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 2017 年 下半年 下午试卷 案例

（考试时间 150 分钟）

试题一 【说明】

某公司拟开发一个共享单车系统，采用北斗定位系统进行单车定位，提供针对用户的 APP 以及微信小程序、基于 Web 的管理与监控系统。该共享单车系统的主要功能如下。

(1）用户注册登录。用户在 APP 端输入手机号并获取验证码后进行注册，将用户信息进行存储。用户登录后显示用户所在位置周围的单车。

(2）使用单车。

①扫码/手动开锁。通过扫描二维码或手动输入编码获取开锁密码，系统发送开锁指令进行开锁，系统修改单车状态，新建单车行程。

②骑行单车。单车定时上传位置，更新行程。

③锁车结账。用户停止使用或手动锁车并结束行程后，系统根据已设置好的计费规则及使用时间自动结算，更新本次骑行的费用并显示给用户，用户确认支付后，记录行程的支付状态。系统还将重置单车的开锁密码和单车状态。

(3）辅助管理。

①查询。用户可以查看行程列表和行程详细信息。

②报修。用户上报所在位置或单车位置以及单车故障信息并进行记录。

(4）管理与监控。

①单车管理及计费规则设置。商家对单车基础信息、状态等进行管理，对计费规则进行设置并存储。

②单车监控。对单车、故障、行程等进行查询统计。

③用户管理。管理用户信用与状态信息，对用户进行查询统计。现采用结构化方法对共享单车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。

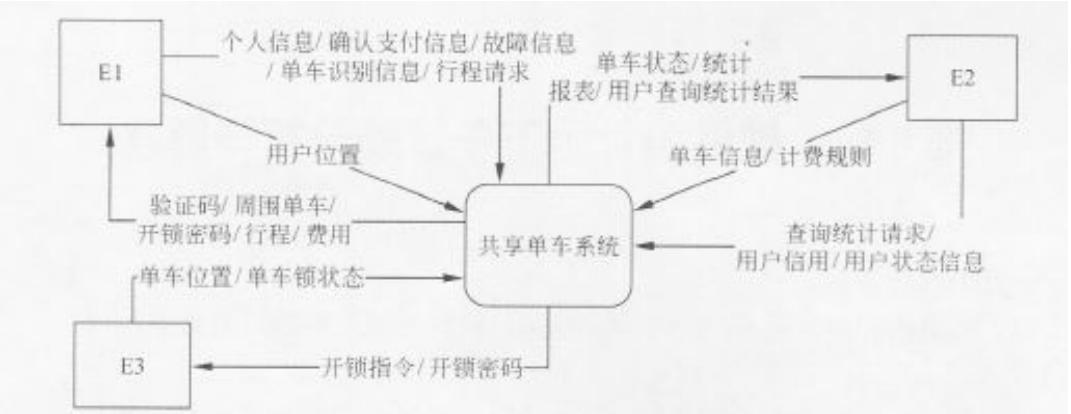


图 1-1　上下文数据流图

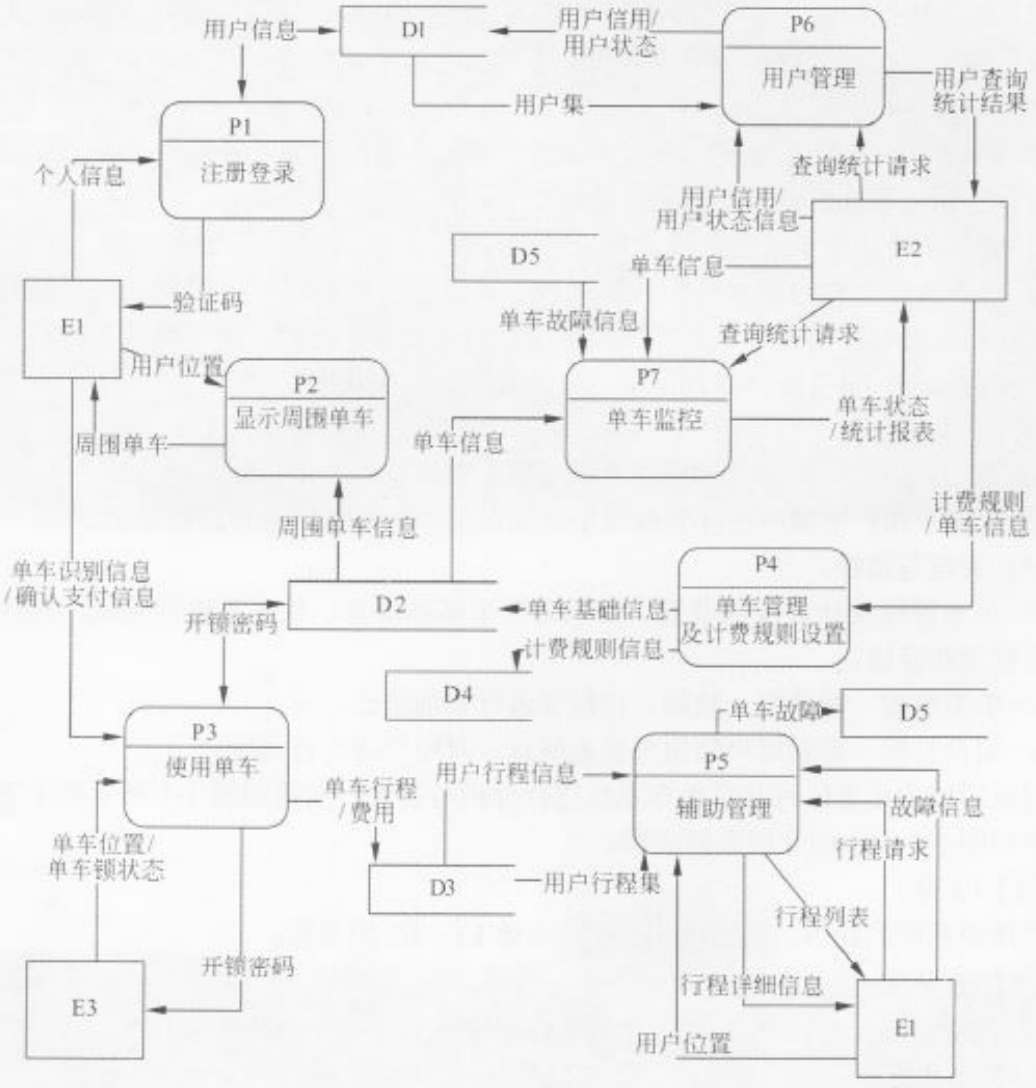


图 1-2　0 层数据流图

问题：1.1

(3 分)

使用说明中的词语，给出图 1-1 中的实体 E1~E3 的名称。

问题：1.2
(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：1.3
(5 分)
根据说明和图中术语及符号，补充图 1-2 中缺失的数据流及其起点和终点。
问题：1.4
(2 分)
根据说明中术语，说明"使用单车"可以分解为哪些子加工?

