技能三：UML 建模

考试分值约占 10%（8 分）

教程说明：

红色字体：核心高频（必需熟记）

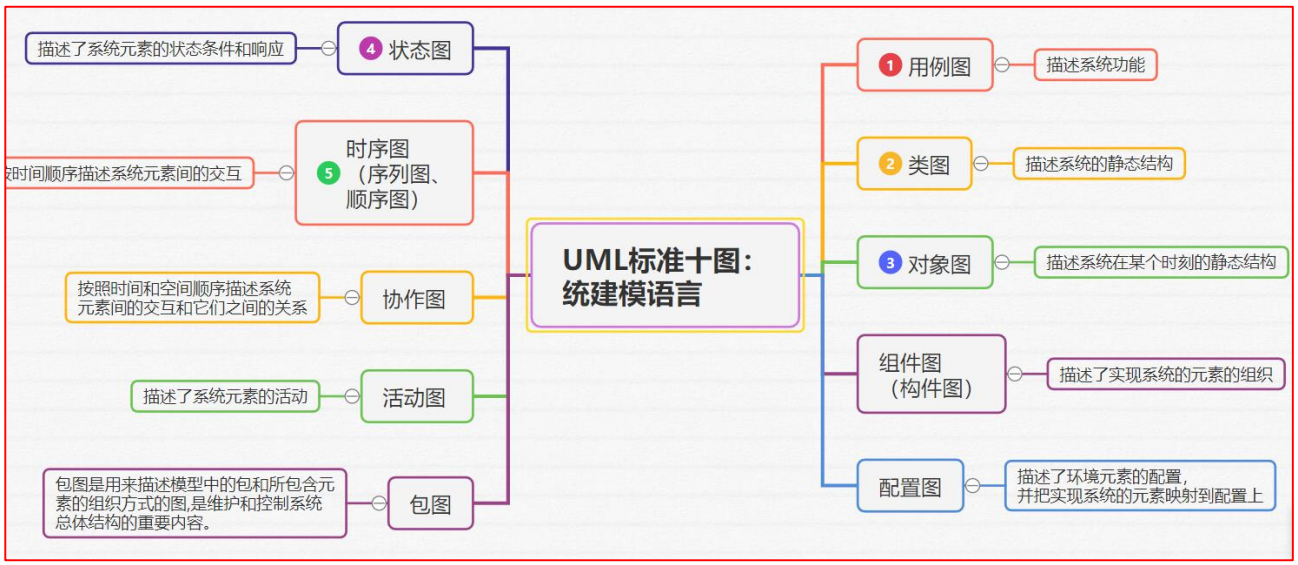
绿色字体：重点内容（必需熟悉）

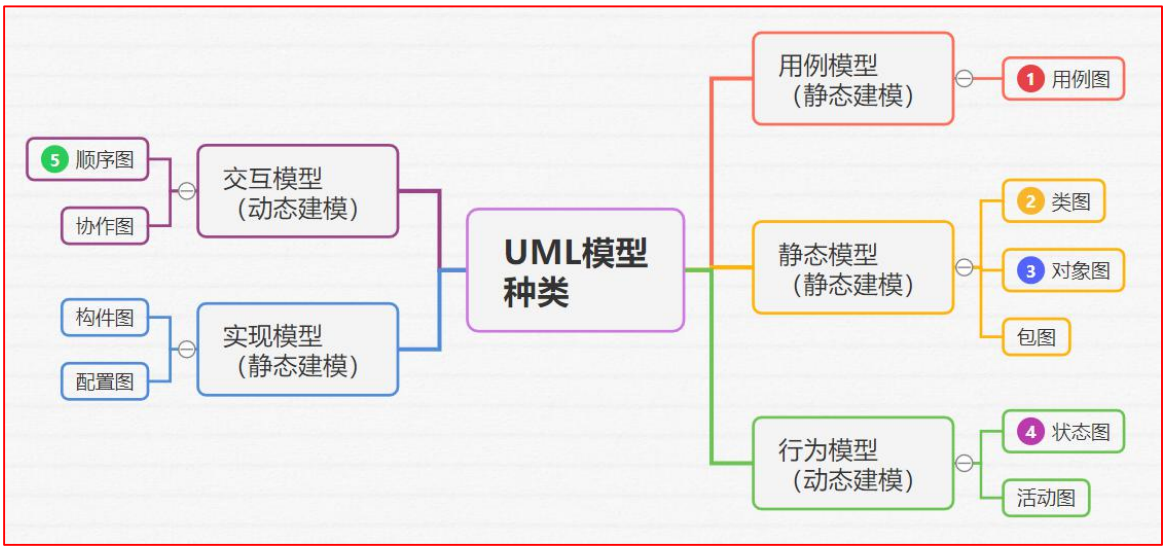
黄色字体：难点内容（尽量掌握）

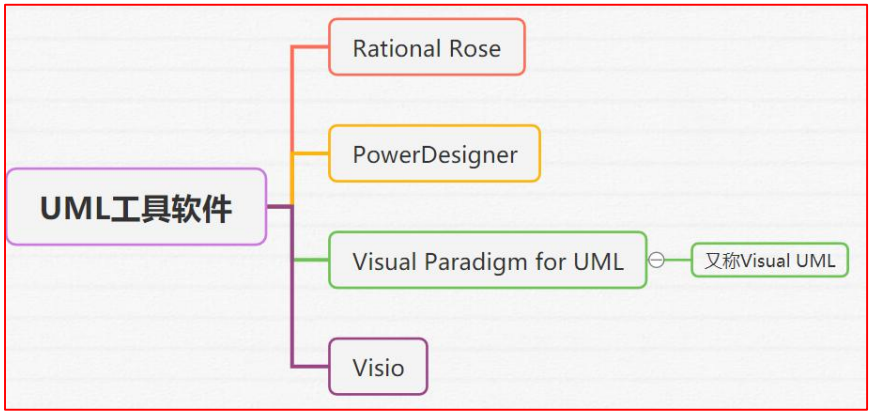
黑色字体：常规内容（必需了解）

1．用例图

UML(统一建模语言)是面向对象建模语言的标准，它可以对任何具有静态结构和动态行为的系统进行建模，它的主要作用是帮助用户进行面向的描述和建模，它可以描述软件从需求分析到软件实现和测试的全过程。







