全国计算机技术与软件专业技术资格（水平）考试

2008 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

2008 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 24 日”，故在答题纸的对应栏内写上“5”和“24”（参看下表）。

例题	解答栏
（1）	5
（2）	24

试题一至试题四是必答题

试题一（共 15 分）

阅读以下说明和图，回答问题1至问题4，将解答填入答题纸的对应栏内。

【说明】

某音像制品出租商店欲开发一个音像管理信息系统，管理音像制品的租借业务。需求如下：

1. 系统中的客户信息文件保存了该商店的所有客户的用户名、密码等信息。对于首次来租借的客户，系统会为其生成用户名和初始密码。

2. 系统中音像制品信息文件记录了商店中所有音像制品的详细信息及其库存数量。

3. 根据客户所租借的音像制品的品种，会按天收取相应的费用。音像制品的最长租借周期为一周，每位客户每次最多只能租借 6 件音像制品。

4. 客户租借某种音像制品的具体流程为：

（1）根据客户提供的用户名和密码，验证客户身份。

（2）若该客户是合法客户，查询音像制品信息文件，查看商店中是否还有这种音像制品。

（3）若还有该音像制品，且客户所要租借的音像制品数小于等于 6 个，就可以将该音像制品租借给客户。这时，系统给出相应的租借确认信息，生成一条新的租借记录并将其保存在租借记录文件中。

（4）系统计算租借费用，将费用信息保存在租借记录文件中并告知客户。

（5）客户付清租借费用之后，系统接收客户付款信息，将音像制品租借给该客户。

5. 当库存中某音像制品数量不能满足客户的租借请求数量时，系统可以接受客户网上预约租借某种音像制品。系统接收到预约请求后，检查库存信息，验证用户身份，创建相应的预约记录，生成预约流水号给该客户，并将信息保存在预约记录文件中。

6. 客户归还到期的音像制品，系统修改租借记录文件，并查询预约记录文件和客户信息文件，判定是否有客户预约了这些音像制品。若有，则生成预约提示信息，通知系统履行预约服务，系统查询客户信息文件和预约记录文件，通知相关客户前来租借音像制品。