试题二 【说明】

M 公司为了便于开展和管理各项业务活动，提高公司的知名度和影响力，拟构建一个基于网络的会议策划系统。

【需求分析结果】

该系统的部分功能及初步需求分析的结果如下 ：

(1) M 公司旗下有业务部、策划部和其他部门。部门信息包括部门号、部门名、主管、联系电话和邮箱号；每个部门只有一名主管，只负责管理本部门的工作，且主管参照员工关系的员工号；一个部门有多名员工，每名员工属于且仅属于一个部门。

(2) 员工信息包括员工号、姓名、职位、联系方式和薪资。职位包括主管、业务员、策划员等。业务员负责受理用户申请，设置受理标志。一名业务员可以受理多个用户申请，但一个用户申请只能由一名业务员受理。

(3) 用户信息包括用户号、用户名、银行账号、电话、联系地址。用户号唯一标识用户信息中的每一个元组。

(4) 用户申请信息包括申请号、用户号、会议日期、天数、参会人数、地点、预算和受理标志。申请号唯一标识用户申请信息中的每一个元组，且一个用户可以提交多个申请，但一个用户申请只对应一个用户号。

(5) 策划部主管为已受理的用户申请制定会议策划任务。策划任务包括申请号、任务明细

和要求完成时间。申请号唯一标识策划任务的每一个元组。一个策划任务只对应一个已受理的用户申请，但一个策划任务可由多名策划员参与执行，且一名策划员可以参与执行，且在项策划任务。

【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。

【关系模型设计】

部门(部门号，部门名，部门主管，联系电话，邮箱号)

员工(员工号，姓名，(a)，联系方式，薪资)

用户(用户名，(b)，电话，联系地址)

用户申请(申请号，用户号，会议日期，天数，参会人数，地点，受理标志，(c))

执行(申请号，任务明细，(d))

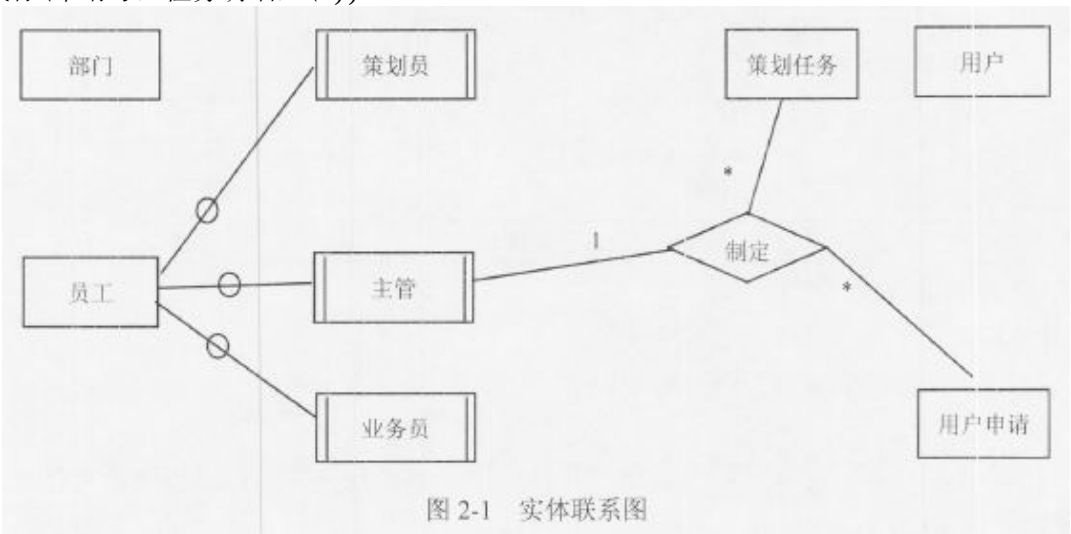


图 2-1　实体联系图

问题：2.1
(5 分)
根据问题描述，补充五个联系，完善图 2-1 的实体联系图。联系名可用联系 1 、联系 2 、联系 3 、联系 4 和联系 5 ，联系的类型为 1:1 、 1 ： n 和 m ： n(或 1:1 、 1 ：*和*：*)。

问题：2.2
(4 分)
根据题意，将关系模式中的空(a)~(d)补充完整，并填入答题纸对应的位置上。
问题：2.3
(4 分)
给出“用户申请”和“策划任务”关系模式的主键和外键。
问题：2.4
(2 分)
请问“执行”关系模式的主键为全码的说法正确吗？为什么？

试题三　【说明】

某大学拟开发一个用于管理学术出版物(Publication)的数字图书馆系统，用户可以从该系统查询或下载已发表的学术出版物。系统的主要功能如下:

1. 登录系统。系统的用户(User)仅限于该大学的学生(Student)、教师(Faculty)和其他工作人员(Staff)。在访问系统之前，用户必须使用其校园账户和密码登录系统。

2. 查询某位作者(Author)的所有出版物。系统中保存了会议文章(ConfPaper)、期刊文章(JurnalArticle)和校内技术报告(TechReport)等学术出版物的信息，如题目、作者以及出版年份等。除此之外，系统还存储了不同类型出版物的一些特有信息；

(1) 对于会议文章，系统还记录了会议名称、召开时间以及召开地点；

(2) 对于期刊文章，系统还记录了期刊名称、出版月份、期号以及主办单位；

(3) 对于校内技术报告，系统记录了由学校分配的唯一 ID 。

3. 查询指定会议集(Proceedings)或某个期刊特定期(Edition)的所有文章。会议集包含了发表在该会议(在某个特定时间段、特定地点召开)上的所有文章。期刊的每一期在特定时间发行，其中包含若干篇文章。

