

全国计算机技术与软件专业技术资格（水平）考试

2007 年上半年 软件设计师 下午试卷

（考试时间 14:00~16:30 共 150 分钟）

请按下述要求正确填写答题纸

1. 在答题纸的指定位置填写你所在的省、自治区、直辖市、计划单列市的名称。
2. 在答题纸的指定位置填写准考证号、出生年月日和姓名。
3. 答题纸上除填写上述内容外只能写解答。
4. 本试卷共 7 道题，试题一至试题四是必答题，试题五至试题七选答 1 道。每题 15 分，满分 75 分。
5. 解答时字迹务必清楚，字迹不清时，将不评分。
6. 仿照下面例题，将解答写在答题纸的对应栏内。

例题

2007 年上半年全国计算机技术与软件专业技术资格(水平)考试日期是(1)月(2)日。

因为正确的解答是“5 月 26 日”，故在答题纸的对应栏内写上“5”和“26”（参看下表）。

例题	解答栏
(1)	5
(2)	26

试题一（共 15 分）

阅读以下说明和图，回答问题1至问题3，将解答填入答题纸的对应栏内。

[说明]

某房屋租赁公司欲建立一个房屋租赁服务系统，统一管理房主和租赁者的信息，从而快速地提供租赁服务。该系统具有以下功能：

1. 登记房主信息。对于每名房主，系统需登记其姓名、住址和联系电话，并将这些信息写入房主信息文件。

2. 登记房屋信息。所有在系统中登记的房屋都有一个唯一的识别号（对于新增加的房屋，系统会自动为其分配一个识别号）。除此之外，还需登记该房屋的地址、房型（如平房、带阳台的楼房、独立式住宅等）、最多能够容纳的房客数、租金及房屋状态（待租赁、已出租）。这些信息都保存在房屋信息文件中。一名房主可以在系统中登记多个待租赁的房屋。

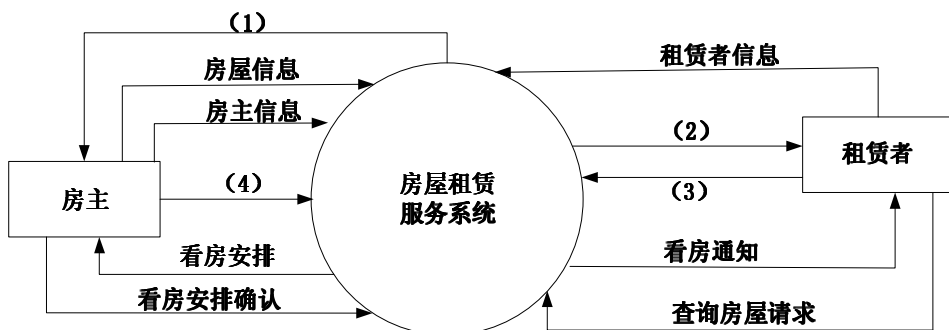
3. 登记租赁者信息。所有想通过该系统租赁房屋的租赁者，必须首先在系统中登记个人信息，包括：姓名、住址、电话号码、出生年月和性别。这些信息都保存在租赁者信息文件中。

4. 租赁房屋。已经登记在系统中的租赁者，可以得到一份系统提供的待租赁房屋列表。一旦租赁者从中找到合适的房屋，就可以提出看房请求。系统会安排租赁者与房主见面。对于每次看房，系统会生成一条看房记录并将其写入看房记录文件中。