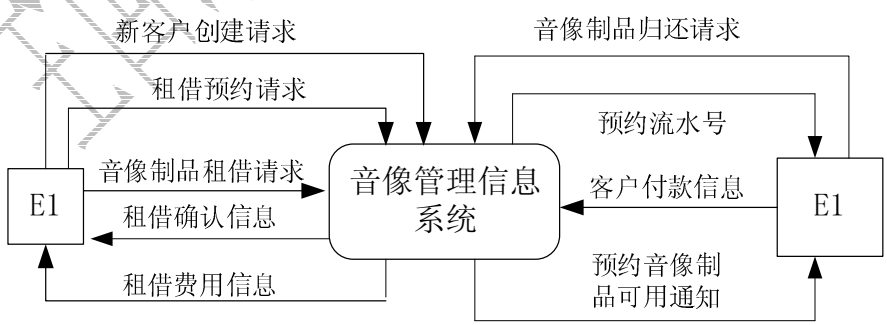


图 1-1 顶层数据流图

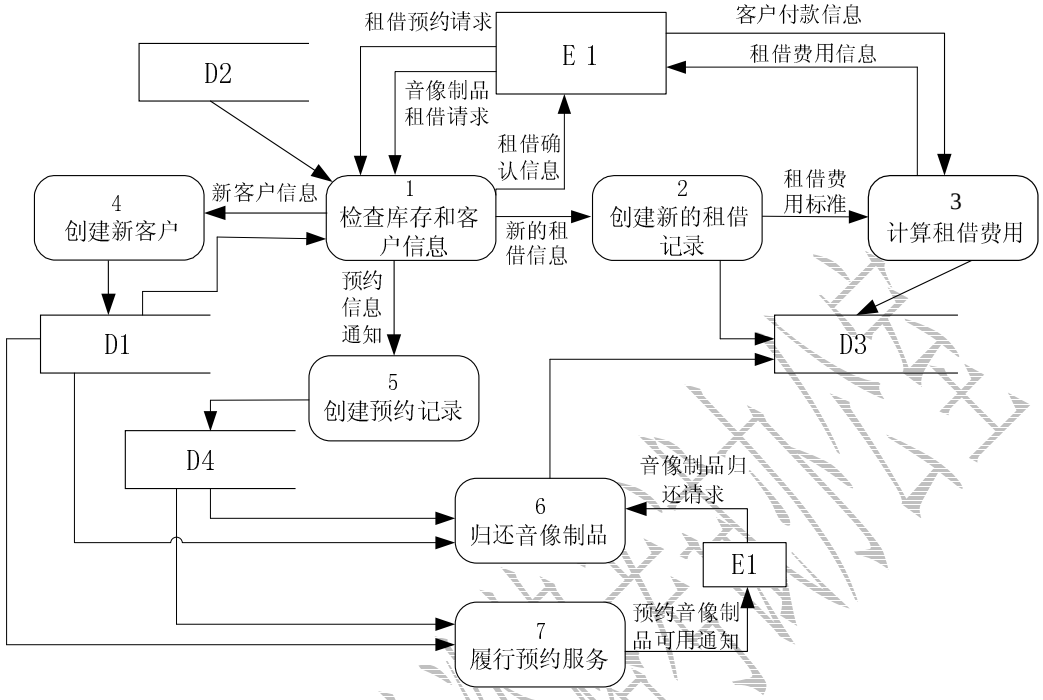


图 1-2　0 层数据流图

【问题 1】(1 分)

图 1-1 中只有一个外部实体 E1。使用**【说明】**中的词语，给出 E1 的名称。

【问题 2】(6 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】(6 分)

数据流图 1-2 缺少了三条数据流，根据说明及数据流图 1-1 提供的信息，分别指出这三条数据流的起点和终点。

起 点	终 点

【问题 4】(2 分)

在进行系统分析与设计时，面向数据结构的设计方法（如 Jackson 方法）也被广泛应用。简要说明面向数据结构设计方法的基本思想及其适用场合。

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】

某地区举行篮球比赛，需要开发一个比赛信息管理系统来记录比赛的相关信息。

【需求分析结果】

1. 登记参赛球队的信息。记录球队的名称、代表地区、成立时间等信息。系统记录球队每个队员的姓名、年龄、身高、体重等信息。每个球队有一个教练负责管理球队，一个教练仅负责一个球队。系统记录教练的姓名、年龄等信息。

2. 安排球队的训练信息。比赛组织者为球队提供了若干个场地，供球队进行适应性训练。系统记录现有的场地信息，包括：场地名称、场地规模、位置等信息。系统可为每个球队安排不同的训练场地，如表 2-1 所示。系统记录训练场地安排的信息。

表 2-1 训练安排表

球队名称	场地名称	训练时间
解放军	一号球场	2008-06-09 14:00－18:00
解放军	一号球场	2008-06-12 09:00－12:00
解放军	二号球场	2008-06-11 14:00－18:00
山西	一号球场	2008-06-10 09:00－12:00

3. 安排比赛。该赛事聘请专职裁判，每场比赛只安排一个裁判。系统记录裁判的姓名、年龄、级别等信息。系统按照一定的规则，首先分组，然后根据球队、场地和裁判情况，安排比赛（每场比赛的对阵双方分别称为甲队和乙队）。记录参赛球队名称、比赛时间、比分、比赛场地等信息，如表 2-2 所示。

4. 所有球员、教练和裁判可能出现重名情况。

表 2-2 比赛安排表

A 组：

甲队----乙队	场地名称	比赛时间	裁判	比分
解放军----北京	一号球场	2008-06-17 15:00	李大明	
天津----山西	一号球场	2008-06-17 19:00	胡学梅	

B 组：

甲队----乙队	场地名称	比赛时间	裁判	比分
上海----安徽	二号球场	2008-06-17 15:00	丁鸿平	
山东----辽宁	二号球场	2008-06-17 19:00	郭爱琪	

【概念模型设计】

根据需求阶段收集的信息，设计的实体联系图和关系模式(不完整)如下：

1．实体联系图

图 2-1 实体联系图

2．关系模式

教练(教练编号，姓名，年龄)

队员(队员编号，姓名，年龄，身高，体重，_____（a）_____)

球队(球队名称，代表地区，成立时间，_____（b）_____)

场地(场地名称，场地规模，位置)

训练记录(_____（c）_____)

裁判(裁判编号，姓名，年龄，级别)

比赛记录(_____（d）_____)

【问题 1】（4 分）

根据问题描述，补充联系及其类型，完善实体联系图 2-1。（联系及其类型的书写格式参照教练与球队之间的联系描述，联系名称也可使用***联系 1***、***联系 2***、…）