4. 下载出版物。系统记录每个出版物被下载的次数。

5. 查询引用了某篇出版物的所有出版物。在学术出版物中引用他人或早期的文献作为相关

工作或背景资料是很常见的现象。用户也可以在系统中为某篇出版物注册引用通知，若有新的出版物引用了该出版物，系统将发送电子邮件通知该用户。

现在采用面向对象方法对该系统进行开发，得到系统的初始设计类图如图 3-1 所示。

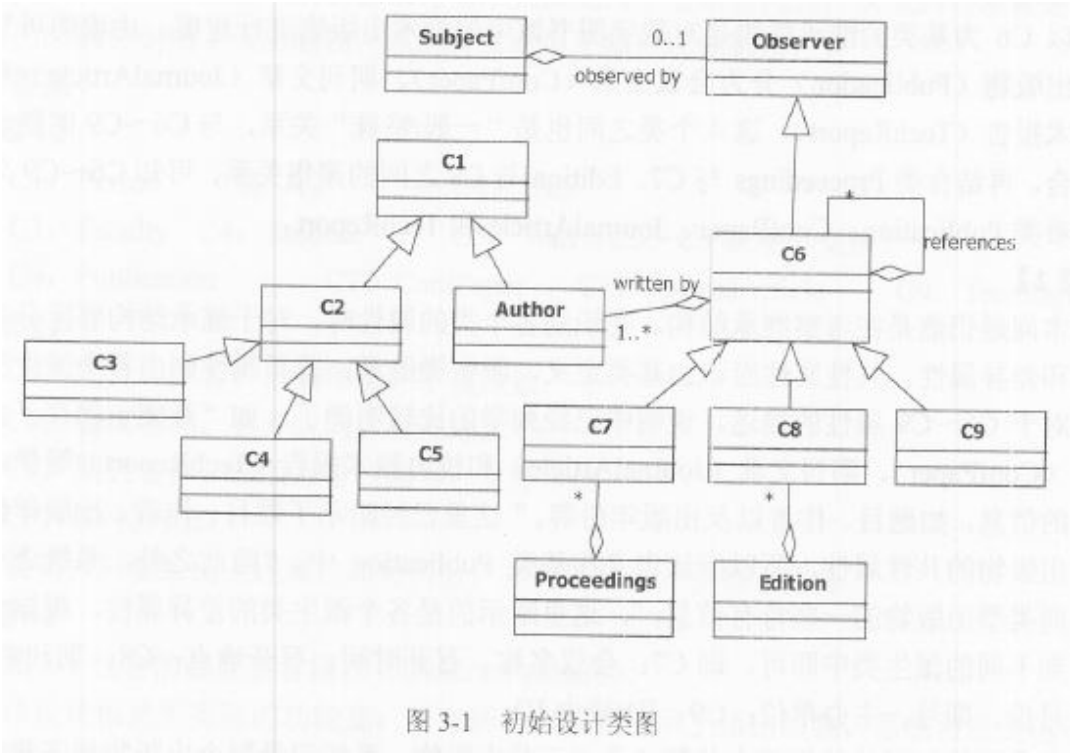


图 3-1　初始设计类图

问题：3.1
(9 分)
根据说明中的描述，给出图 3-1 中 C1~C9 所对应的类名。
问题：3.2
(4 分)
根据说明中的描述，给出图 3-1 中类 C6~C9 的属性。
问题：3.3
(2 分)
图 3-1 中包含了哪种设计模式？实现的是该系统的哪个功能？

试题四　【说明】

一个无向连通图 G 点上的哈密尔顿(Hamiltion)回路是指从图 G 上的某个顶点出发，经过图上所有其他顶点一次且仅一次，最后回到该顶点的路劲。一种求解无向图上哈密尔顿回路算法的基础私下如下：

假设图 G 存在一个从顶点 V_0 出发的哈密尔顿回路 V_0——V_1——V_2——V_3——...——V_{n-1}——V_0。算法从顶点 V_0 出发，访问该顶点的一个未被访问的邻接顶点 V_1，接着从顶点 V_1 出发，访问 V_1 一个未被访问的邻接顶点 V_2，对顶点 V_i，重复进行以下操作：访问 V_i 的一个未被访问的邻接接点 V_{i+1}；若 V_i 的所有邻接顶点均已被访问，则返回到顶点 V_{i-1}，考虑 V_{i-1} 的下一个未被访问的邻接顶点，仍记为 V_i；直到找到一条哈密尔顿回路或者找不到哈密尔顿回路，算法结束。

【C 代码】

下面是算法的 C 语言实现。

(1) 常量和变量说明

n：图 G 中的顶点数

c[][]：图 G 的邻接矩阵

K：统计变量，当期已经访问的定点数为 k+1

x[k]：第 k 个访问的顶点编号，从 0 开始

Visited[x[k]]：第 k 个顶点的访问标志，0 表示未访问，1 表示已访问

(2) C 程序

```
#include <stdio.h>
#include <stdlib.h>
#define MAX 100

 void Hamilton(int n, int x[MAX], int c[MAX][MAX]){
    int i;
    int visited[MAX];
    int k;
/* 初始化 x 数组和 visited 数组 */
    for (i = 0; i < n; i++){
       x[i] = 0;
       visited[i] = 0;
    }
    /* 访问起始顶点 */
    k = 0;
    ____(1)____;
    x[0] = 0;
    k = k + 1;
/*访问其他顶点*/
    while(k >= 0){
       x[k] = x[k] + 1;
```

```
        while(x[k] < n){
            if( ___(2)___ && c[x[k - 1]][x[k]] == 1){ /*邻接顶点x[k]未被
                                                        访问过*/
                break;
            } else {
                x[k] = x[k] + 1;
            }
        }
        if(x[k] < n && k == n - 1 && ___(3)___ ){ /*找到一条哈密尔顿回路*/
            for(k = 0; k < n; k++){
                printf("%d--", x[k]);    /* 输出哈密尔顿回路 */
            }
            printf("%d\n", x[0]);
            return;
        } else if(x[k] < n && k < n - 1){  /*设置当前顶点的访问标志，继续下
                                             一个顶点*/
            ___(4)___;
            k = k + 1;
        } else { /*没有未被访问过的邻接顶点，回退到上一个顶点*/
            x[k] = 0;
            visited[x[k]] = 0;
            ___(5)___;
        }
    }
}
```

问题： 4.1
(10 分)
根据题干说明。填充 C 代码中的空(1)~(5).
问题： 4.2
(5 分)
根据题干说明和 C 代码，算法采用的设计策略为(6)，该方法在遍历图的顶点时，采用的是(7)方法(深度优先或广度优先)。

试题五 阅读下列说明和 C 函数代码，将应填入(n)处的字句写在答题纸的对应栏内。

【说明】

某图像预览程序要求能够查看 BMP 、 JPEG 和 GIF 三种格式的文件，且能够 Windows 和 Linux 两种操作系统上运行。程序需具有较好的扩展性以支持新的文件格式和操作系统。为满足上述需求并减少所需生成的子类数目，现采用桥接(Bridge)模式进行设计，得到如

