码中的注释，得到答案 size[i][j]=size[i-1][j]。

(4)动态规划算法
(5) $O(n^2)$
(6) O(n)

题干在叙述过程中，较明显的提到了动态规划策略的几个特点，如最优子结构，递归式，自底向上求解等，因此这是一个动态规划算法。算法的时间复杂度分析也较简单。函数 maxNum 中有两重循环，时间复杂度为 $O(n^2)$。函数 constructSet 中有一重循环，时间复杂度为 O(n)。
(7) 4
(8) (3， 4)(5， 5)(7， 9)(9， 10)

本问题考查该算法的一个实例，理解了题干就可以直接计算出该实例的解，即最大不相交连接数为 4，连线为：(3， 4)(5， 5)(7， 9)(9， 10)。

（1）当 i = 1 时，$size(1,j)=\begin{cases}0 & j<\pi(1)\\ 1 & \text{其他情况}\end{cases}$

（2）当 i > 1 时，$size(i,j)=\begin{cases}size(i-1,j) & j<\pi(i)\\ \max\{size(i-1,j),size(i-1,\pi(i)-1)+1\} & \text{其他情况}\end{cases}$

试题五 答案：　解析：　(1) Address*address;

(2) address->street() ;

(3) address->zip() ;

(4) address->city() ;

(5) DutchAddress *addr=new DutchAddressAdaptor(addr)

本题考查 Adapter(适配器)模式的基本概念和应用。

Adapter 模式的设计意图是，将一个类的接口转换成客户希望的另外一个接口。 Adapter 模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

Adapter 模式有两种实现方式。类适配使用多重继承对一个接口与另一个接口进行匹配，其结构如图 5-2 所示。

/

对象适配器依赖于对象组合，其结构如图 5-3 所示。

•定义 Client 使用的与特定领域相关的接口。

•Client 与符合接口的对象协同。

•Adaptee 定义一个已经存在的接口，这个接口需要适配。

•Adapter 对 Adaptee 的接口与 Target 接口进行适配。

Adapter 模式适用于：

•想使用一个已经存在的类，而它的接口不符合要求。

•想创建一个可以复用的类，该类可以与其他不相关的类或不可预见的类(即那些接口可能不一定兼容的类)协同工作。

•(仅适用于对象 Adapter)想使用一个已经存在的子类，但是不可能对每一个都进行子类化以匹配他们的接口。对象适配器可以适配它的父类接口。

本题中采用对象适配器，题中类 DutchAdddressAdapter 对应图 5-3 中的 Adapter 、DutchAddress 对应图 5-3 中的 Target 、 Address 对应图 5-3 中的 Adaptee 。

由图 5-3 可知，在 Adapter 中应该有一个 Adaptee 的对象，因此空(1) 处应该填写的是

Address 的对象： Address*address 。

类 DutchAddress 的实现采用了 C++中的抽象类，作为其子类 DutchAddressAdapter ，必须对 DutchAddress 中的 3 个纯虚拟函数进行重置，所以空(2) （ 4）是在考查这 3 个纯虚拟函数在子类中的实现方式。由图 5-3 可知， Adapter 中方法的实现方式还是要借助于 Adaptee 中所提供的行为，也就是说， DutchAddressAdapter 中 3 个纯虚拟函数的实现与 Address 是密不可分的。由此可知，空(2) （ 4）分别应填入： address->street() 、address-〉 zip() 和 address->city() 0

第(5) 空考查 Adapter 模式的使用。这里调用普通函数 testDutch 来进行测试，这个函数要求传递 DutchAddress 类型的参数，并且给出了实参的名字： addrAdatper ，，因此第(5) 空应该填写的是 addrAdapter 的创建语句，这里需要使用到 DutchAddress 的构造函数。因此第(5) 空应填写： DutchAddress*addrAdapter=newDutchAddressAdapter(addr)。