【问题 2】（8 分）

根据实体联系图 2-1，填充关系模式中的（a）、（b）、（c）和（d），并给出训练记录和比赛记录关系模式的主键和外键。

【问题 3】（3 分）

如果考虑记录一些特别资深的热心球迷的情况，每个热心球迷可能支持多个球队。热心球迷包括：姓名、住址和喜欢的俱乐部等基本信息。根据这一要求修改图 2-1 的实体联系图，给出修改后的关系模式。（仅给出增加的关系模式描述）

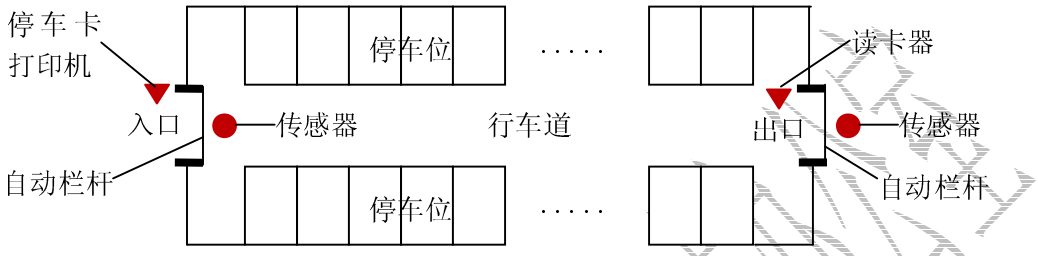
试题三（共 15 分）

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】

某汽车停车场欲建立一个信息系统，已经调查到的需求如下：

1. 在停车场的入口和出口分别安装一个自动栏杆、一台停车卡打印机、一台读卡器和一个车辆通过传感器，示意图如下：



2. 当汽车到达入口时，驾驶员按下停车卡打印机的按钮获取停车卡。当驾驶员拿走停车卡后，系统命令栏杆自动抬起；汽车通过入口后，入口处的传感器通知系统发出命令，栏杆自动放下。

3. 在停车场内分布着若干个付款机器。驾驶员将在入口处获取的停车卡插入付款机器，并缴纳停车费。付清停车费之后，将获得一张出场卡，用于离开停车场。

4. 当汽车到达出口时，驾驶员将出场卡插入出口处的读卡器。如果这张卡是有效的，系统命令栏杆自动抬起；汽车通过出口后，出口传感器通知系统发出命令，栏杆自动放下。若这张卡是无效的，系统不发出栏杆抬起命令而发出告警信号。

5. 系统自动记录停车场内空闲的停车位的数量。若停车场当前没有车位，系统将在入口处显示“车位已满”信息。这时，停车卡打印机将不再出卡，只允许场内汽车出场。

根据上述描述，采用面向对象方法对其进行分析与设计，得到了表 3-1 所示的类/用例/状态列表、图 3-1 所示的用例图、图 3-2 所示的初始类图以及图 3-3 所示的描述入口自动栏杆行为的 UML 状态图。

表 3-1 类/用例/状态列表

用例名	说明	类名	说明	状态名	说明
Car entry	汽车进入停车场	CentralComputer	停车场信息系统	Idle	空闲状态，汽车可以进入停车场
Car exit	汽车离开停车场	PaymentMachine	付款机器	Disable	没有车位
Report Statistics	记录停车场的相关信息	CarPark	停车场，保存车位信息	Await Entry	等待汽车进入
		Barrier	自动护栏	Await Ticket Take	等待打印停车卡
Car entry when full	没有车位时，汽车请求进入停车场	EntryBarrier	入口的护栏	Await Enable	等待停车场内有空闲车位
		ExitBarrier	出口的护栏		