图 5-1 所示的类图。

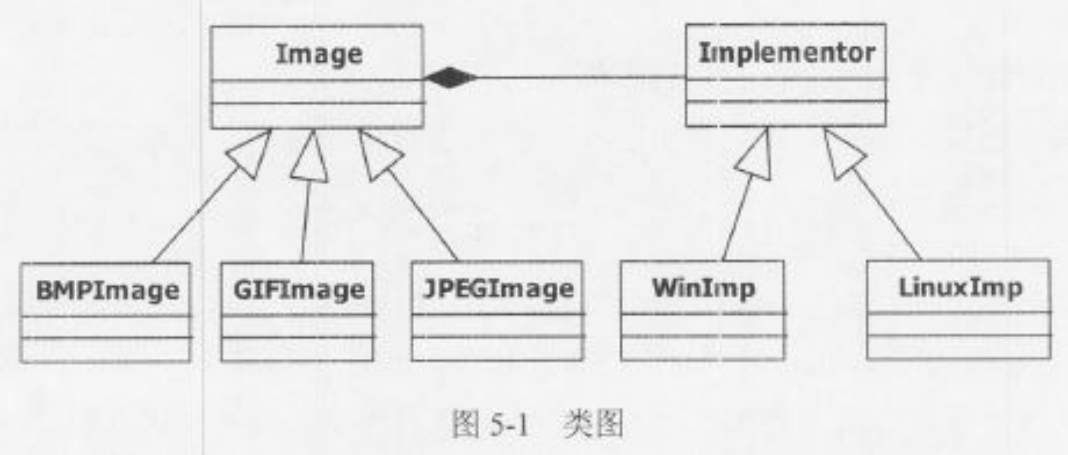


图 5-1 类图

问题： 5.1

【C++代码】

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

class Matrix{    // 各种格式的文件最终都被转化为像素矩阵
    // 此处代码省略
};
class Implementor{
public:
    ___(1)___; // 显示像素矩阵 m
};
class WinImp : public Implementor{
public:
    void doPaint(Matrix m) { /*调用 Windows 系统的绘制函数绘制像素矩阵*/ }
};
class LinuxImp : public Implementor{
public:
   void doPaint(Matrix m) { /*调用 Linux 系统的绘制函数绘制像素矩阵*/ }
};
class Image {
public:
    void setImp(Implementor *imp)  {this->imp = imp;}
    virtual void parseFile(string fileName) = 0;
protected:
    Implementor *imp;
};
class BMPImage : public Image{
   //此处代码省略
};
class GIFImage : public Image{
public:
    void parseFile(string fileName){
      // 此处解析 GIF 文件并获得一个像素矩阵对象 m
     ___(2)___;  //显示像素矩阵 m
    }
};
class JPEGImage : public Image{
// 此处代码省略
};
```

```
int main(){
    // 在Linux操作系统上查看demo.gif图像文件
    Image *image = ____(3)____;
    Implementor *imageImp = ____(4)____;
    ____(5)____;
    image->parseFile("demo.gif");
    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】

某图像预览程序要求能够查看 BMP 、 JPEG 和 GIF 三种格式的文件，且能够在 Windows 和 Linux 两种操作系统上运行。程序需具有较好的扩展性以支持新的文件格式和操作系统。为满足上述需求并减少所需生成的子类数目，现采用桥接模式进行设计，得到如图 6-1 所示的类图。

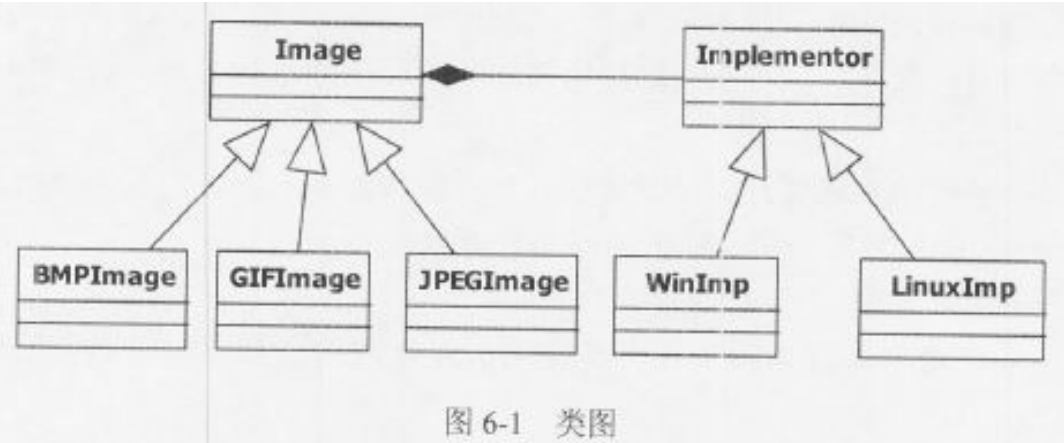


图 6-1 类图

问题： 6.1

【Java 代码】

```
import java.util.*;

class Matrix{    // 各种格式的文件最终都被转化为像素矩阵
    // 此处代码省略
};
abstract class Implementor{
    public ____(1)____;  // 显示像素矩阵 m
};
class WinImp extends Implementor{
    public void doPaint(Matrix m){   //调用 Windows 系统的绘制函数绘制像素矩阵
    }
};
class LinuxImp extends Implementor{
    public void doPaint(Matrix m){  //调用 Linux 系统的绘制函数绘制像素矩阵
    }
};
abstract class Image {
    public void setImp(Implementor imp){ this.imp = imp; }
    public abstract void parseFile(String fileName);
    protected Implementor imp;
};
class BMPImage extends Image{
    // 此处代码省略
};
class GIFImage extends Image{
    public void parseFile(String fileName){
```

```
        // 此处解析 BMP 文件并获得一个像素矩阵对象 m
        ____(2)____;  // 显示像素矩阵 m
    }
};
class JPEGImage extends Image{
    //此处代码省略
};
class Main{
    public static void main(String[] args){
        // 在 Linux 操作系统上查看 demo.gif 图像文件
        Image image = ____(3)____;
        Implementor imageImp = ____(4)____;
        ____(5)____;
        image.parseFile("demo.gif");
    }
}
```

试题一 答案：　解析：　E1:用户

E2:商家

E3:单车

本题考查采用结构化方法进行软件系统的分析与设计，主要考查数据流图(DFD) 的应用，考点与往年类似。 DFD 是结构化分析与设计方法中面向数据流建模的工具，它将系统建模成输入、加工(处理)、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，并采用自顶向下分层且逐层细化的方式，建模不同详细程度的数据流图模型。

首先需要建模上下文数据流图(顶层 DFD)来确定系统边界。在上下文 DFD 中，待开发软件系统被看作一个加工，为系统提供输入数据，以及接受系统输出数据的是外部实体，外部实体和加工之间的输入输出即为数据流。