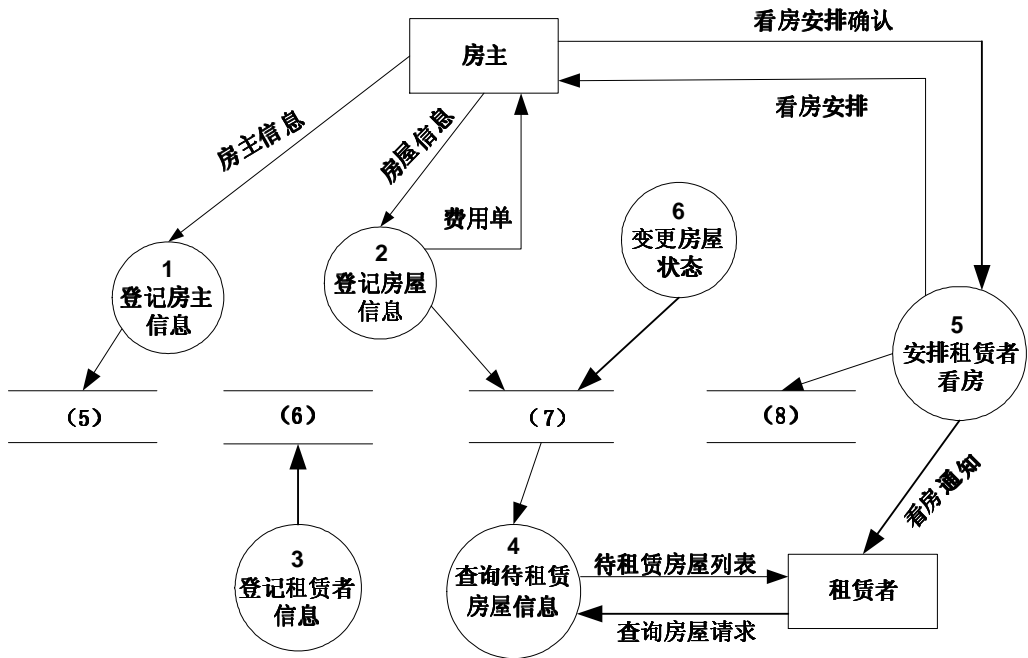
5. 收取手续费。房主登记完房屋后，系统会生成一份费用单，房主根据费用单交纳相应的费用。

6. 变更房屋状态。当租赁者与房主达成租房或退房协议后，房主向系统提交变更房屋状态的请求。系统将根据房主的请求，修改房屋信息文件。

数据流图 1-1 和 1-2 分别给出了该系统的顶层数据流图和 0 层数据流图。



数据流图 1-1



数据流图 1-2

[问题 1] (4 分)

使用[说明]中给出的词汇，将数据流图 1-1 中 (1) ~ (4) 处的数据流补充完整。

[问题 2] (4 分)

使用[说明]中给出的词汇，将数据流图 1-2 中的 (5) ~ (8) 补充完整。

[问题 3] (7 分)

数据流程图 1-2 中缺失了三条数据流，请指出这三条数据流的起点、终点和数据流名称。

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]

某医院的门诊管理系统实现了为患者提供挂号、处方药品收费的功能。具体的需求及设计如下：

1. 医院医师具有编号，姓名，科室，职称，出诊类型和出诊费用，其中出诊类型分为专家门诊和普通门诊，与医师职称无关；各个医师可以具有不同的出诊费用，与职称和出诊类型无关。

2. 患者首先在门诊挂号处挂号，选择科室和医师，根据选择的医师缴纳挂号费（医师出诊费）。收银员为患者生成挂号单，如表 2-1 所示，其中，就诊类型为医师的出诊类型。

表 2-1 XX 医院门诊挂号单

收银员：13011

时间：2007 年 2 月 1 日 08:58

就诊号	姓名	科室	医师	就诊类型	挂号费
20070205015	叶萌	内科	杨玉明	专家门诊	5 元

3. 患者在医师处就诊后，凭借挂号单和医师手写处方到门诊药房交费买药。收银员根据就诊号和医师处方中开列的药品信息，查询药品库（如表 2-2 所示）并生成门诊处方单（如表 2-3 所示）。

表 2-2 药品库

药品编码	药品名称	类型	库存	货架编号	单位	规格	单价
12007	牛蒡子	中药	51590	B1401	G	炒	0.0340
11090	百部	中药	36950	B1523	G	片	0.0313