1.1 参与者与用例的基本概述

1、**参与者**

（1）**人与系统在进行交互时能够担任的不同角色称为参与者（actors）**。

（2）参与者一般对应于对系统的一个特定的访问级别，由参与者能够执行的系统功能的类别定义。与用户并不一一对应

（3）举例：管理员、普通工人、经理、客户、销售人员等。

2、**用例**

（1）**是软件工程或系统工程中对系统如何反应外界请求的描述，是一种通过用户的使用场景来获取需求的技术。**

（2）每个用例提供了一个或多个场景，该场景说明了系统是如何和最终用户或其它系统互动，也就是谁可以用系统做什么，从而获得一个明确的业务目标。

（3）组成：**用例名称、参与者、假设/前提条件、后置条件、业务规则、正常的流程、备用流程流**。

1.2 用例描述的格式

1、**用例的描述的格式要求**

（1）**用例名称**：**用例应该被标记，以便它立即描述用例的用途。**动词/名词都行。例如，“订单”。

（2）**参与者**：**确定用例中涉及的主要个人和系统将是谁**。必须事先命名在用例的事件流中使用的任何actor。

（3）**假设/前提条件**：**定义在用例中描述的事件流开始之前世界的状态。**

（4）**后置条件**：**解释用例中描述的事件流发生后世界的状态。**

（5）**业务规则**：**这些是管理或约束流程发生的环境的操作规则。**例如，“下订单只能在上午 9 点到下午 5 点之间发生”是一项业务规则。用例必须遵守这些规则。

（6）**正常的流程**：**定义流程的“快乐路径”，一步一步，顺序编号。**“快乐路径”实质上是指默认流程是什么，而不考虑异常、条件或错误。在后置条件下，应当得出成功的结论。

（7）**备用流程流**：**当存在为用例的结论创建不同路径的主要决策点或异常时，这些备选流程流中的每一个都必须像对普通流程流所做的那样逐步定义。**如果按顺序对正常流程流进行编号，、备用流程在某个步骤中开始，则对备用流程的描述就从那里开始。

2、实例：

参与者：负责人
简要说明： 负责人用来填写和修改家教网站首页的公告，公告最终显示在家教网站的首页上。
前置条件： 负责人已经登陆家教网站管理系统
基本事件流： 1. 负责人鼠标点击“修改公告”按钮 2. 系统出现一个文本框，显示着原来的公告内容 3. 负责人可以在文本框上修改公告，也可以完全删除，重新写新的公告 4. 负责人编辑完文本框，按“提交”按钮，首页公告就被修改 5. 用例终止
其他事件流 A1： 在按“提交”按钮之前，负责人随时可以按“返回”按钮，文本框的任何修改内容都不会影响网站首页的公告
异常事件流： 1. 提示错误信息，负责人确认 2. 返回到管理系统主页面
后置条件： 网站首页的公告信息被修改
注释：无

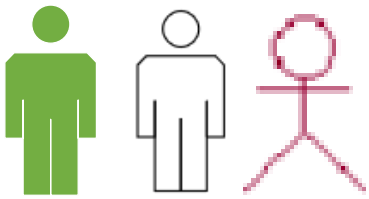
1.3 绘制用例图

1、用例图

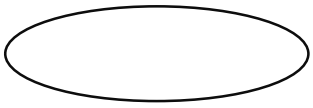
（1）用例图（use case diagram）以图解的形式概括了系统中的不同参与者和用例，并显示了哪些参与者能够参与哪些用例。