在上下文 DFD 中确定的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，分别识别这些加工的输入数据流以及经过加工变换后的输出数据流，建模 0 层 DFD 。根据 0 层 DFD 中加工的复杂程度进一步建模加工的内容。
在建模分层 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。建模时，需要注意加工和数据流的正确使用，一个加工必须既有输入又有输出；数据流须和加工相关，即数据流至少有一头为加工。注意要在绘制下层数据流图时要保持父图与子图平衡，即父图中某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)数据流并集。
本题题干描述清晰，易于分析，要求考生细心分析题目中所描述的内容。
本问题考查的是上下文 DFD ，要求确定外部实体。在上下文 DFD 中，待开发系统名称“共享单车系统”作为唯一加工的名称，外部实体为这一加工提供输入数据流或者接收其输出数据流。通过考查系统的主要功能发现，系统中涉及用户、单车、商家。根据描述(1) 中“用户在 APP 端输入手机号并获取验证码后进行注册”，描述(2) ② 中“单车定时上传位置”、描述(4)①中“商家对单车基础信息、状态等进行管理”等信息，对照图 1-1 ，即可确定 E1 为“用户”实体， E2 为“商家”实体， E3 为“单车” 实体。
D1 ：用户
D2 ：单车
D3 ：行程或行程及费用
D4 ：计费规则
D5 ：单车故障
(注：名称后面可以带有“文件”或“表”)

本问题要求确定图 1-20 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。说明(1) 中“将用户信息进行存储”以及说明“用户登录后……”，可知加工“注册登录”需要将新注册用户信息存储在 D1,并从 D1 中读取用户信息进行登录验证，由此可知 D1 为“用户”。说明(1) 中“显示用户所在位置周围的单车”，可知加工“显示周围车辆”需要从 D2 中获取用户周围单车信息，说明(2) 中“获取开锁密码”和“系统还将重置单车的开锁密码和单车状态”可知加工“使用单车”从 D2 中获取对应单车的密码和向对应单车信息中更新重置的密码，由此可知 D2 为“单车”。说明(2) 中骑行单车时会“更新行程”、说明(3) 中“用户可以查看行程列表和行程详细信息”，可知 D3 为“行程”。说明(4)中“商家对单车基础信息、状态等进行管理，对计费规则进行设置并存储。”可知 D4 为“计费规则”；说明(3) 中“用户上报所在位置或单车位置以及单车故障信息并进行记录”，可知 D5 为“单车故障”。

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，缺少了从加工到外部实体 E1（用户)的数据流—“费用”和“开锁密码”，从加工到外部实体 E3（单车)的数据流—“开锁指令”。

再考查题干中的说明判定是否缺失内部的数据流，不难发现缺失的数据流。根据使用单车描述“系统修改单车状态”“系统根据已设置好的计费规则及使用时间自动结算，更新本次骑行的费用并显示给用户”可知加工使用单车(P3)发给 D2 单车的“单车状态”，从存储 D4（计费规则)获取“计费规则”，说明(4)②中单车监控“对单车、故障、行程等进行查询统计”，而图 1-2 中缺少了从 D3（行程)至加工单车监控(P7)的用于查询统计的行程集信息。

扫码/手动开锁、^新行程、锁车结账

或

扫码/手动开锁、更新行程、锁车、计算费用、重置开锁密码、用户确认支付

在自顶向下建模分层 DFD 时，根据功能的粒度，可以进一步进行分解。在图 1-2 所示的 0 层数据流中，“使用单车”对应于说明(2)，其中分为 3 个主要子功能—扫码/ 手动开锁、骑行单车和锁车结账，涉及输入输出数据流数量多，将其根据建模所需粒度进行分解可以分解为扫码/手动开锁、更新行程(骑行单车的主要功能)、锁车结账，或者可以进行更细粒度的分解，可以分解为扫码/手动开锁、更新行程、锁车、计算费用、重置开锁密码、用户确认支付。

数据流	起点	终点
单车状态	P3 或使用单车	D2 或单车
开锁指令	P3 或使用单车	E3 或单车
计费规则	D4 或计费规则	P3 或使用单车
费用	P3 或使用单车	E1 或用户
开锁密码	P3 或使用单车	E1 或用户
行程集信息	D3 或行程或行程及费用	P7 或单车监控

（注：数据流没有顺序要求）

试题二 答案：　解析：　完善后的实体联系图如下所示(所补充的联系和类型如虚线所示)：

本题考查数据库系统中实体联系模型(E-R 模型)和关系模式设计知识及应用。

可分析如下：

根据描述“每个部门只有一名主管，只负责管理本部门的工作”所以部门和主管之间有一

个“管理”联系，联系类型为 1:1 。
根据描述“一个部门有多名员工，每名员工只属于一个部门”，部门和员工之间有一个“所属”联系，联系类型为 1:*。
根据描述“一个用户可以提交多个申请，但一个申请对应唯一的一个用户号”，所以用户和用户申请之间有一个“提交”联系，联系类型为 1:*。
根据描述“一名业务员可以受理多个用户申请，但一个用户申请只能由一名业务员受理”，所以业务员与用户申请之间有一个“受理”联系，联系类型为 1:*。
根据描述“一个策划任务可由多名策划员参与执行，且一名策划员可以参与执行多项策划任务”，所以策划员与策划任务之间有一个“执行”联系，联系类型为*:*。
(a）部门号，职位|
(b）用户号，银行账号
(c）预算费用，业务员号/员工号
(d）要求完成时间，主管号/员工号