表 2-3 XX 医院门诊处方单

时间：2007 年 2 月 1 日 10:31

就诊号	20070205015	病人姓名	叶萌	医师姓名	杨玉明
金额总计	0.65	项目总计	2	收银员	21081
药品编码	药品名称	数量	单位	单价	金额(元)
12007	牛蒡子	10	G	0.0340	0.34
11090	百部	10	G	0.0313	0.31

4. 由于药品价格会发生变化，因此，门诊管理系统必须记录处方单上药品的单价。

根据需求阶段收集的信息，设计的实体联系图和关系模式（不完整）如下所示：

1. 实体联系图

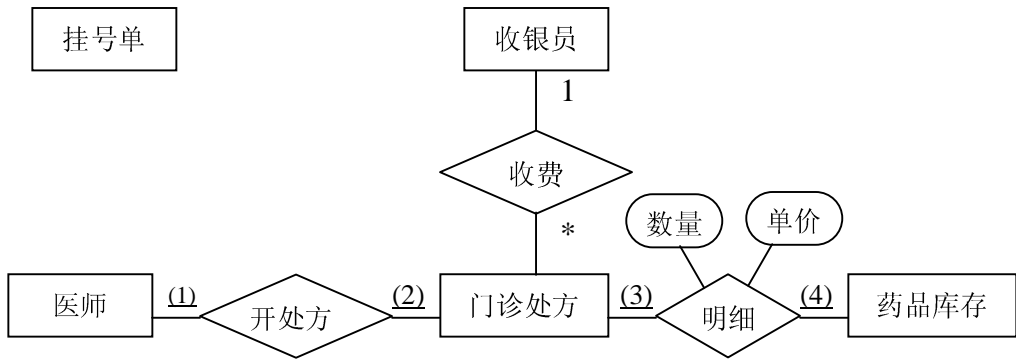


图 2-1 实体联系图

2. 关系模式

挂号单（就诊号，病患姓名，医师编号，时间，(5)）

收银员（编号，姓名，级别）

医师（编号，姓名，科室，职称，出诊类型，出诊费用）

门诊处方（(6)，收银员，时间）

处方明细（就诊号，(7)）

药品库（药品编码，药品名称，(8)）

[问题 1] (4 分)

根据问题描述，填写 2-1 实体联系图中(1)~(4)处联系的类型。

[问题 2] (4 分)

图 2-1 中还缺少几个联系？请指出每个联系两端的实体名，格式如下：

实体 1：实体 2

例如，收银员与门诊处方之间存在联系，表示为：

收银员：门诊处方 或 门诊处方：收银员

[问题 3] (7 分)

根据实体联系图 2-1，填写挂号单、门诊处方、处方明细和药品库关系模式中的空 (5)~(8) 处，并指出挂号单、门诊处方和处方明细关系模式的主键。

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]

某图书管理系统的主要功能如下：

1. 图书管理系统的资源目录中记录着所有可供读者借阅的资源，每项资源都有一个唯一的索引号。系统需登记每项资源的名称、出版时间和资源状态（可借阅或已借出）。

2. 资源可以分为两类：图书和唱片。对于图书，系统还需登记作者和页数；对于唱片，还需登记演唱者和介质类型（CD 或者磁带）。

3. 读者信息保存在图书管理系统的读者信息数据库中，记录的信息包括：读者的识别码和读者姓名。系统为每个读者创建了一个借书记录文件，用来保存读者所借资源的相关信息。

现采用面向对象方法开发该图书管理系统。识别类是面向对象分析的第一步。比较常用的识别类的方法是寻找问题描述中的名词，再根据相关规则从这些名词中删除不可能成为类的名词，最终得到构成该系统的类。表 3-1 给出了[说明]中出现的所有名词。

表 3-1

图书管理系统	资源目录	读者	资源
索引号	系统	名称	出版时间
资源状态	图书	唱片	作者
页数	演唱者	介质类型	CD
磁带	读者信息	读者信息数据库	识别码
姓名	借书记录文件	信息	