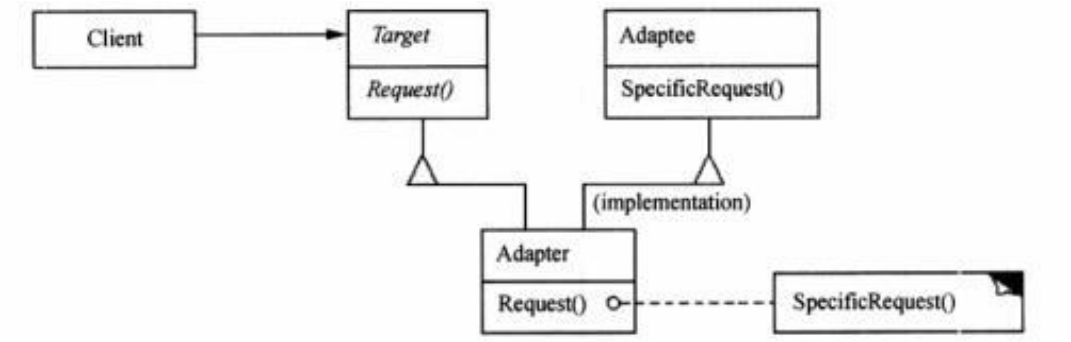


图 5-2 类适配器结构图

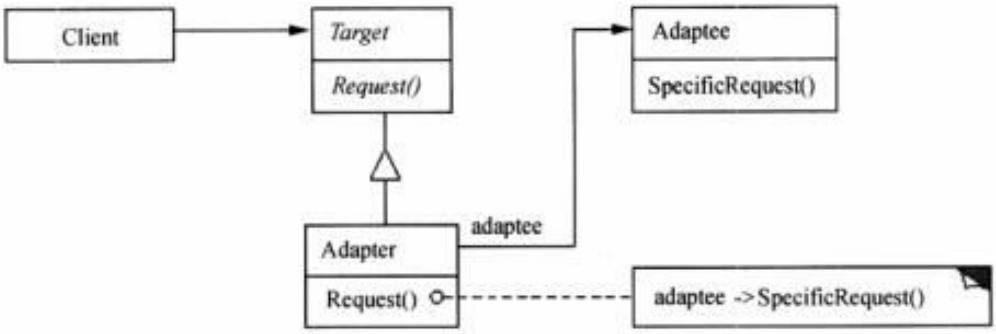


图 5-3 对象适配器结构图

试题六 答案： 解析： (1) Address address;

(2) address.street() ;

(3) address.zip() ;

(4) address.city() ;

(5) DutchAddress addrAdapter=new DutchAddressAdaptor(addr)

本题考查 Adapter(适配器)模式的基本概念和应用。

Adapter 模式的设计意图是，将一个类的接口转换成客户希望的另外一个接口。 Adapter 模式使得原本由于接口不兼容而不能一起工作的那些类可以一起工作。

Adapter 模式有两种实现方式。类适配使用多重继承对一个接口与另一个接口进行匹配，其结构如图 6-2 所示。

对象适配器依赖于对象组合，其结构如图 6-3 所示。

本题中采用对象适配器，题中类 DutchAdddressAdapter 对应图 6-3 中的 Adapter 、DutchAddress 对应图 6-3 中的 Target 、 Address 对应图 6-3 中的 Adaptee 。

由图 6-3 可知，在 Adapter 中应该有一个 Adaptee 的对象，因此空(1) 处应该填写的是 Address 的对象： Addressaddress 。

空(2) （ 4）考查父类中的 3 个方法在子类 DutchAddressAdapter 的实现方式。由图 6-3 可知， Adapter 中方法的实现方式还是要借助于 Adaptee 中所提供的行为，也就是说，DutchAddressAdapter 中 3 个方法的实现与 Address 是密不可分的。由此可知，空(2) （4）分别应填入： address.street() 、 address.zip() 和 address.city() 。

第(5) 空考查 Adapter 模式的使用。这里使用方法 testDutch 来进行测试这个方法要求传递 DutchAddress 类型的参数，并且给出了实参的名字： addrAdatper 。因此第(5) 空应该填写的是 addrAdapter 的创建语句，这里需要使用到 DutchAddress 的构造函数。因此第(5) 空应填写： DutchAddressaddrAdapter=newDutchAddressAdapter(addr)。

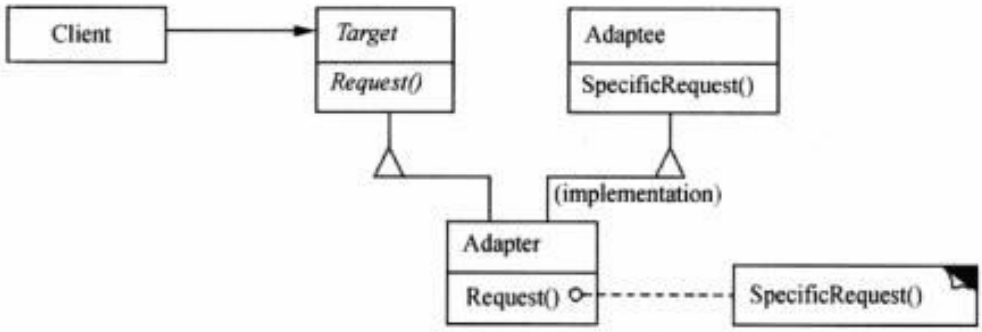


图 6-2　类适配器结构图

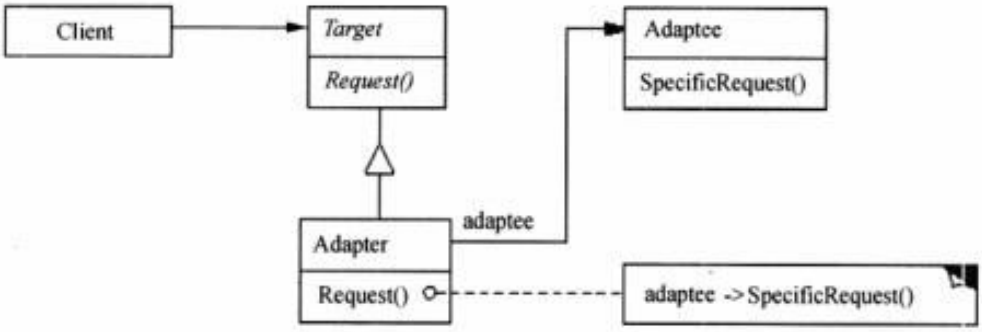


图 6-3　对象适配器结构图