（2）图例图元素：

①、**参与者（Actor）**：**小人**



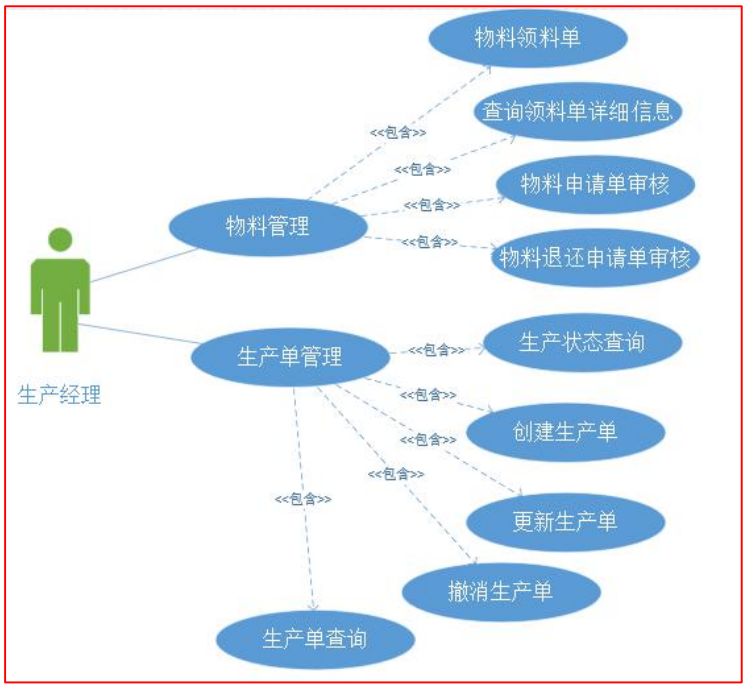
②、**用例**：**椭圆**



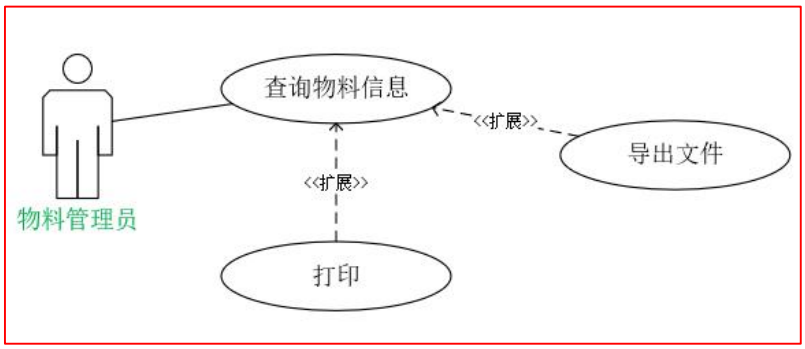
③、以及用例图中对象间到的**关系(包含、扩展和泛化)**：**箭头**

其中关系有**包含、扩展**是用例图中特有的，**泛化**在其他类图中同样存在。

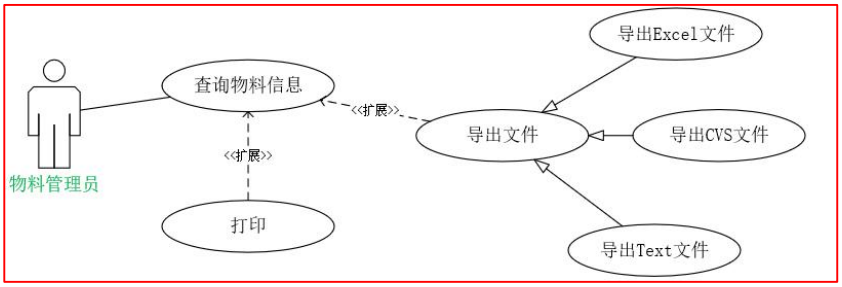
1）包含（<< include>>）用例：**当可以从两个或两个以上的用例中提取公共行为时，应该使用包含的关系来表示它们。**其中这个提取出来的公共用例成为抽象用例，而把原始用例成为基本用例或基础用例。其中“<<include>>”是包含关系的构造型，箭头指向抽象用例。

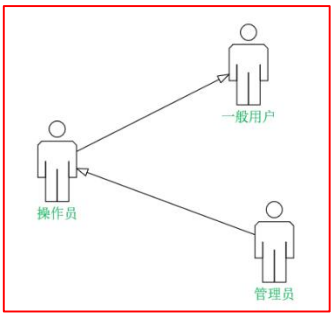


2）扩展(<<extend>>)用例：如果一个用例明显地混合了两种或者两种以上的不同场景，即根据情况可能发生多种分支，则可以将这个用例分为一个基本用例和一个或多个扩展用例，这样可能会使描述更加清晰。扩展用例为基用例添加新的行为。扩展用例可以访问基用例的属性，因此他能根据基用例中扩展点的当前状态来决定是否执行自己。而扩展用例对基用例不可见。



3）泛化用例：当多个用例共同拥有一种类似的结构和行为时，可以将他们的共性抽象成为父用例，其他的用例作为泛化关系的子用例。在用例的泛化关系中，子用例是父用例的一种特殊形式，它继承了父用例的所有结构、行为、关系。其中三角箭头指向父用例。





2．类图和对象图

2.1 类图的基本概述

1、类

（1）**类与对象是面向对象程序设计中概念。**

（2）面向对象的特性：三个基本特征是**封装、继承、多态。**

2、对象：**对象是对客观事物的抽象，类是对对象的抽象。**

3、类与对象的关系：

（1）类是对象的模板，对象是类的实例。

（2）一个类可以有若干个实例。

4、**类图(Class diagram)是显示了模型的静态结构，特别是模型中存在的类、类的内部结构以及它们与其他类的关系等。**

5、**用例图、类图和对象图都是静态图。**

2.2 分析识别类与绘制类图

1、基于用例分析建模过程（替换教学视频中过程描述）：

（1）理解用例模型：理解用例模型和词汇表，适当补充系统内部情况的描述。

（2）**识别分析类：找出可能的能够执行用例行为的分析类。**

（3）定义交互行为：将用例行为分配到分析类中。

（4）**建立分析类图：确定分析类关键属性和责任，定义分析类之间的关系。**

（5）检查分析模型：

①、检查“正确性”：用户是否理解相关术语表，描述的与用户下的定义是否一致等

②、检查“完整性”：每个分析类是否都是用例需要，属性是何时被设置，类型是否符合实际情况等。

③、检查“一致性”：类或用例是否重名，实体是否以同样的细节进行描述等

④、检查“可行性”：是否可行，性能是否符合可靠性需求，是否在相关原形或硬上进行验证。

2、**分析类**

（1）在分析模型中，分析类是概念层次上的内容，用于描述系统中较高层次的对象。

（2）分析类直接及应用逻辑有相关，而不关注于技术实现的问题。

（3）**分析类的类型**：

①、实体类：表示系统存储和管理的永久信息。

②、边界类：表示参与者与系统之间交互。

③、控制类：表示系统在运行过程中的业务控制逻辑。

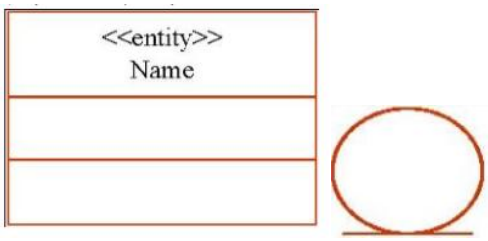
3、**实体类**

（1）**描述必须存贮的信息及其相关行为。**

（2）**通常对应现实世界中的“事物”。**

（3）**实体类及数据库的表对应，类的实例对应于表中的一条记录；类中的属性和记录中的字段对应。**

（4）**实体类的 UML 表示**：



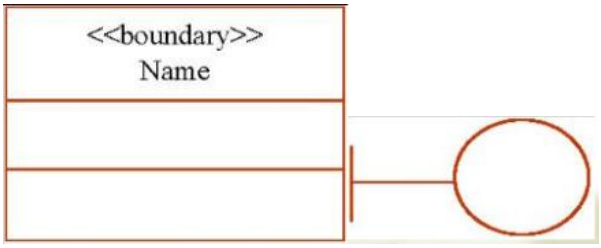
4、**边界类**

（1）**描述外部的参与者与系统之间的交互**

（2）类型：**用户界面、系统接口、设备接口**等

（3）**边界类系统的用户界面，直接跟系统外部参与者交互，与系统进行信息交流。**如网上购物系统中登陆子功能里的登录页面（login.html 或 index.jsp）