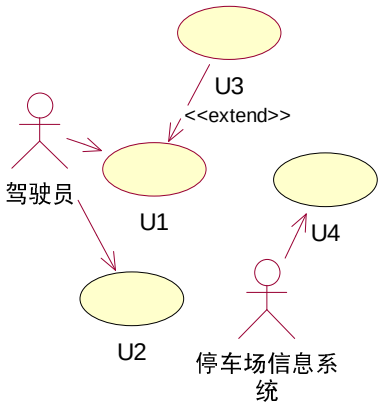


图 3-1　用例图

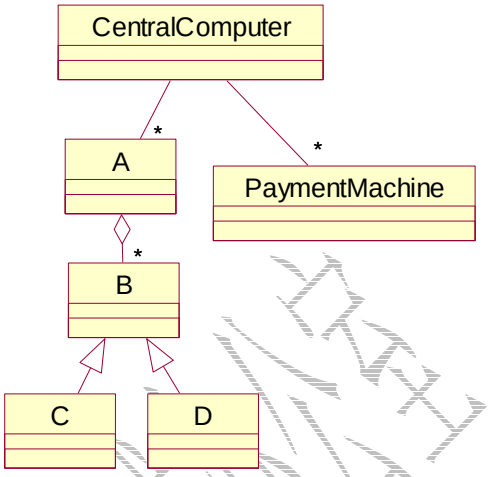


图 3-2　初始类图

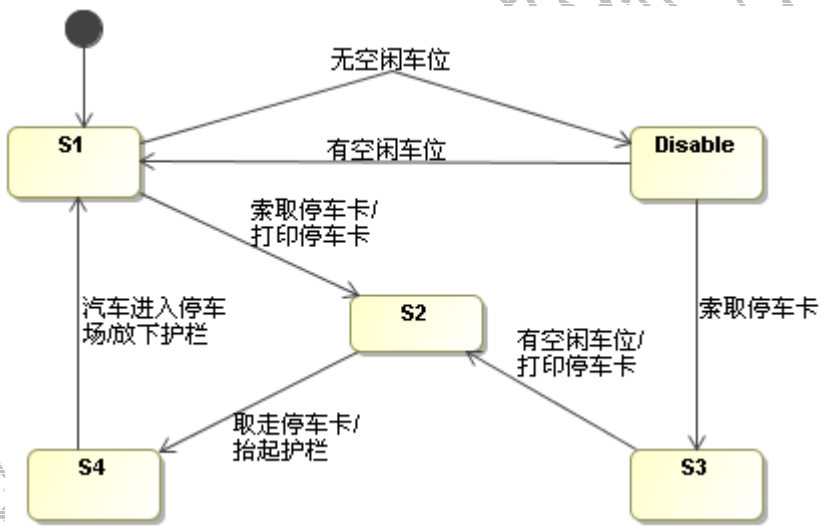


图 3-3　入口护栏的状态图

【问题 1】（3 分）

根据说明中的描述，使用表 3-1 给出的用例名称，给出图 3-1 中 U1、U2 和 U3 所对应的用例。

【问题 2】（5 分）

根据说明中的描述，使用表 3-1 给出的类的名称，给出图 3-2 中的 A~D 所对应的类。

【问题 3】（4 分）

根据说明中的描述，使用表 3-1 给出的状态名称，给出图 3-3 中 S1~S4 所对应的状态。

【问题 4】（3 分）

简要解释图 3-1 中用例 U1 和 U3 之间的 extend 关系的内涵。

试题四（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】

快速排序是一种典型的分治算法。采用快速排序对数组 A[p..r]排序的三个步骤如下:

分解：选择一个枢轴(pivot)元素划分数组。将数组 A[p..r]划分为两个子数组（可能为空）A[p..q-1]和 A[q+1..r]，使得 A[q]大于等于 A[p..q-1]中的每个元素，小于 A[q+1..r]中的每个元素。q 的值在划分过程中计算。

递归求解：通过递归的调用快速排序，对子数组 A[p..q-1]和 A[q+1..r]分别排序。

合并：快速排序在原地排序，故不需合并操作。

【问题 1】（6 分）

下面是快速排序的伪代码，请填补其中的空缺。伪代码中的主要变量说明如下：

A：待排序数组

p, r：数组元素下标，从 p 到 r

q：划分的位置

x：枢轴元素

i：整型变量，用于描述数组下标。下标小于或等于 i 的元素的值小于或等于枢轴元素的值

j：循环控制变量，表示数组元素下标

```
QUICKSORT(A, p, r){
    if (p < r){
        q = PARTITION(A,p,r) ;
        QUICKSORT(A, p, q-1);
        QUICKSORT(A, q+1, r);
    }
}
PARTITION(A, p, r){
    x = A[r];   i = p – 1;
    for (j = p ;   j ≤ r – 1;   j++){
        if (A[j] ≤ x){
                i = i + 1 ;
                交换 A[i] 和 A[j]
        }
    }
    交换 （1） 和 （2）   //注：空（1）和空（2）答案可互换，但两空全部答对方可得分
    return （3）
}
```

