某停车场运营方为了降低运营成本，减员增效，提供良好的停车体验，欲开发无人值守停车系统，该系统的主要功能是：

1.信息维护。管理人员对车位（总数、空余车位数等）计费规则等基础信息进行设置。

2.会员注册。车主提供手机号、车牌号等信息进行注册，提交充 值信息（等级、绑定并授权支付系统进行充值或交费的支付账号） 不同级别和充值额度享受不同停车折扣点。

3.车牌识别。当车辆进入停车场时，若有（空余车位数大于 1），自动识别车牌号后进行道闸控制，当车主开车离开停车场时，识别车牌号，计费成功后，请求道闸控制。

4.计费。更新车辆离场时间，根据计费规则计算出停车费用，若 车主是会员，提示停车费用：若储存余额够本次停车费用，自动 扣费，更新余额；若储值余额不足，自动使用授权缴费账号请求 支付系统进行支付，获取支付状态。若非会员临时停车，提示停 车费用，车主通过扫描费用信息中的支付码调用支付系统自助交 费，获取支付状态。

5.道闸控制。根据道闸控制请求向道闸控制系统发送放行指令和接收道闸执行状态。

若道闸执行状态为正常放行时， 对入场车辆， 将车牌号及其入场时间信息存入停车记录，修改空余车位数；对 出场车辆更新停车状态，修改空余车位数。当因道闸重置系统出 现问题（断网断电或是故障为抬杠等情况），而无法在规定的时 间内接收到其返回的执行状态正常放行时，系统向管理人员发送 异常告警信息，之后管理人员安排故障排查处理，确保车辆有序 出入停车场。