通过对表 3-1 中的名词进行分析，最终得到了图 3-1 所示的 UML 类图（类的说明如表 3-2 所示）。

表 3-2

类名	说明
LibrarySystem	图书管理系统
BorrowerDB	保存读者信息的数据库
CatalogItem	资源目录中保存的每项资源
Borrower	读者
BorrowerItems	为每个读者创建的借书记录文件

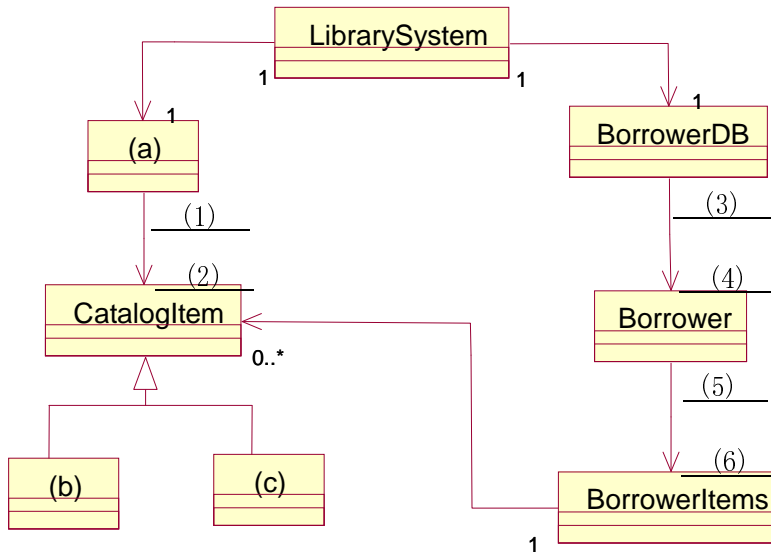


图 3-1

[问题 1] (3 分)

表 3-2 所给出的类并不完整，根据[说明]和表 3-1，将图 3-1 中的 (a) ~ (c) 处补充完整。

[问题 2] (6 分)

根据[说明]中的描述，给出图 3-1 中的类 `CatalogItem` 以及 (b)、(c) 处所对应的类的关键属性（使用表 3-1 中给出的词汇），其中，`CatalogItem` 有 4 个关键属性；(b)、(c) 处对应的类各有 2 个关键属性。

[问题 3] (6 分)

识别关联的多重度是面向对象建模过程中的一个重要步骤。根据[说明]中给出的描述，完成图 3-1 中的 (1) ~ (6)。

试题四(共 15 分)

阅读以下说明和图，填补流程图中的空缺，将解答填入答题纸的对应栏内。

[说明]

在一条农村公路的一边稀疏地分布着房子，其分布如图 4-1 所示。某电信公司需要在某些位置放置蜂窝电话基站，由于基站的覆盖范围是 6 公里，因此必须使得每栋房子到某个基站的直线距离不超过 6 公里。为简化问题，假设所有房子在同一直线上，并且基站沿该直线放置。现采用贪心策略实现用尽可能少的基站覆盖所有的房子。

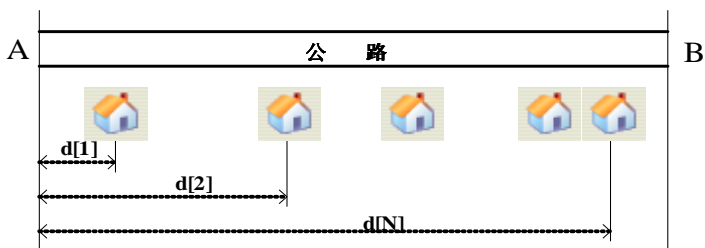


图 4-1

实现贪心算法的流程如图 4-2 所示，请填写其中空白并计算该算法的时间复杂度，其中：

1. $d[i](1 \leq i \leq N)$ 表示第 i 个房子到公路 A 端的距离， N 表示房子的总数，房子的编号按照房子到公路 A 端的距离从小到大进行编号。
2. $s[k]$ 表示第 k ($k \geq 1$) 个基站到公路 A 端的距离，算法结束后 k 的值为基站的总数。