（4）**边界类的 UML 表示**：



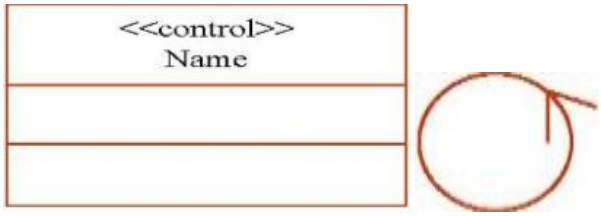
5、**控制类**

（1）**描述一个用例所具有的事件流控制行为**

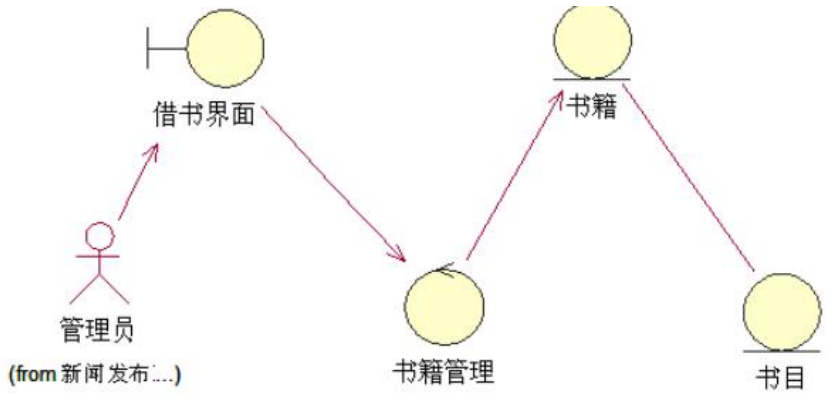
（2）**实现对用例行为的封装，将用例的执行逻辑及边界和实体进行隔离**

（3）**控制类控制系统中对象之间的交互，通常每个用例者是一个控制类**

（4）**控制类的 UML 表示**



6、**熟悉一下图中的各种类**

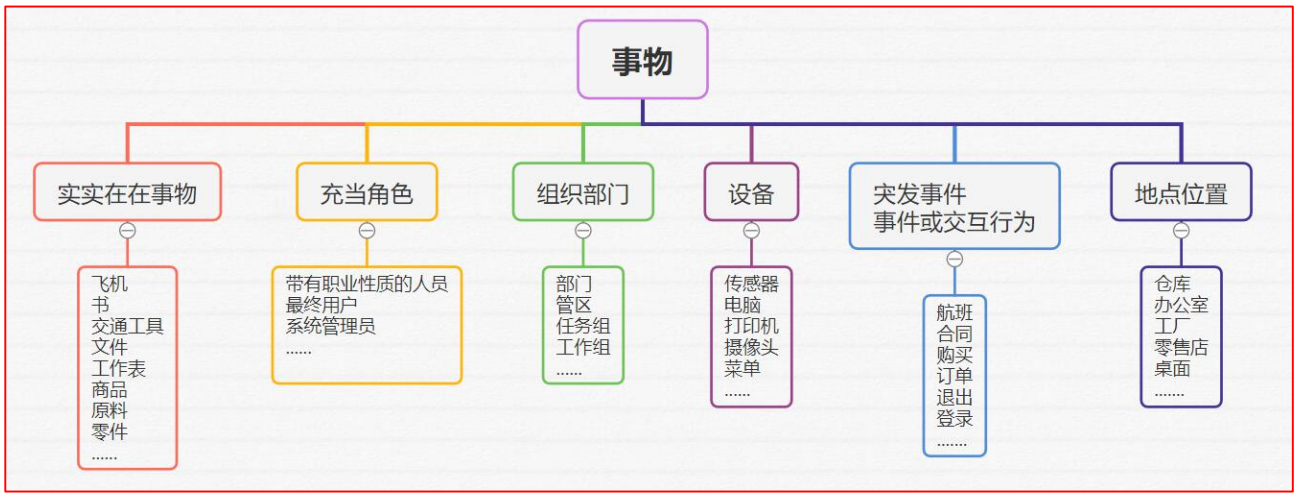


7、**识别分析类**

（1）**识别实体类**：

①、**实体类通常是用例中的参与对象，对应着现实世界中的“事物”**

②、**事物列举**：



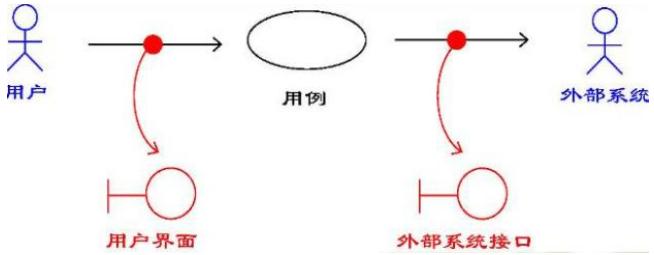
③、识别实体类应当注意的问题

- 实体类的识别质量在很大程度上取决于分析人员书写文档的风格和质量
- 自然语言不是精确的，因此在分析自然语言描述时应该规范化描述文档中的一些措辞，尽量弥补这种不足；
- 在自然语言描述中，名词可以对应类、属性或同义词等多种类型，开发人员需要花费大量时间进行筛选。

（2）识别边界类

①、通常一个参与者与一个用例之间的交互或通信关联对应一个边界类。

②、列举：



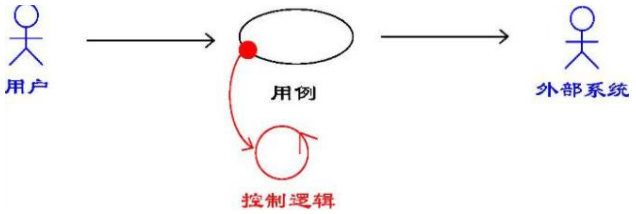
③、识别边界类应当注意的问题

- 边界类应关注于参与者与用例之间交互的信息或响应的事件，不要描述窗口组件等界面的组成元素；
- 在分析阶段，力求使用用户的术语描述界面
- 边界类实例的生命周期并不仅限于用例的事件流,如果两个用例同时与一个参与者交互，那么它们有可能 会有共用一个边界类，以便增加边界类的复用性。

（3）识别控制类

①、控制类负责协调边界类的实体类，通常在现实世界中没有对应的事物。一般来说，一个用例对应一个控制类。

②、列举：



③、识别控制类应当注意的问题

- 当用例比较复杂时，特别是产生分支事件流的情况下，一个用例可以用多个控制类。
- 在有些情况下，用例事件流的逻辑结构十分简单，这时段有必要使用控制类，边界类可以实现用例的行为。
- 如果不同用例包含的任务之间存在着比较密切的联系，则这些用例可以使用一个控制类，其目的是复用相似部分以便降低复杂性。