现采用结构化方法对无人值守停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。

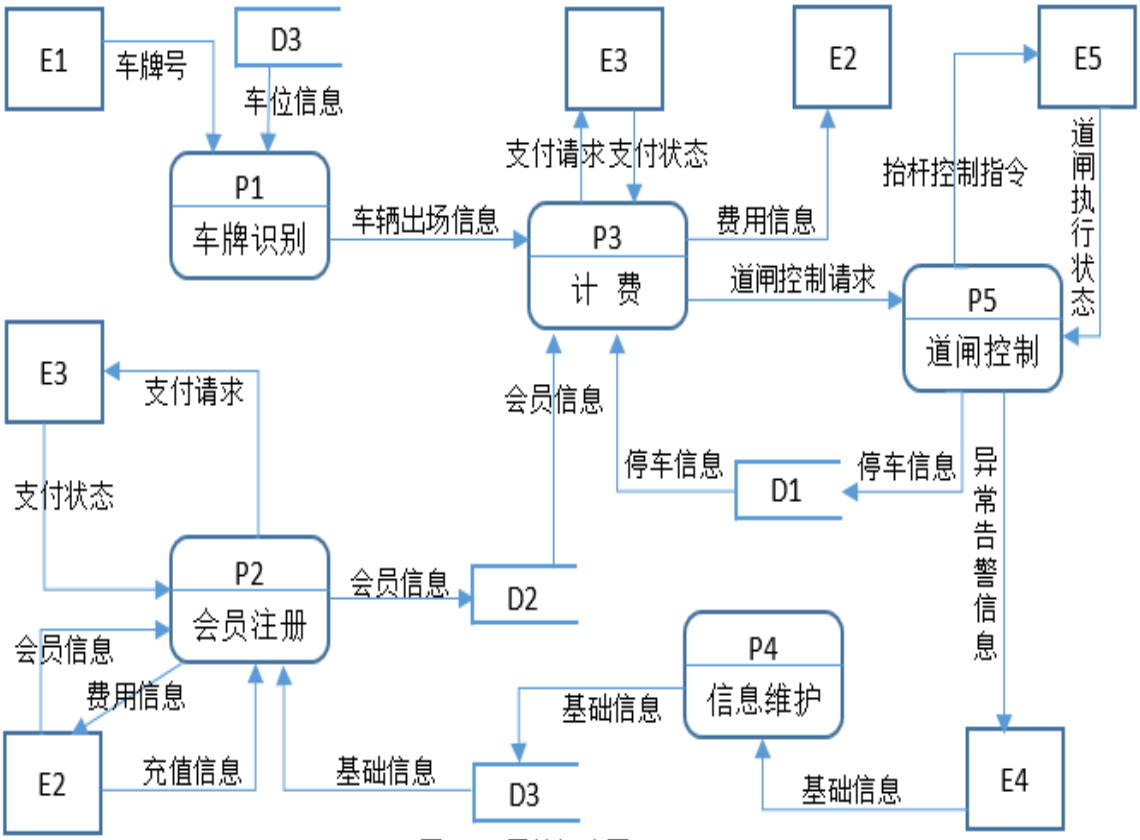


图 1-2 0 层数据流图

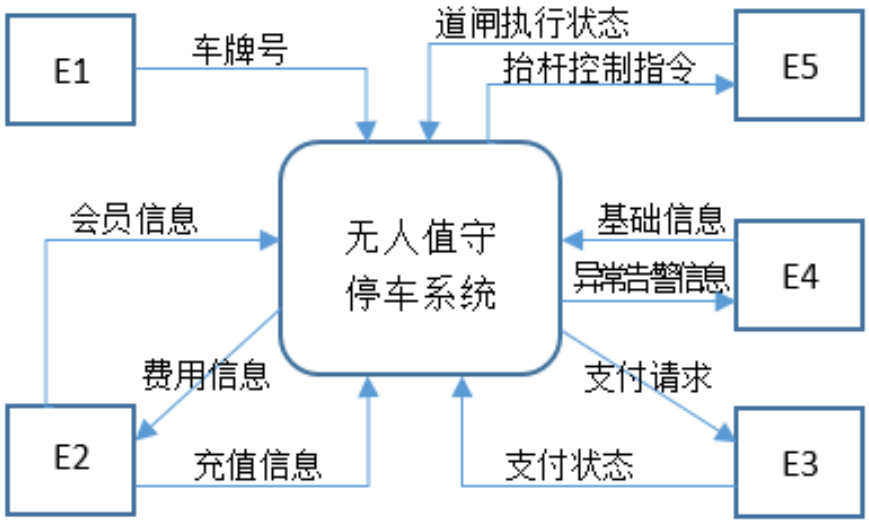


图 1-1 上下文数据流图

问题 1（5 分）

使用说明中的词语，给出图 1-1 中的实体 E1~E5 的名称。

问题 2 (3 分)

使用说明中的词语，给出图 1-2 中的数据存储 D1~D3 的名称

问题 3 (4 分)

根据说明和图中术语，补充图 1-2 中缺失的数据流及其起点和终点。

问题 4 (3 分)

根据说明，采用结构化语言对"道闸控制"的加工逻辑进行描述。

参考答案 问题1 (5分)

E1：车辆　　E2：车主　　E3：支付系统
E4：管理人员　　E5：道闸控制系统

问题2 (3分)

D1：停车记录表　D2：会员信息表　D3：基础信息表

问题4 (3分)

"道闸控制"加工过程

```
IF(道闸执行状态正常)
  IF(车辆入场) THEN
    将车牌号及其入场时间信息存入停车记录，修改空余车位数
  ELSEIF(车辆出场) THEN
    更新停车状态，修改空余车位数
  ENDIF
ELSESIF(未在规定的时间内接收到其返回的执行状态正常放行) THEN
  向管理人员发送异常告警信息
ENDIF
```


序号	名称	起点和终点
1	计费规则信息	D3--P3
2	道闸控制请求	P1--P5
3	更新车位信息	P5--D3
4	更新余额	P3--D2

试题三：

某中医医院拟开发一套线上抓药 APP，允许患者凭借该医院医生 开具的处方线上抓药，并提供免费送药上门服务。该系统的主要 功能描述如下：

(1)注册。患者扫描医院提供的二维码进行注册，注册过程中，患者需提供其病历号，系统根据病历号自动获取患者基本信息。

(2)登录。已注册的患者可以登录系统进行线上抓药，未注册的患 者系统拒绝其登陆。

(3)确认处方。患者登录后，可以查看医生开具的所有处方。患者选择需要抓药的处方和数量（需要抓几副药），同时说明是否需 要煎制。选择取药方式：自行到店取药或者送药上门，若选择送 药上门，患者需要提供提供收货人姓名、联系方式和收货地址。 系统自动计算本次抓药的费用，患者可以使用微信或支付宝等支 付方式支付费用。支付成功之后，处方被发送给药师进行药品配 制。