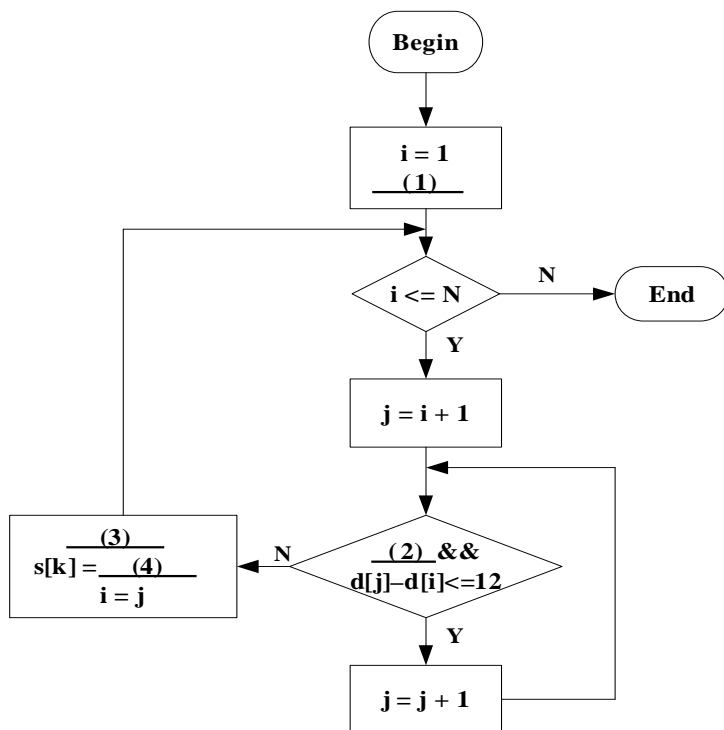


图 4-2

该算法的时间复杂度为 (5)。

从下列的 3 道试题（试题五至试题七）中任选 1 道解答。
如果解答的试题数超过 1 道，则题号小的 1 道解答有效。

试题五（共 15 分）

阅读以下说明和 C 语言函数，将应填入 (n) 处的字句写在答题纸的对应栏内。

[说明]

在一个分布网络中，资源（石油、天然气、电力等）可从生产地送往其他地方。在运输过程中，资源会有损耗。例如，天然气的气压会减少，电压会降低。我们将需要输送的资源信息称为信号。在信号从信源地送往消耗地的过程中，仅能容忍一定范围的信号衰减，称为容忍值。分布网络可表示为一个树型结构，如图 5-1 所示。信号源是树根，树中的每个节点（除了根）表示一个可以放置放大器的子节点，其中某些节点同时也是信号消耗点，信号从一个节点流向其子节点。

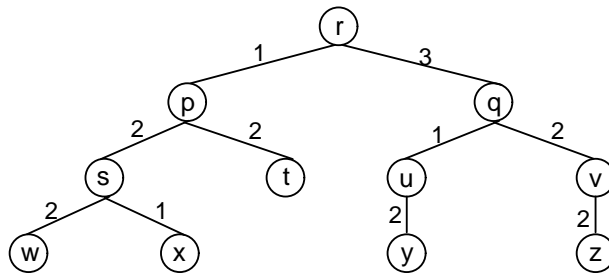


图 5-1

每个节点有一个 d 值，表示从其父节点到该节点的信号衰减量。例如，在图 5-1 中，节点 w 、 p 、 q 的 d 值分别为 2、1、3，树根节点表示信号源，其 d 值为 0。

每个节点有一个 M 值，表示从该节点出发到其所有叶子的信号衰减量的最大值。显然，叶子节点的 M 值为 0。对于非叶子节点 j ， $M(j)=\max\{M(k)+d(k) \mid k \text{ 是 } j \text{ 的孩子节点}\}$ 。在此公式中，要计算节点的 M 值，必须先算出其所有子节点的 M 值。