8、建立分析类图

（1）定义关系

找出分析类之间的关系。

（2）定义属性

①、按照一般常识，找出对象的某些属性；

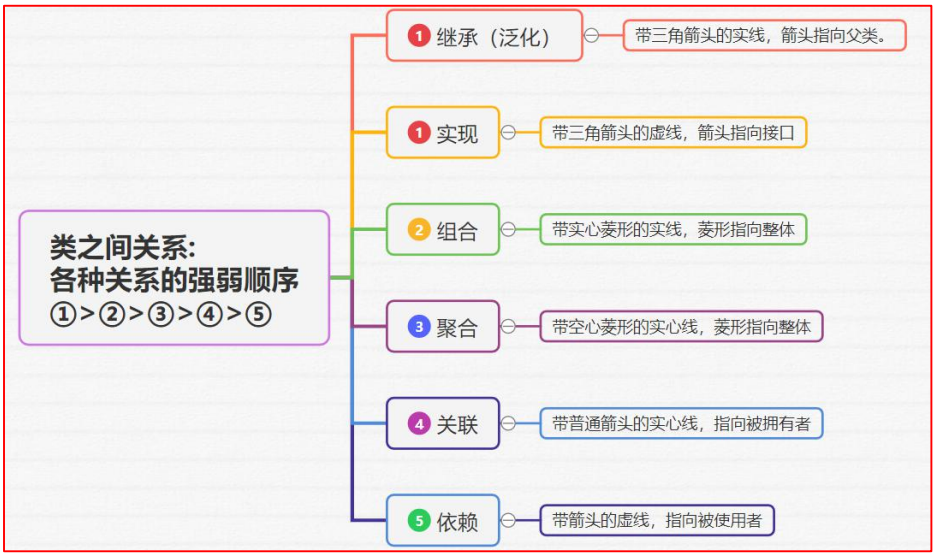
②、认真研究问题域，找出对象的某些属性；

③、根据系统责任的要求，找出对象的某些属性；

④、考虑对象需要系统保存的信息，找出对象的相应属性；

⑤、对象为了服务中实现其功能，需要增设一些属性；
⑥、识别对象需要区别的状态，考虑是否需要增加一个属性来区别这些状态
⑦、确定属性表示整体及部分结构和实例连接。

9、类之间关系



（1）继承（泛化）关系
①、是一种继承关系，表示一般与特殊的关系，它指定了子类如何特化父类的所有特征和行为。继承是类与类或者接口与接口之间最常见的关系；
②、UML 表示：带三角箭头的实线，箭头指向父类。

③、实例：老虎与动物的关系

（2）实现关系
①、是一种类与接口的关系，表示类是接口所有特征和行为的实现. 一个 class 类实现 interface 接口（可以是多个）的功能；实现是类与接口之间最常见的关系；
②、UML 表示：带三角箭头的实线，箭头指向父类。

③、实例：画笔实现画

（3）**依赖关系**

①、**是一种使用的关系，即一个类的实现需要另一个类的协助，所以要尽量不使用双向的互相依赖.** 一个类 A 使用到了另一个类 B,：而这种使用关系是具有偶然性的、临时性的、非常弱的，但是 B 类的变化会影响到 A；比如某个老师要授课，则需要有这么一门课让他教授，此时老师与课之间的关系就是依赖；

②、**UML 表示：带箭头的虚线，指向被使用者**。

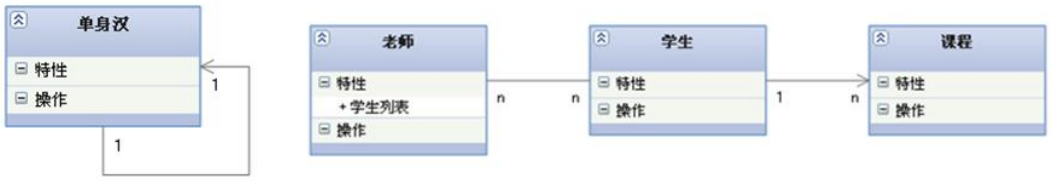
③、实例：现代人与计算机

（4）**关联关系**

①、**是类与类之间的联接，它使一个类知道另一个类的属性和方法。**两个类或者类与接口之间语义级别的一种强依赖关系，这种关系比依赖更强，一般是长期性的，而且双方的关系一般是平等的。比如：老师与学生，丈夫与妻子关联可以是双向的，也可以是单向的。**双向的关联可以有两个箭头或者没有箭头，单向的关联有一个箭头**。

②、**UML 表示：带普通箭头的实心线，指向被拥有者**。

③、实例： 单身汉自己；老师、学生和课程三者之间



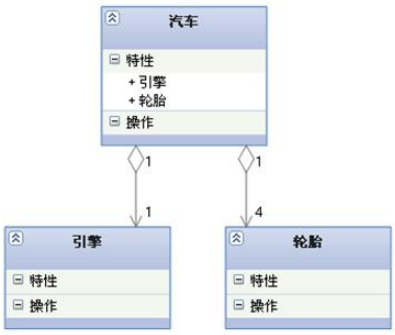
（5）**聚合关系**

①、**是关联关系的一种特例，是强的关联关系.** 聚合是整体与部分的关系，且部分可以离开整体而单独存

在，他们可以具有各自的生命周期，部分可以属于多个整体对象，也可以为多个整体对象共享，比如计算机与 CPU、公司与员工、车和轮胎的关系等；表现在代码层面，和关联关系是一致的，只能从语义级别来区分；

②、**UML 表示：带空心菱形的实心线，菱形指向整体**

③、实例：汽车与引擎和轮胎



（6）**组合关系**

①、**也是关联关系的一种特例，是比聚合关系还要强的关系，也称为强聚合**；他同样体现整体与部分间的关系，但此时整体与部分是不可分的，整体的生命周期结束也就意味着部分的生命周期结束；比如：公司和部门是整体和部分的关系，没有公司就不存在部门合成关系不能共享。