【问题 2】（5 分）

（1）假设要排序包含 n 个元素的数组，请给出在各种不同的划分情况下，快速排序的时间复杂度，用 O 记号。最佳情况为__（4）__，平均情况为__（5）__，最坏情况为__（6）__。

（2）假设要排序的 n 个元素都具有相同值时，快速排序的运行时间复杂度属于哪种情况？__（7）__。（最佳、平均、最坏）

【问题 3】（4 分）

（1）待排序数组是否能被较均匀地划分对快速排序的性能有重要影响，因此枢轴元素的选取非常重要。有人提出从待排序的数组元素中随机地取出一个元素作为枢轴元素，下面是随机化快速排序划分的伪代码—利用原有的快速排序的划分操作，请填充其中的空缺处。其中，RANDOM(i, j)表示随机取 i 到 j 之间的一个数，包括 i 和 j。

```
RANDOMIZED-PARTITION(A,p,r){
    i = RANDOM(p,r);
    交换__（8）__和__（9）__;//注：空（8）和空（9）答案可互换，但两空全部答对方可得分
    return PARTITION(A,p,r);
}
```

(2) 随机化快速排序是否能够消除最坏情况的发生？__（10）__。（是或否）

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】

栈(Stack)结构是计算机语言实现中的一种重要数据结构。对于任意栈，进行插入和删除操作的一端称为栈顶(Stack Top)，而另一端称为栈底(Stack Bottom)。栈的基本操作包括：创建栈(NewStack)、 判断栈是否为空(IsEmpty)、判断栈是否已满(IsFull)、获取栈顶数据(Top)、压栈/入栈(Push)、弹栈/出栈(Pop)。

当设计栈的存储结构时，可以采取多种方式。其中，采用链式存储结构实现的栈中各数据项不必连续存储（如图 5-1）。

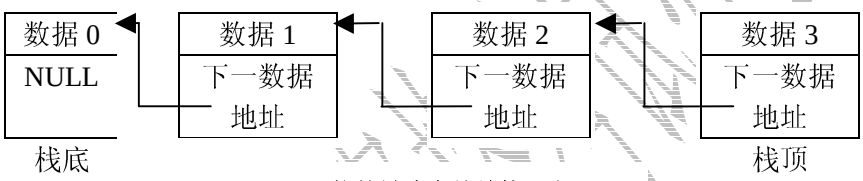


图 5-1　栈的链式存储结构示意图

以下 C 代码采用链式存储结构实现一个整数栈操作。

【C 代码】

```
  typedef struct List {
    int data;              // 栈数据
    struct List* next;     // 上次入栈的数据地址
  }List;

  typedef struct Stack {
    List* pTop;  // 当前栈顶指针
  }Stack;

Stack* NewStack() { return (Stack*)calloc(1,sizeof(Stack)); }

int IsEmpty(Stack* S){ //判断栈S是否为空栈
    if( (1) ) return 1;
    return 0;
}
```

```
int Top(Stack* S){ //获取栈顶数据。若栈为空，则返回机器可表示的最小整数
    if( IsEmpty(S) ) return INT_MIN;
    return  （2）  ;
}

void Push(Stack* S, int theData) {//将数据theData压栈
    List* newNode;
    newNode = (List*)calloc(1, sizeof(List));
    newNode->data = theData;
    newNode->next = S->pTop;
    S->pTop =  （3）  ;
}

void Pop(Stack* S)  {//弹栈
    List* lastTop;
    if( IsEmpty(S) ) return;
    lastTop = S->pTop;
    S->pTop =  （4）  ;
    free(lastTop);
}

#define MD(a)  a<<2

int main(){
    int i;
    Stack* myStack;
    myStack = NewStack();
    Push(myStack, MD(1));
    Push(myStack, MD(2));
    Pop(myStack);
    Push(myStack, MD(3)+1);
    while( !IsEmpty(myStack) ){
        printf("%d ", Top(myStack));
        Pop(myStack);
    }
    return 0;
}
```