在计算 M 值的过程中，对于某个节点 i ，其有一个子节点 k 满足 $d(k)+M(k)$ 大于容忍值，则应在 k 处放置放大器，否则，从节点 i 到某叶子节点的信号衰减量会超过容忍值，使得到达该叶子节点时信号不可用，而在节点 i 处放置放大器并不能解决到达叶子节点的信号衰减问题。

例如，在图 5-1 中，从节点 p 到其所有叶子节点的最大衰减量为 4。若容忍值为 3，则必须在 s 处放置信号放大器，这样可使得节点 p 的 M 值为 2。同样，需要在节点 q 、 v 处放置信号放大器，如图 5-2 阴影节点所示。若在某节点放置了信号放大器，则从该节点输出的信号与信号源输出的信号等价。

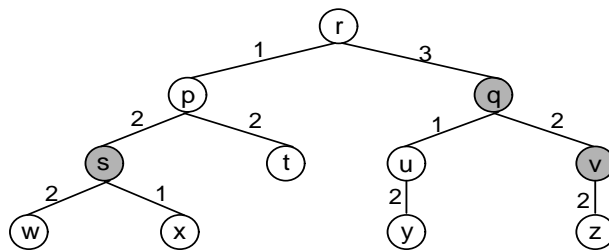


图 5-2

函数 placeBoosters(TreeNode *root)的功能是：对于给定树型分布网络中各个节点，计算其信号衰减量的最大值，并确定应在树中的哪些节点放置信号放大器。

全局变量 Tolerance 保存信号衰减容忍值。

树的节点类型定义如下：

```
typedef struct TreeNode {
    int id;                /*当前节点的识别号*/
    int ChildNum;         /*当前节点的子节点数目*/
    int d;                /*父节点到当前节点的信号衰减值*/
    struct TreeNode **childptr; /*向量，存放当前节点到其所有子节点的指针*/
    int M;                /*当前节点到其所有子节点的信号衰减值中的最大值*/
    bool boost;          /*是否在当前节点放置信号放大器的标志*/
}TreeNode;
```

[C 语言函数]

```
void placeBoosters(TreeNode *root )
{ /* 计算 root 所指节点处的衰减值，如果衰减值超出了容忍值，则放置放大器 */
    TreeNode *p;
    int i, degradation;
    if ( (1) ) {
        degradation = 0; root->M = 0;
        i = 0;
        if (i >= root->ChildNum)
            return;
        p = (2);
        for(;i < root->ChildNum && p; i++, p = (3)) {
            p->M = 0;
            (4);
            if (p->d + p->M > Tolerance) { /*在 p 所指节点中放置信号放大器*/
                p->boost = true;
                p->M = 0;
            }
            if (p->d + p->M > degradation)
                degradation = p->d + p->M;
        }
        root -> M = (5);
    }
}
```

试题六 (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]

某游戏公司现欲开发一款面向儿童的模拟游戏，该游戏主要模拟现实世界中各种鸭子的发声特征、飞行特征和外观特征。游戏需要模拟的鸭子种类及其特征如表 6-1 所示：

表 6-1

鸭子种类	发声特征	飞行特征	外观特征
灰鸭 (MallardDuck)	发出“嘎嘎”声 (Quack)	用翅膀飞行 (FlyWithWings)	灰色羽毛
红头鸭 (RedHeadDuck)	发出“嘎嘎”声 (Quack)	用翅膀飞行 (FlyWithWings)	灰色羽毛、头部红色
棉花鸭 (CottonDuck)	不发声 (QuackNoWay)	不能飞行 (FlyNoWay)	白色
橡皮鸭 (RubberDuck)	发出橡皮与空气摩擦的声 (Squeak)	不能飞行 (FlyNoWay)	黑白橡皮颜色