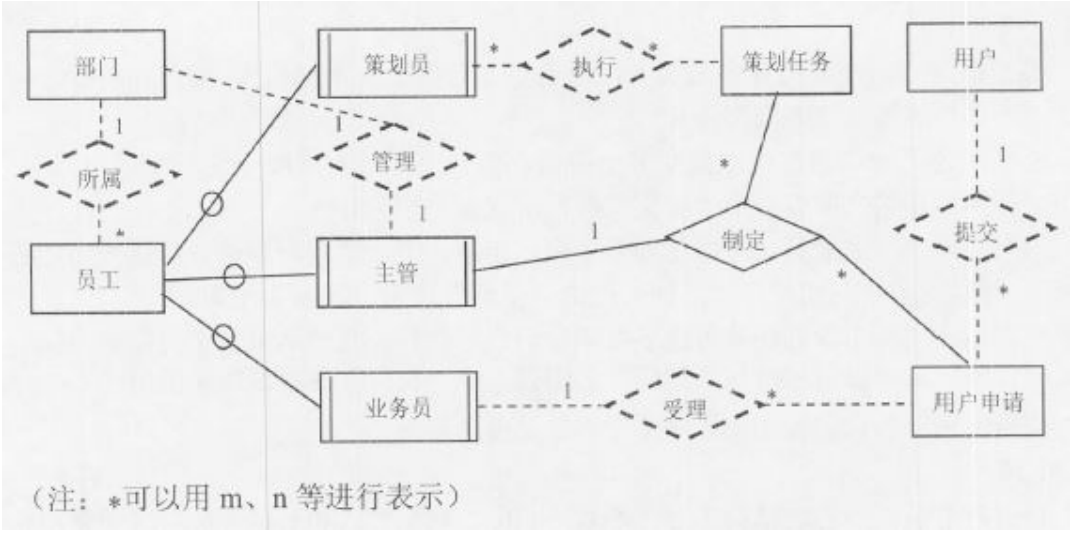
根据题意员工信息包括员工号、姓名、职位、联系方式和薪资，故员工关系模式中需要添加“职位”；又因为部门和员工之间有一个 1:*的“所属”联系需要将一端的码“部门号”并入多端，故员工关系模式中需要添加“部门号”。根据分析，空(a)应填写部门号，职位。
根据题目描述，用户信息包括用户号、用户名、账号、电话、联系地址，给定的用户关系模式中，不含用户号、账号，故空(b)应填写用户号，账号。
由于用户申请包括申请号、用户号、会议日期、天数、参会人数、地点、预算费用、受理标志和业务员，故空(c)应填写预算费用，业务员。
根据题目描述，策划任务包括申请号、任务明细、要求完成时间、主管，所以空(d)应填写要求完成时间，主管。
“用户申请”关系模式：主键为申请号
外键为用户号，业务员号/员工号
“策划任务”关系模式：主键为申请号
外键为主管号/员工号

根据描述，“申请号唯一标识申请信息中的每一个元组，且一个用户可以提交多个申请，但一个用户申请只对应一个用户号”，所以用户申请关系模式的主键为申请号。用户申请关系模式的外键为用户号、业务员，因为用户号是用户关系的主键，根据外键定义可知，用户号是用户申请关系的外建；又因为“业务员参照员工关系的员工号”，所以根据外键定义业务员是用户申请关系的外建。
策划任务关系模式的主键为申请号、外键为主管。根据题意“申请号唯一标识策划任务的

每一个元组”，所以策划任务关系模式的主键为申请号；又因为“主管参照员工关系的员工号”，所以根据外键定义主管是策划任务关系的外建。

不正确。

因为全码是指关系模式的所有属性组是这个关系模式的候选码，而“执行”关系模式的主键为申请号、策划员。



试题三 答案：　解析：　C1：Person（或人）

C2：User

C3：Faculty

C4：Student

C5：Staff（C3　C5 次序可交换）

C6：Publication

C7：ConfPaper

C8：JoumalArticle

C9：TechReport

本题属于经典的考题，主要考查面向对象分析与设计的基本概念。在建模方面，本题只涉及到了类图，并结合了设计模式的概念在其中。

在解答补充类图的问题时，应首先对所给出的类图的结构进行分析。由图 3-1 可知，该系统的类图由两个继承结构构成，一个以 C1 为基类，一个以 C6 为基类。根据图中已经给出的类可知，以 C1 为基类的继承结构应该是对数字图书馆系统的使用者进行建模。由说明可知，使用者有两类，一类是大学的用户，一类是图中已经给出的 Author 。User 与

Student 、 Faculty 和 Staff 之间是“一般/特殊”关系，可以构成一个继承结构。这个结构刚好与图 3-1 中的 C2　C5 吻合。由此可知， C2　C5 分别对应 User 、 Faculty 、 Student 和 Staff 。 User 和 Author 可以继续进行“一般/特殊”的抽象，扩展这个继承结构的层次。 因此， C1 对应的是更高层的基类—人。这个术语在说明中并没有直接给出来，但是可以很容易地根据继承关系推断出来。

以 C6 为基类的继承结构是对数字图书馆中的学术出版物进行建模。由说明可知，学术出版物(Publication)分为会议文章(ConfPaper)、期刊文章(JoumalArticle)和校内技术报告(TechReport)。这 4 个类之间也是“一般/特殊”关系，与 C6　C9 的继承结构吻合。再结合类 Proceedings 与 C7 、 Edition 与 C8 之间的聚集关系，可知 C6　C9 分别对应着类 Publication 、 ConfPaper 、 JoumalArticle 和 TechReport 。

C6 ：题目、作者、出版年份、下载次数

C7 ：会议名称、召开时间、召开地点

C8 ：期刊名称、出版月份、期号、主办单位

C9 ： ID/校内 ID

说明：只要给出上述属性即可得分，多写不扣分，少写不得分。

本问题仍然是在考察继承结构。在识别各个类的属性时，对于继承结构要区分共性属性和差异属性。共性属性应该由基类定义，派生类继承；差异属性则由各个派牛类定义。对于 C6　C9 属性的描述，说明中已经列举的比较明确了。如“系统中保存了会议文章(ConfPaper)、期刊文章(JoumalArticle)和校内技术报告(TechReport)等学术出版物的信息，如题目、作者以及出版年份等。”这里已经暗示了题目、作者、出版年份是所有出版物的共性属性，所以应该定义在基类 Publication 中。“除此之外，系统还存储了不同类型出版物的一些特有信息：”，这里暗示的是各个派生类的差异属性，根据说明对应到不同的派生类中郎可。即 C7:会议名称、召开时间、召开地点； C8:期刊名称、出版月份、期号、主办单位； C9: ID/校内 ID 。

但是不能忽视的是说明中的第 4 条“下载出版物。系统记录每个出版物被下载的次数。”这里的“下载次数”显然也是与出版物相关的属性，而且是每个出版物都应该具有的属性，因此“下载次数”也是基类 Publication 的属性之一。这样的话， C6 的属性应该包括题目、作者、出版年份、下载次数。

图 3-1 包含的是观察者模式(Observerpattern)。

该设计模式所实现的功能是：若有新的出版物引用了该出版物，系统将发送电子邮件通知该用户。

这个题目考查对设计模式概念的理解。由图 3-1 可以看到，图中定义了“Subject” 和“

Obsever”类，很容易让人联想到观察者模式。观察者模式的类图如图 3-2 所示。

观察者模式的设计意图是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。这个设计意图与说明中的第 5 个功能点非常吻合。所以在设计这个类图运用了观察者模式，实现的功能就是：若有新的出版物引用了该出版物，系统将发送电子邮件通知该用户。