以上程序运行时的输出结果为：（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】

已知某企业欲开发一家用电器遥控系统，即用户使用一个遥控器即可控制某些家用电器的开与关。遥控器如图 6-1 所示。该遥控器共有 4 个按钮，编号分别是 0 至 3，按钮 0 和 2 能够遥控打开电器 1 和电器 2，按钮 1 和 3 则能遥控关闭电器 1 和电器 2。由于遥控系统需要支持形式多样的电器，因此，该系统的设计要求具有较高的扩展性。 现假设需要控制客厅电视和卧室电灯，对该遥控系统进行设计所得类图如 6-2 所示。

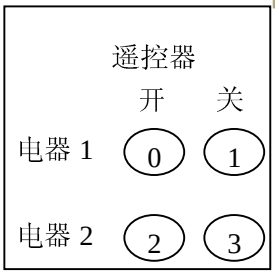


图 6-1 遥控器

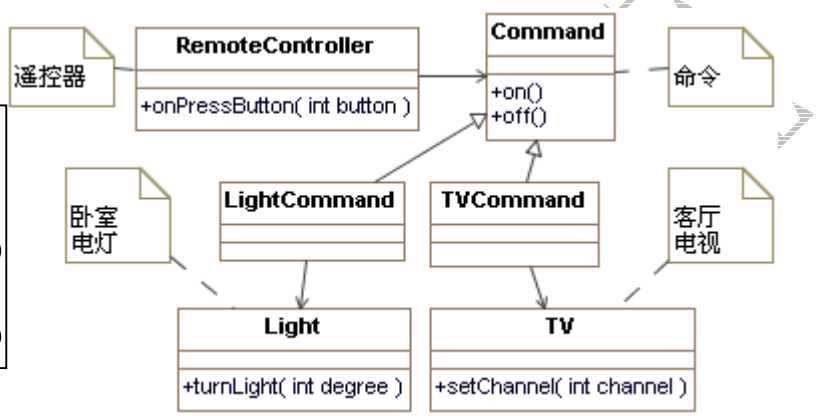


图 6-2 设计类图

图 6-2 中，类 RomoteController 的方法 onPressButton(int button)表示当遥控器按键按下时调用的方法，参数为按键的编号；Command 接口中 on 和 off 方法分别用于控制电器的开与关；Light 中 turnLight(int degree)方法用于调整电灯灯光的强弱，参数 degree 值为 0 时表示关灯，值为 100 时表示开灯并且将灯光亮度调整到最大；TV 中 setChannel(int channel)方法表示设置电视播放的频道，参数 channel 值为 0 时表示关闭电视，为 1 时表示开机并将频道切换为第 1 频道。

【C++代码】

```
class Light{    //电灯类
public:
    void turnLight(int degree){ //调整灯光亮度，0 表示关灯，100 表示亮度最大};
};
class TV{ //电视机类
public:
    void setChannel(int channel){//调整电视频道，0 表示关机，1 表示开机并切换到 1 频道};
};
class Command{ //抽象命令类
public:
    virtual void on()=0;
    virtual void off()=0;
};
```

```
class RemoteController{    //遥控器类
protected:
    Command *commands[4]; //遥控器有 4 个按钮，按照编号分别对应 4 个 Command 对象
public:
    void onPressButton(int button){    //按钮被按下时执行命令对象中的命令
        if(button % 2 == 0)commands[button]->on();
        else commands[button]->off();
    }
    void setCommand(int button,Command * command){
        （1） = command; //设置每个按钮对应的命令对象
    }
};
class LightCommand : public Command{    //电灯命令类
protected:      Light *light;    //指向要控制的电灯对象
public:
    void on(){light->turnLight(100);};
    void off(){light->（2）;};
    LightCommand(Light * light){this->light = light;};
};
class TVCommand : public Command{ //电视机命令类
protected:      TV * tv;     //指向要控制的电视机对象
public:
    void on(){tv->（3）;};
    void off(){tv->setChannel(0);};
    TVCommand(TV * tv){ this->tv = tv; };
};
void main(){
    Light light;     TV tv; //创建电灯和电视对象
    LightCommand lightCommand(&light);
    TVCommand tvCommand(&tv);
    RemoteController remoteController;
    remoteController.setCommand(0,（4）);    //设置按钮 0 的命令对象
    …//此处省略设置按钮 1、按钮 2 和按钮 3 的命令对象代码
}
```