为支持将来能够模拟更多种类鸭子的特征，采用策略设计模式 (Strategy) 设计的类图如图 6-1 所示：

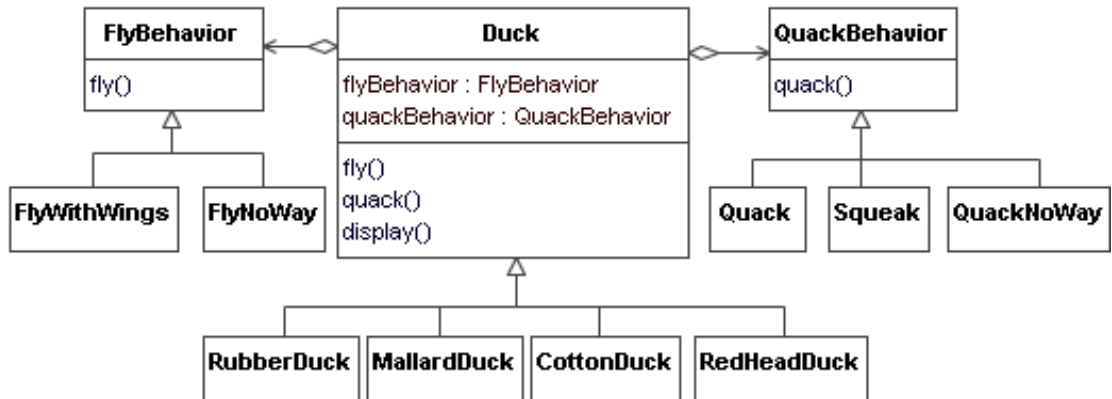


图 6-1

其中，Duck 为抽象类，描述了抽象的鸭子，而类 RubberDuck、MallardDuck、CottonDuck 和 RedHeadDuck 分别描述具体的鸭子种类，方法 fly()、quack() 和 display() 分别表示不同种类的鸭子都具有飞行特征、发声特征和外观特征；类 FlyBehavior 与 QuackBehavior 为抽象类，分别用于表示抽象的飞行行为与发声行为；类 FlyNoWay 与 FlyWithWings 分别描述不能飞行的行为和用翅膀飞行的行为；类 Quack、Squeak 与 QuackNoWay 分别描述发出“嘎嘎”声的行为、发出橡皮与空气摩擦声的行为与不发声的行为。请填补以下代码中的空缺。

[C++代码]

```

#include<iostream>
using namespace __(1);
class FlyBehavior {

```

```

    public : (2) fly () = 0;
};
class QuackBehavior {
    public: (3) quack () = 0;
};
class FlyWithWings:public FlyBehavior{
    public: void fly() { cout << "使用翅膀飞行 !" << endl; }
};
class FlyNoWay:public FlyBehavior{
    public: void fly() { cout << "不能飞行 !" << endl; }
};
class Quack:public QuackBehavior{
    public: void quack() { cout << "发出\`嘎嘎\`声 !" << endl; }
};
class Squeak:public QuackBehavior{
    public: void quack() { cout << "发出空气与橡皮摩擦声 !" << endl; }
};
class QuackNoWay:public QuackBehavior{
    public: void quack () { cout << "不能发声 !" << endl; }
};
class Duck {
protected:
    FlyBehavior * (4) ;
    QuackBehavior * (5) ;
public:
    void fly() { (6) ; }
    void quack () { (7) ; };
    virtual void display ()=0;
};
class RubberDuck: public Duck {
public:
    RubberDuck () {
        flyBehavior = new (8) ;
        quackBehavior = new (9) ;
    }
    ~RubberDuck () {
        if(!flyBehavior) delete flyBehavior;
        if(!quackBehavior) delete quackBehavior;
    }
    void display () { /*此处省略显示橡皮鸭的代码 */ }
};
//其它代码省略

```

试题七（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]