(4)处理处方。药师根据处方配置好药品。若患者要求煎制，药师 对配置好的药品进行煎制。煎制完成，药师将该处方设置为已完 成。若患者选择的是自行取药，取药后确认已取药。

(5)药品派送。处方完成后，对于选择送药上门的患者，系统将给 快递人员发送药品配送信息，等待快递人员取药；并给患者发送 收货验证码。(6)送药上门。快递人员将配制好的药品送到患者指定的收货地址。 患者收货时，向快递人员出示收货验证码，快递人员使用该验证 码确认药品已送到。

现采用面向对象分析与设计方法开发上述系统，得到如图 3-1 所示 的用例图以及图 3-2 所示的类图。

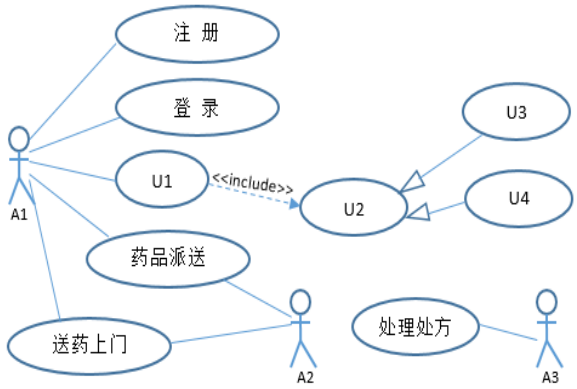


图 3-1 用例图

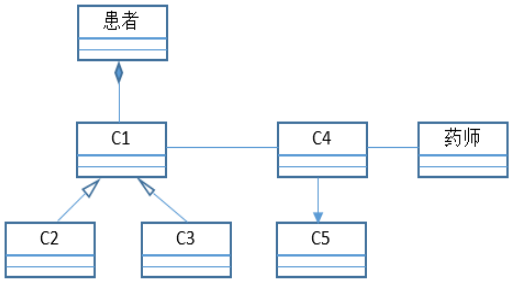


图 3-2 类图

问题 1 (7 分)

根据说明中的描述，给出图 3-1 中 A1~A3 所对应的参与者名称和 U1~U4 处所对应的用例名称。

问题 2(5 分)

根据说明中的描述，给出图 3-2 中 C1~C5 所对应的类名。

问题 3(3 分)

简要解释用例之间的 include, extend 和 generalize 关系的内涵。

解析：

泛化：是一种一般与特殊、一般与具体之间关系的描述。

关联：表示类与类之间的联接，它使一个类知道另一个类的属性和方法。 聚合：是关联关系的一种特例，是强的关联关系。聚合是整体和部分之间的关系，整体与个体可以有各自的生命周期。

组合：也是关联关系的一种特例。组合是一种整体与部分的关系，比聚合更强。部分与整体的生命周期一致，整体的生命周期结束也就意味着部分的生命周期结束。

-

参考答案

问题1 (7分)

A1：患者：　　A2：快递人员　　A3：药师

U1：确认处方　　U2：支付方式　　U3：微信支付

U4：支付宝支付　（U3U4可以互换）

问题2(5分)

C1：支付方式　C2：微信支付　　C3：支付宝支付

C4：处方　　C5：药品　　（C2C3可以互换）

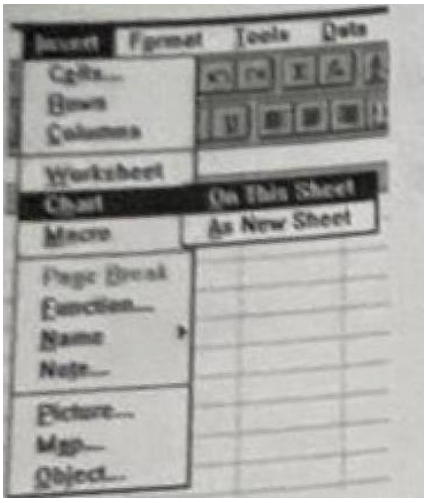
-

问题三

- include：是一种依赖关系，加了版型<<include>>，两个以上用例有共同 功能，可分解到单独用例，其中这个提取出来的公共用例称为抽象用例， 形成包含依赖；执行基本用例时，每次都必须调用被包含的用例。
- extend：如果一个用例明显地混合了两种或两种以上的不同场景，即根 据情况可能发生多种事情，则可以断定将这个用例分为一个主用例和一 个或多个辅用例进行描述可能更加清晰。
- generalize 泛化关系：当多个用例共同拥有一种类似的结构和行为的时候， 可以将它们的共性抽象成为父用例，其他的用例作为泛化关系中的子用 例。在用例的泛化关系中，子用例是父用例的一种特殊形式，子用例继承了父用例所有的结构、行为和关系。