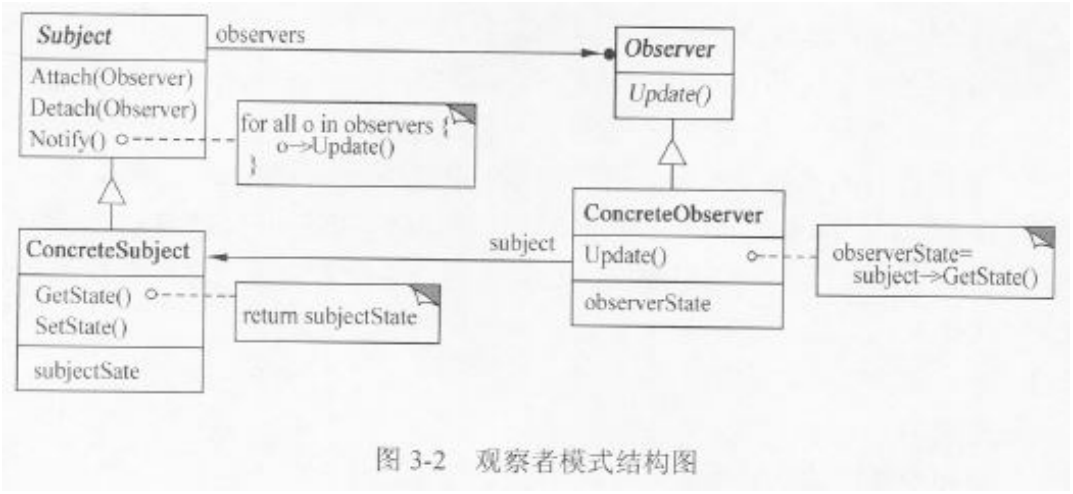


图 3-2　观察者模式结构图

试题四 答案：　解析：　(1) visited[0]=1

(2) visited[x[k]]=J=0 或等价形式

(3) c[x[k]][0]==1 或 c[x[n-1]][0]===1 或等价形式

(4) visited[x[k]]=1

(5) k=k-1 或等价形式

本题考查算法设计与分析的基础知识。

解答该类题目，首先需要理解问题和求解问题的算法思想，一般在题干中已经清晰的叙述了算法的基本思想。

求图的哈密尔顿回路是一个典型的计算问题。根据题干说明、代码注释和代码上下文，空(1）处应该填 visited[0]=1，设置起始顶点为已访问标志。空(2）处后面有注释，邻接顶点没有被访问，而第二个判断条件是是否为邻接顶点，因此第一个判断条件应该是没有被访问，因此应该填 visited[x[k]]==0 。空(3）处后面有注释，找到一条哈密尔顿回路，第一个判断条件是第 k 个访问的顶点编号是 0 到 n-1, 第二个判断条件是目前已经访问了 n 个顶点，因此第三个判断条件应该是最后访问的点和起始顶点有边连接，即 c[x[k]][0]==1 。b 空(4）处所在的程序块有注释，还没有找到哈密尔顿回路，需要继续找，那么应该先标

记当前顶点为已访问，即 visited[x[k]]=1，然后继续找。空(5) 处所在的程序块有注释，没有不被访问过的邻居顶点，往回找，把当前标记已访问的顶点标记为未访问，并返回是上一个顶点，即 k=k-1 。

(6) 回溯法

(7) 深度优先

根据题干和说明，可以比较明显的看出，在寻找哈密尔顿回路的过程中，首先是一直往后找顶点，找不到时再往回退，然后继续往前……这是典型的回溯算法的求解过程。在遍历时，其采用的是深度优先方法。

试题五 答案：　解析： (1) virtual void doPaint(Matrix m)=0

(2) imp->doPaint(m)

(3) new GIFImage()

(4) new Linuxlmp()

(5) image->setlmp(imagelmp)

本题考查设计模式的概念及其应用。

桥接(Bridge)模式是典型的结构型设计模式。结构型设计模式涉及如何组合类和对象以获得更大的结构。结构型模式采用继承机制来组合接口或实现。

桥接模式的设计意图是将抽象部分与其实现部分分离，使它们都可以独立地变化。桥接模式的结构如图 5-2 所示。

其中：

• Abstraction 定义抽象类的接口，维护一个指向 Implementor 类型对象的指针。

• RefinedAbstraction 扩充由 Abstraction 定义的接口。

• Implementor 定义实现类的接口，该接口不一定要与 Abstraction 的接口完全一致；事实上这两个接口可以完全不同。一般来说， Implementor 接口仅提供基本操作，而 Abstraction 定义了基于这些基本操作的较高层次的操作。

• Concretelmplementor 实现 Implementor 接口并定义它的具体实现。

Bridge 模式适用于：

•不希望在抽象和它的实现部分之间有一个固定的绑定关系。例如，这种情况可能是因为，在程序运行时刻实现部分应可以被选择或者切换。

•类的抽象以及它的实现都应该可以通过生成子类的方法加以扩充。这是 Bridge 模式使得开发者可以对不同的抽象接口和实现部分进行组合，并分别对它们进行扩充。

•对一个抽象的实现部分的修改应对客户不产生影响，即客户代码不必重新编译。

• (C++)想对客户完全隐藏抽象的实现部分。

•有许多类要生成的类层次结构。

•想在多个对象间共享实现(可能使用引用计数)，但同时要求客户并不知道这一点。

对比图 5-1 可知，以类 Image 为基类的继承结构对应的是桥接模式中的 Abstraction 继承结构。题目所给的 C++源代码已经将桥接模式的基本框架给出了，所需填写的空主要考察的是如何在实际问题中应用桥接模式。

第(1) 空考查的是桥接模式中实现类的接口，这个接口在基类 Implementor 中定义，由其派生类进行重置。在 C++中通常采用虚拟函数来进行实现。由 main 函数可知，在程序中并没有创建 Implementor 类的实例，使用的是指向 Implementor 的指针。所以类 Implementor 实际上是抽象类，那么在这个类中应至少定义一个纯虚拟函数。结合派生类 Winlmp 和 Linuxlmp 的代码，可知(1) 处应填写的纯虚拟函数为 virtualvoiddoPaint(Matrixm)=0 。

第(2) 空考查的是 Image 这个继承结构的实现。在这个继承结构中，需要调用 Implementor 中定义的接口，即 Implementor::doPaint 。所以(2) 处应填写 imp->doPaint (m)。

(3) （ 5）空考查的是桥接模式的使用。(3) 、(4) 处分别创建指向两个虚基类的指针，分别应填写 newGIFImage() 、 newLinuxlmp() 0 第(5) 空实现这两个继承结构之间的聚集关系，因此应填写 image->setlmp(imagelmp)。