本题中，应用命令模式能够有效让类（5）和类（6）、类（7）之间的耦合性降至最小。

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】

已知某企业欲开发一家用电器遥控系统，即用户使用一个遥控器即可控制某些家用电器的开与关。遥控器如图 7-1 所示。该遥控器共有 4 个按钮，编号分别是 0 至 3，按钮 0 和 2 能够遥控打开电器 1 和电器 2，按钮 1 和 3 则能遥控关闭电器 1 和电器 2。由于遥控系统需要支持形式多样的电器，因此，该系统的设计要求具有较高的扩展性。 现假设需要控制客厅电视和卧室电灯，对该遥控系统进行设计所得类图如 7-2 所示。

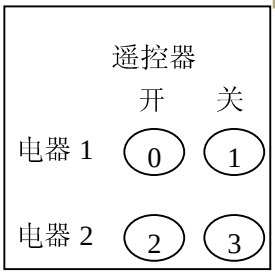


图 7-1 遥控器

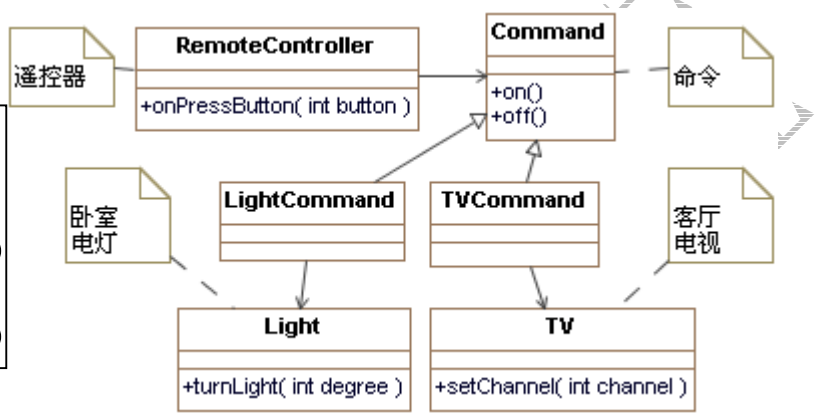


图 7-2 设计类图

图 7-2 中，类 RomoteController 的方法 onPressButton(int button)表示当遥控器按键按下时调用的方法，参数为按键的编号；Command 接口中 on 和 off 方法分别用于控制电器的开与关；Light 中 turnLight(int degree)方法用于调整电灯灯光的强弱，参数 degree 值为 0 时表示关灯，值为 100 时表示开灯并且将灯光亮度调整到最大；TV 中 setChannel(int channel)方法表示设置电视播放的频道，参数 channel 值为 0 时表示关闭电视，为 1 时表示开机并将频道切换为第 1 频道。

【Java 代码】

```
class Light{    //电灯类
    public void turnLight(int degree){ //调整灯光亮度，0表示关灯，100表示亮度最大}
};
class TV{ //电视机类
    public void setChannel(int channel){// 0表示关机，1表示开机并切换到1频道 }
};
interface Command{ //抽象命令类
    void on();
    void off();
};
```

```
class RemoteController{    //遥控器类
    protected Command []commands = new Command[4];
    //遥控器有4个按钮，按照编号分别对应4个Command对象
    public void onPressButton(int button){
    //按钮被按下时执行命令对象中的命令
        if(button % 2 == 0)commands[button].on();
        else commands[button].off();
    }
    public void setCommand(int button, Command command){
        （1）  = command; //设置每个按钮对应的命令对象
    }
};
class LightCommand implements Command{    //电灯命令类
    protected Light light;    //指向要控制的电灯对象
    public void on(){light.turnLight(100);};
    public void off(){light.（2）;};
    public LightCommand(Light light){this.light = light;};
};
class TVCommand implements Command{ //电视机命令类
    protected TV tv;     //指向要控制的电视机对象
    public void on(){tv.（3）;};
    public void off(){tv.setChannel(0);};
    public TVCommand(TV    tv){this.tv = tv;};
};
public class rs{
    public static void main(String []args){
        Light light = new Light();      TV tv = new TV();//创建电灯和电视对象
        LightCommand lightCommand = new LightCommand(light);
        TVCommand tvCommand = new TVCommand(tv);
        RemoteController remoteController = new RemoteController();
        //设置按钮和命令对象
        remoteController.setCommand(0,（4）);
        …//此处省略设置按钮 1、按钮 2 和按钮 3 的命令对象代码
    }
}
```

本题中，应用命令模式能够有效让类（5）和类（6）、类（7）之间的耦合性降至最小。