试题五（15 分）

层叠菜单是窗口风格的软件系统中经常采用的一种系统功能组织 方式。层叠菜单(如图 5-1 示例)中包含的可能是一个菜单项(直接对 应某个功能) ，也可能是一个子菜单。



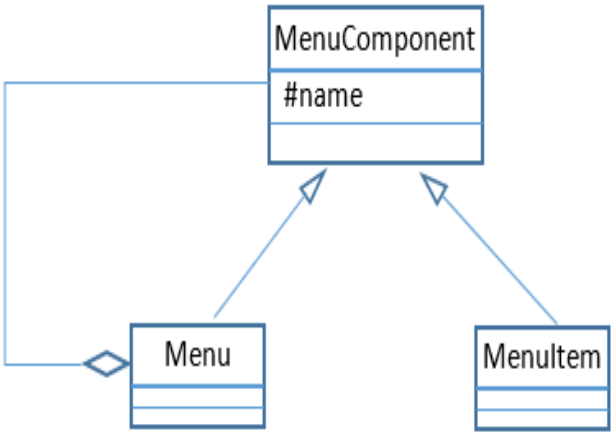


图 5-2 类图

C++代码

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

class MenuComponent {//构成层叠菜单的元素
( 1 );
   string name;       //菜单项或子菜单名称
public;
   void printMenu(){ cout << name; }
   ( 2 );
   virtual void removeMenuElement(MenuComponent *element)=0;
   ( 3 );
};
```

```
class MenuItem: public MenuComponent{
public:
    MenuItem(string name){ this->name=name;}
    void addMenuElement(MenuComponent *element){ return; }
    void removeMenuElement(MenuComponent *element) { return; }
    list<MenuComponent*> *getElement(){ return NULL;}
}
class Menu: public MenuComponent{
private:
   ( 4 )
public:
   Menu(string name) { this->name=name;}
   void addMenuElement(MenuComponent *element){ elementListpush_bach(element);}
   void removeMenuElement(menuComponent *element){ elementListremove_bach(element);}
   list<MenuComponent*> *getElement(){ return &elementList;}
};