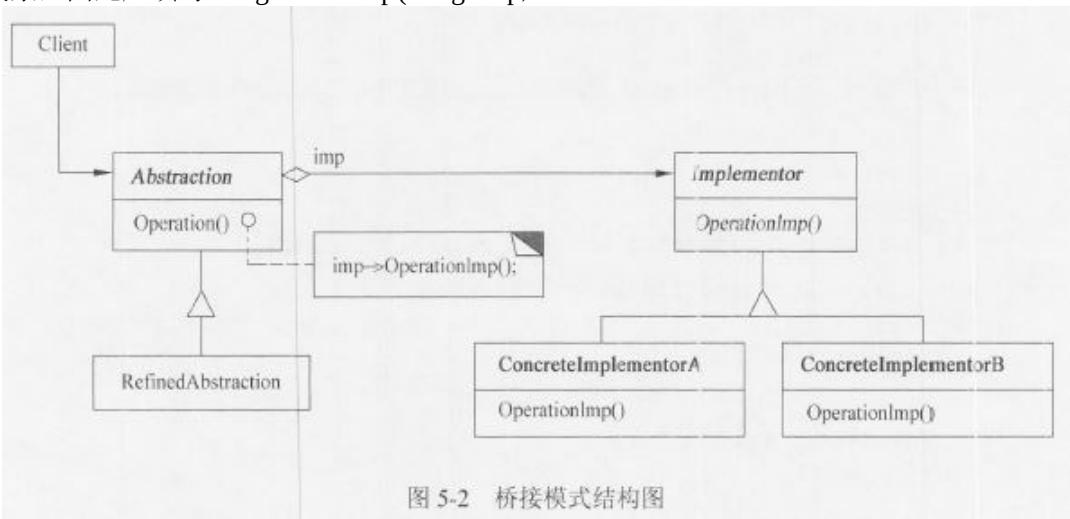


图 5-2 桥接模式结构图

试题六 答案： **解析：** (1) abstract void doPaint(Matrix m)

(2) imp.doPaint(m)

(3) new GIFImage()

(4) new Linuxlmp()

(5) image.setlmp(imagelmp)

本题考查设计模式的概念及其应用。

桥接(Bridge)模式是典型的结构型设计模式。结构型设计模式涉及如何组合类和对象以获得更大的结构。结构型模式采用继承机制来组合接口或实现。

桥接模式的设计意图是将抽象部分与其实现部分分离，使它们都可以独立地变化。桥接模式的结构如图 6-2 所示。

其中：

•Abstraction 定义抽象类的接口，维护一个指向 Implementor 类型对象的指针。

• RefinedAbstraction 扩充由 Abstraction 定义的接口。

• Implementor 定义实现类的接口，该接口不一定要与 Abstraction 的接口完全一致；事实上这两个接口可以完全不同。一般来说， Implementor 接口仅提供基本操作，而 Abstraction 走义了基于这些基本操作的较高层次的操作。

• Concretelmplementor 实现 Implementor 接口并定义它的具体实现。

Bridge 模式适用于：

•不希望在抽象和它的实现部分之间有一个固定的绑定关系。例如，这种情况可能是因为，在程序运行时刻实现部分应可以被选择或者切换。

•类的抽象以及它的实现都应该可以通过生成子类的方法加以扩充。这是 Bridge 模式使得开发者可以对不同的抽象接口和实现部分进行组合，并分别对它们进行扩充。

•对一个抽象的实现部分的修改应对客户不产生影响，即客户代码不必重新编译。

• (C++)想对客户完全隐藏抽象的实现部分。

•有许多类要生成的类层次结构。

•想在多个对象间共享实现(可能使用引用计数)，但同时要求客户并不知道这一点。

对比图 6-1 可知，以类 Image 为基类的继承结构对应的是桥接模式中的 Abstraction 继承结构。题目所给的 Java 源代码已经将桥接模式的基本框架给出了，所需填写的空主要考察的是如何在实际问题中应用桥接模式。

第(1) 空考查的是桥接模式中实现类的接口，这个接口在基类 Implementor 中定义，由其派生类进行重置。在 Java 中可以采用抽象类和抽象方法来进行实现。结合派生类 Winlmp 和 Linuxlmp 的代码，可知(1) 处应填写抽象方法 abstractvoiddoPaint(Matrixm)。

第(2) 空考查的是 Image 这个继承结构的实现。在这个继承结构中，需要调用

Implementor 中定义的接口，即 Implementor::doPaint 。所以(2) 处应填写 imp.doPaint (m)。

(3) （ 5）空考查的是桥接模式的使用。(3) 、(4) 处分别创建两个抽象类的引用，分别应填写 newGIFImage() 、 newLinuxlmp() 。第(5) 空实现这两个继承结构之间的聚集关系，因此应填写 image.setlmp(imagelmp)。

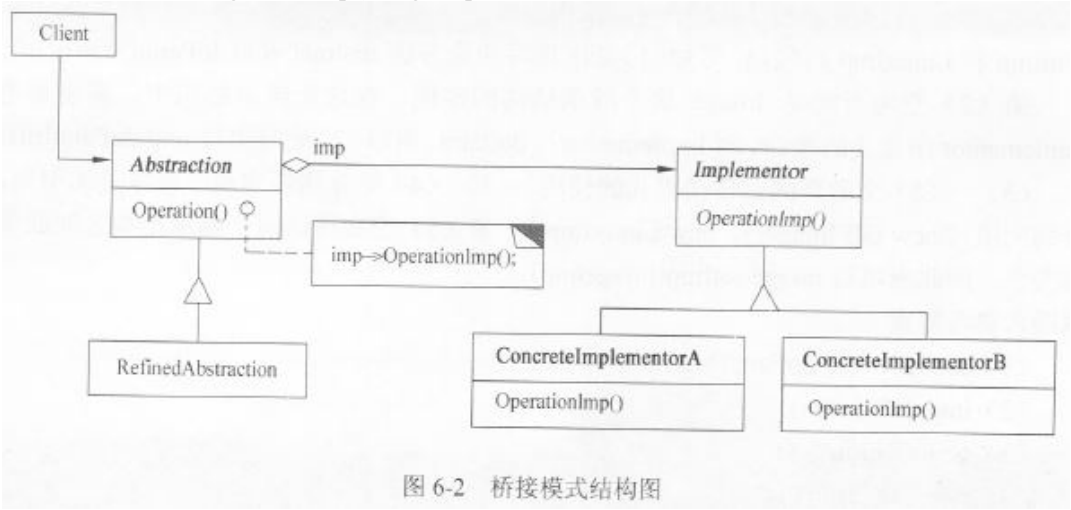


图 6-2 桥接模式结构图