某游戏公司现欲开发一款面向儿童的模拟游戏，该游戏主要模拟现实世界中各种鸭子的发声特征、飞行特征和外观特征。游戏需要模拟的鸭子种类及其特征如表 7-1 所示：

表 7-1

鸭子种类	发声特征	飞行特征	外观特征
灰鸭 (MallardDuck)	发出“嘎嘎”声 (Quack)	用翅膀飞行 (FlyWithWings)	灰色羽毛
红头鸭 (RedHeadDuck)	发出“嘎嘎”声 (Quack)	用翅膀飞行 (FlyWithWings)	灰色羽毛、头部红色
棉花鸭 (CottonDuck)	不发声 (QuackNoWay)	不能飞行 (FlyNoWay)	白色
橡皮鸭 (RubberDuck)	发出橡皮与空气摩擦的声 (Squeak)	不能飞行 (FlyNoWay)	黑白橡皮颜色

为支持将来能够模拟更多种类鸭子的特征，采用策略设计模式(Strategy)设计的类图如图 7-1 所示：

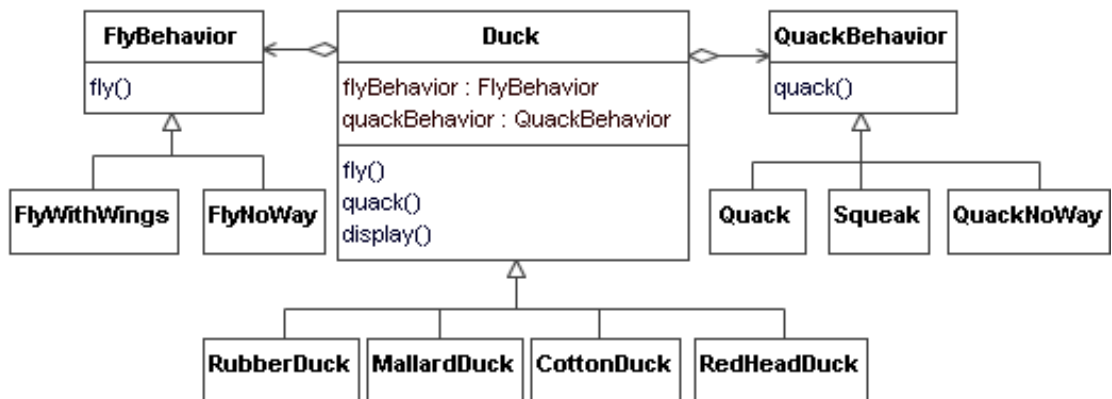


图 7-1

其中，Duck 为抽象类，描述了抽象的鸭子，而类 RubberDuck、MallardDuck、CottonDuck 和 RedHeadDuck 分别描述具体的鸭子种类，方法 fly()、quack() 和 display() 分别表示不同种类的鸭子都具有飞行特征、发声特征和外观特征；接口 FlyBehavior 与 QuackBehavior 分别用于表示抽象的飞行行为与发声行为；类 FlyNoWay 与 FlyWithWings 分别描述不能飞行的行为和用翅膀飞行的行为；类 Quack、Squeak 与 QuackNoWay 分别描述发出“嘎嘎”声的行为、发出橡皮与空气摩擦声的行为与不发声的行为。请填补以下代码中的空缺。

[Java 代码]

```

(1) FlyBehavior {
    public void fly();

```

```

};
(2) QuackBehavior {
    public void quack();
};
class FlyWithWings implements FlyBehavior{
    public void fly() { System.out.println("使用翅膀飞行！"); }
};
class FlyNoWay implements FlyBehavior{
    public void fly() { System.out.println("不能飞行！"); }
};
class Quack implements QuackBehavior{
    public void quack() { System.out.println("发出'嘎嘎'声！"); }
};
class Squeak implements QuackBehavior{
    public void quack() { System.out.println("发出空气与橡皮摩擦声！"); }
};
class QuackNoWay implements QuackBehavior{
    public void quack () { System.out.println("不能发声！"); }
};
abstract class Duck {
    protected FlyBehavior (3);
    protected QuackBehavior (4);
    public void fly() { (5); }
    public void quack() { (6); };
    public (7) void display();
};
class RubberDuck extends Duck {
    public RubberDuck() {
        flyBehavior = new (8);
        quackBehavior = new (9);
    }
    public void display() { /* 此处省略显示橡皮鸭的代码 */ }
};
//其它代码省略

```