int main(){
   MenuComponent *mainMenu = new Menu("Insert");
   MenuComponent *subMenu = new Menu("chart");
   MenuComponent *element = new MenuItem("On This Sheet");
   ( 5 );
   subMenu->addMenuElement(element);
   return 0;
}
```

解析：

UML中，可见性分为4级：

1、public 公用的，用 + 前缀表示，该属性对所有类可见

2、protected 受保护的，用 # 前缀表示，对该类的子孙可见

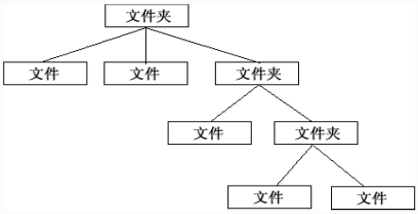
3、private 私有的，用 - 前缀表示，只对该类本身可见

4、package 包的，用 ~ 前缀表示，只对同一包声明的其他类可见

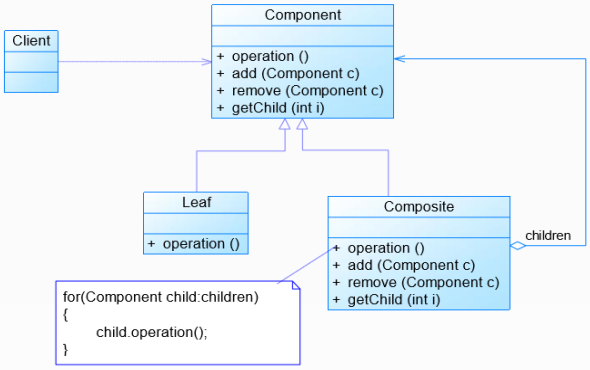
8.组合模式（Composite）

将对象组合成树形结构以表示“部分-整体”的层次结构。使得用户对单个对象和组合对象的使用具有一致性。

组合模式描述了如何将容器对象和叶子对象进行递归组合，使得用户在使用时无须对它们进行区分，可以一致地对待容器对象和叶子对象。



组合模式结构



组合模式包含如下角色：

- Component: 抽象构件
- Leaf: 叶子构件
- Composite: 容器构件
- Client: 客户类

参考答案

(1)protected

(2)irtual void addMenuElement(MenuComponent *element)=0;

(3)virtual list<MenuComponent *> getElement()=0;

(4)list<MenuComponent *> elementList;

(5)mainMenu->addMenuElement(subMenu);

12.4 强化训练

12.4.1 综合知识试题

试题 1

Extreme Programming (XP) is a discipline of software development with __(1)__ of simplicity, communication, feedback and courage. Successful software development is a team effort - not just the development team, but the larger team consisting of customer, management and developers. XP is a simple process that brings these people together and helps them to success together. XP is aimed primarily at object-oriented projects using teams of a dozen or fewer programmers in one location. The principles of XP apply to any __(2)__ project that needs to deliver quality software rapidly and flexibly.

An XP project needs a __(3)__ customer to provide guidance. Customers, programmers, managers, are all working __(4)__ to build the system that's needed. Customers - those who have software that needs to be developed - will learn simple, effective way to __(5)__ what they need, to be sure that they are getting what they need, and to steer the project to success.

(1) A. importance B. keys C. roles D. values

(2) A. small-sized B. moderately-sized
C. large-sized D. huge-sized

(3) A. part-time B. casual C. seldom D. full-time

(4) A. together B. by themselves C. separately D. alone

(5) A. tell B. know C. communicate D. feedback

试题 2

Ravi, like many project __(6)__, had studied the waterfall model of software development the primary software life-cycle __(7)__. He has all set to use it for an upcoming project, his assignment. However, Ravi found that the waterfall model could not be used because the

customer wanted the software delivered in stages, something that implied that the system had to be delivered and built in __(8)__ and not as __(9)__.

The situation in many other projects is not very different. The real world rarely presents a problem in which a standard process, or the process used in a previous project, is the best choice. To be the most situation, an existing process must be __(10)__ to the new problem.

A development process, even after tailoring, generally cannot handle change requests. To accommodate change requests without losing control of the project, you must supplement the development process with a requirement change management process.

(6)	A. customers	B. managers	C. users	D. administrators
(7)	A. activity	B. procedure	C. process	D. progress
(8)	A. parts	B. modules	C. software	D. a whole
(9)	A. parts	B. modules	C. software	D. a whole
(10)	A. modified	B. used	C. suited	D. tailored

12.4.2 综合知识试题参考答案

【试题 1】

参考答案：(1) D；(2) B；(3) D；(4) A；(5) C。

参考译文：极限编程(XP)是一种软件开发方法，其价值观是简单、沟通、反馈和勇气。成功的软件开发是团队努力的结果，这里的团队不仅仅是开发团队，而是包括了客户、管理者和开发者在内的更大的团队。XP 是一种将上述人员组织起来并帮助他们取得成功的简单的过程。XP 主要针对那些由十几个或更少的在同一地点工作的程序员组成的团队所进行的面向对象的项目。XP 原则适用于需要快速且灵活地交付高质量软件的中等规模项目组。

一个 XP 项目需要一个全程参与的客户给予指导。客户、程序员和项目经理协同工作来构建需要的系统。客户，也就是需要软件的人，将学到简单而有效的沟通方法，来确保获得他们所需要的，从而引导项目走向成功。

【试题 2】

参考答案：(6) B；(7) C；(8) A；(9) D；(10) D。

参考译文：和许多项目经理一样，Ravi 研究了作为主要软件生命周期过程的瀑布模型。他打算在首个任务，也就是即将开始的项目中使用瀑布模型。但是，他发现瀑布模型不能满足要求，原因是客户希望软件分阶段交付，也就意味着系统必须一部分一部分地交付和构建，而不是作为一个整体进行。

这种情况在很多其他项目中也类似。现实世界中，很难有一种标准的过程或在前期的项目中使用的过程作为目前项目的最佳选择。对于大部分情况，必须对已存在的过程进行调整，以适应新的问题。

但是，即使经过调整，一个开发过程也很难处理变化的需求。为了适应变化的需求而不失去对项目的控制，你必须用需求变更管理过程从而对开发过程进行补充。