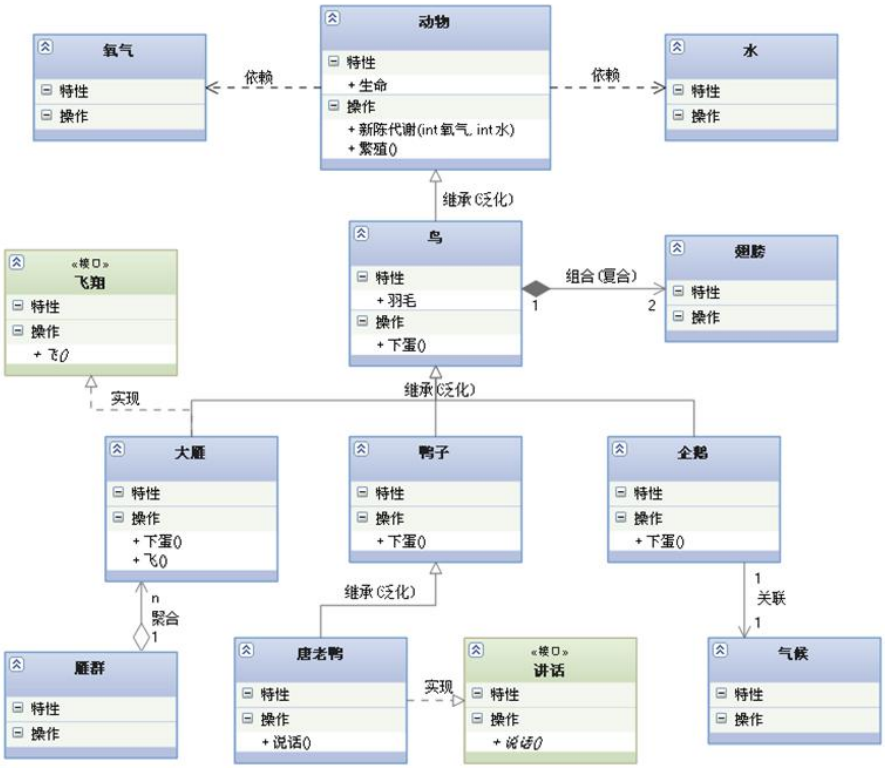
②、**UML 表示：带实心菱形的实线，菱形指向整体**

③、实例：公司与部门之间关系

10、**画类图**

（1）**根据识别出来的类，相关类定义的相关属性及行为（方法），以类之间的关系进行绘制**。

（2）实例：



2.3 绘制对象图

1、**对象图**

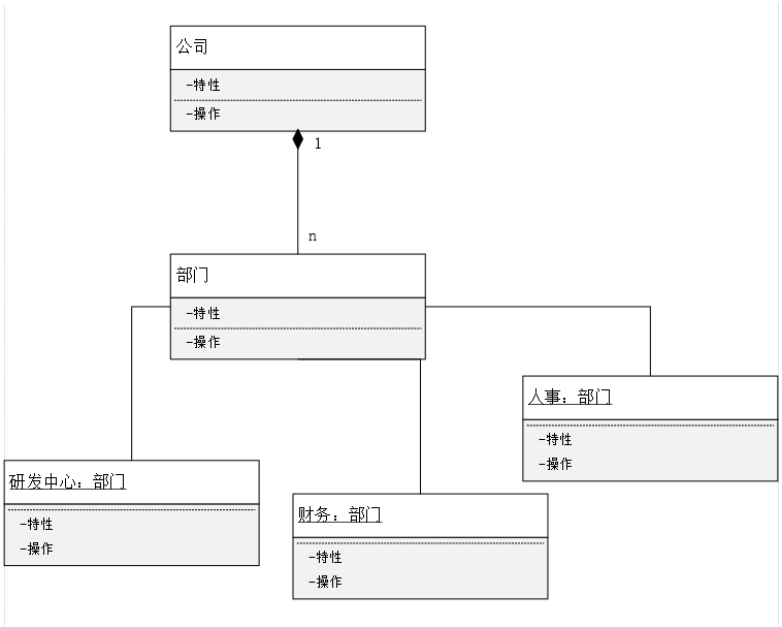
（1）**对象图(Object Diagram) 是显示了一组对象和他们之间的关系**。

（2）**使用对象图来说明数据结构，类图中的类或组件等的实例的静态快照**。

2、画法:

对象名：类名

3、实例:



3．顺序图

3.1 顺序图的基本概述与组成

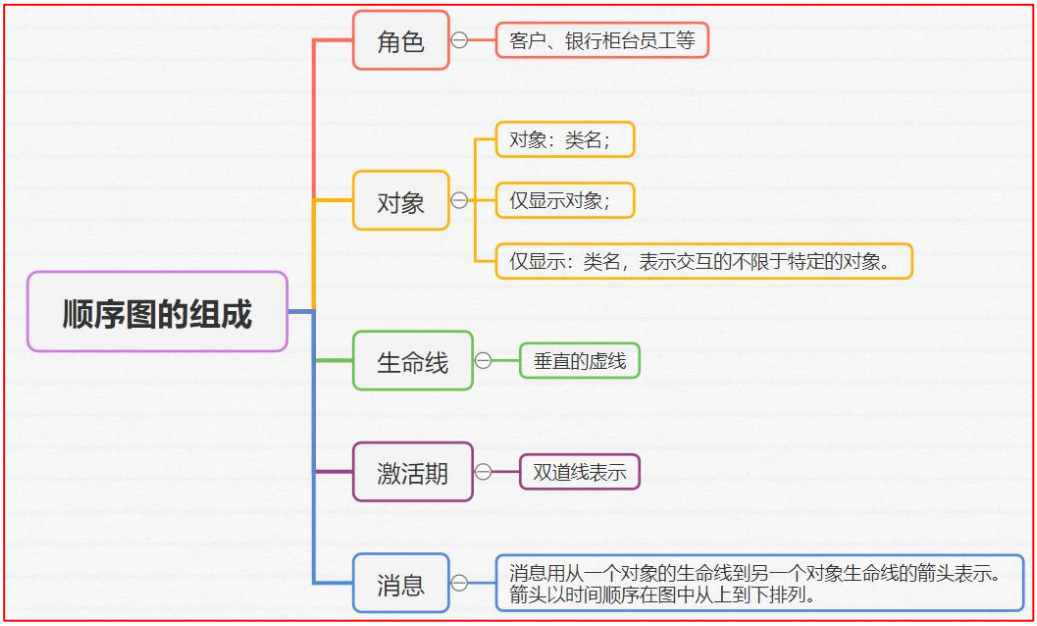
1、**顺序图（序列图、时序图）**

（1）顺序图是交互图的一种形式,它显示对象沿生命线发展,对象之间随时间的交互表示为从源生命线指向目标生命线的消息。

（2）纵向是时间轴，时间沿竖线向下延伸。

（3）横向轴代表了在协作中各独立对象的类元角色。类元角色用生命线表示。当对象存在时，生命线用一条虚线表示，当对象的过程处于激活状态时，生命线是一个双道线。

2、顺序图的组成

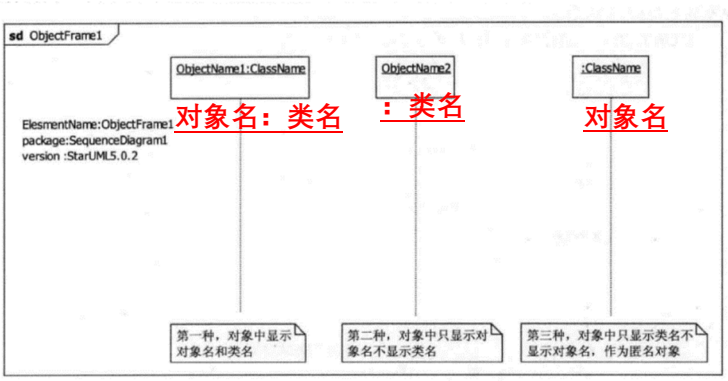


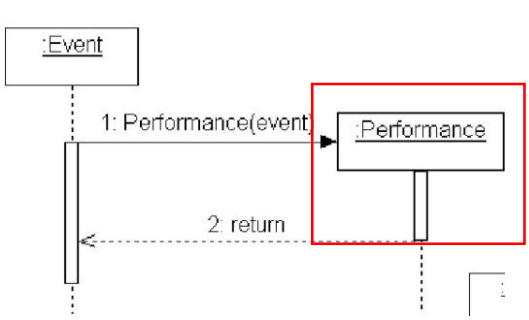
3、对象

（1）顺序图中的对象可以是系统的参与者或者任何有效的系统对象。

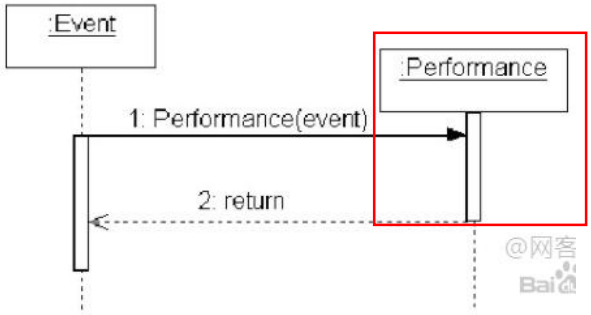
（2）对象的表示形式也和对象图中的对象的表示方式一样，使用包围名称的矩形框来标记，所显示的对象及其类的名称带有下划线，两者使用：隔开，使用“对象名：类名”的形式，对象的下部有一条被称为“生命线”的垂直虚线。

（3）列举：





交互时创建对象



交互时对象已经存在

4、生命线

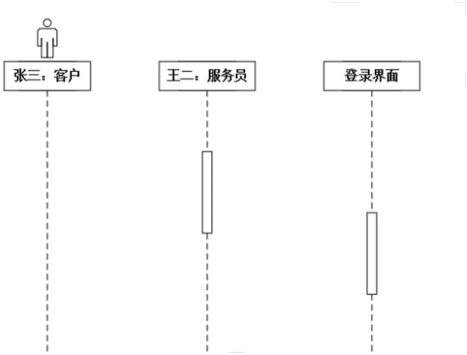
（1）生命线是一条垂直的虚线，用来表示顺序图中的对象在一段时间内的存在。

（2）每个对象的底部中心位置带有生命线。

（3）生命线是一个时间线，从顺序图的顶部一直延伸到底部，所用的时间取决于交互持续的时间，也就是说生命线表现了对象存在的时段。

（4）对象与生命线结合在一起称为对象的生命线。

（5）列举



5、激活期

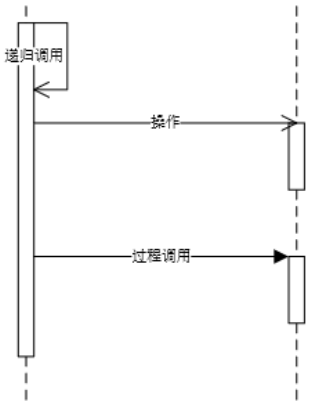
（1）激活是对象操作的执行，它表示一个对象直接地或通过从属操作完成操作的过程。

（2）它对执行的持续时间和执行与其调用者之间的控制关系进行建模。

（3）激活在顺序图中用一个细长的矩形框表示，它的顶端与激活时间对齐而底端与完成时间对齐。

（4）被执行的操作根据不同风格表示成一个附在激活符号旁或在左边空白处的文字标号。

（5）列举:



3.2 顺序图中的消息类型

1、顺序图中消息

（1）消息用从一个对象的生命线到另一个对象生命线的箭头表示。

（2）箭头以时间顺序在图中从上到下排列。

（3）消息可以激发某个操作、创建或解构某对象。

2、消息类型

（1）同步消息

（2）返回消息

（3）自关联消息

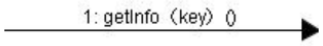
（4）异步消息

3、同步消息

（1）一个对象向另一个对象发出同步消息后，将处于阻塞状态，一直等到另一个对象的回应。

（2）消息的名称是被调用者的方法名。

（3）UML 表示：带有实心箭头实线

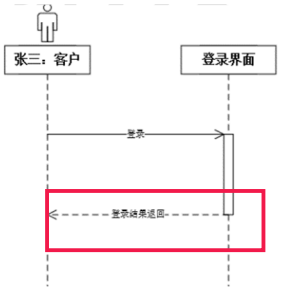


4、返回消息

（1）返回（return）消息表示从过程调用返回

（2）非必要的，是可选的。

（3）UML 表示：带普通箭头虚线表示



5、自关联消息

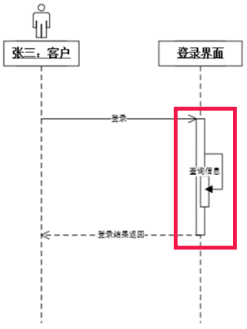
（1）自身调用自身的方法。

（2）即自我调用的同步消息。

（3）一般是用来描述对象内部函数的互相调用。

（4）UML 表示：带普通箭头实线表示

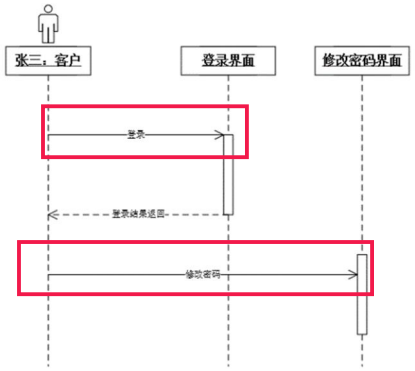
（5）列举：



6、**异步消息**

（1）**一个对象向另一个对象发出异步消息后，这个对象可以进行其他的操作，不需要等到另一个对象的响应。**

（2）**UML 表示：带普通箭头实线表示**



3.3 绘制顺序图

1、画顺序图步骤：

(1)确定交互的范围

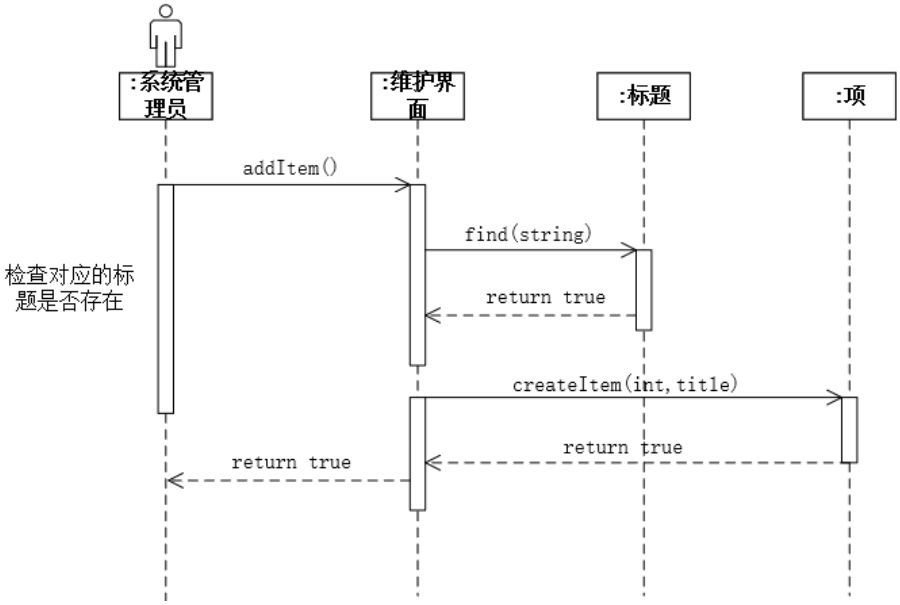
(2)确定参与交互的活动者与对象

(3)确定活动者、对象的生存周期

(4)确定交互中产生的消息

(5)细化消息的内容

2、实例：系统管理员添加书籍的时序图



4．状态图

4.1 状态机的含义

1、状态机

（1）**状态机是一种行为，它说明对象在其生命周期中响应事件所经历的状态变化序列以及对那些时间的响应。**

（2）**一般情况下，一个状态机依附于一个类，用来描述这个类的实例的状态及其转换，和对接收到的事件所做出的响应。**

（2）此外，状态机也可以依附于用例、操作、协作等元素上，描述它们的执行过程。

（3）**状态机从对象的初始状态开始，响应事件并执行某些动作，从而引起状态的转换；在新状态下又继续响应事件并执行动作，如此循环进行到对象的终结状态。**

2、状态机的组成

状态机主要由状态、转换、事件、动作和活动 5 部分组成。

（1）**状态**表示对象的生命周期中的一种条件或情况。

（2）**转换**表示两种状态间的一种关系。

（3）**事件**表示在某一时间与空间下所发生的有意义的事情。

（4）**动作**表示一个可执行的原子操作，是 UML 能够表达的最小计算单元

（5）**活动**表示状态机中的非原子执行，一般由一系列动作组成。

4.2 状态机图中的基本标记符

1、**状态机图**

（1）**状态机图用于对系统的动态方面进行建模，适合描述一个对象在其生命周期中的各种状态及状态的转换。**

（2）**状态图和活动图是状态机的两种表现形式，均可以表现行为状态。**

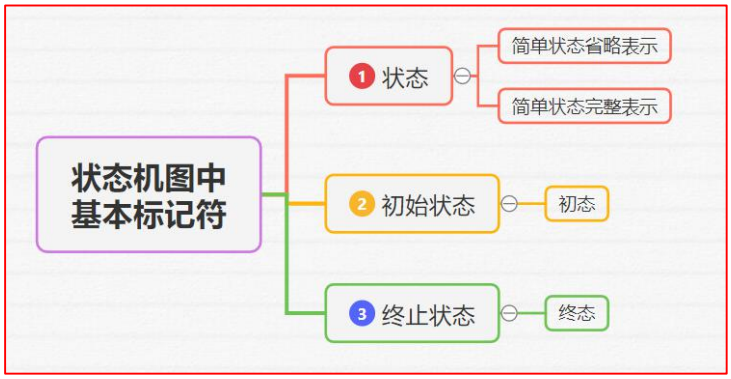
（4）**状态图=状态+迁移**

（5）**一个状态图描述一个状态机。**

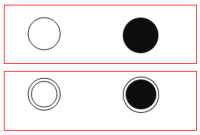
（6）**状态图表现从一个状态到另一个状态的控制流**

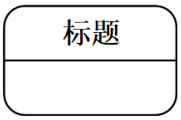
2、状态机图中基本标记符

（1）**状态**

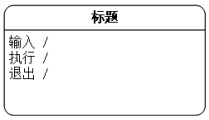


（2）**UML 表示**





简单状态省略表示



简单状态完整表示

4.3 绘制状态图

1、实例

新生入学后，学校在三个月内按照国家招生规定对其进行复查。复查合格者予以注册，取得学籍。复查不合格者，学校区别情况予以处理，直至取消入学资格。

学生可以休学:传染病，其他原因

学生休学至少一学期，一般以一年为限。学生复学后，休学之前已记入成绩档案的考核成绩继续有效，并作为学籍处理依据.

学生复学按下列规定办理:

（一）学生因伤病休学申请复学时，须持有二级甲等以上医院诊断书，证明身体健康，并经学校指定医院复查合格，方可复学;

（二）学生休学期满后应于学期的注册期内持有关证明，经教务处核准后编入原专业相应班级选课学习;

学生以下情况被退学:

（一）学生在读期间，3 次出现在一学期中取得的课程学分不足 10 学分（不含重修和补考学分；毕业学期除外；第一次提出警告，第二次提出退学警告，由教务处公布名单，院系负责通知学生家长）;

（二）休学、保留学籍期满，在规定期限内不办理复学手续;

（三）休学累计满二年，经复查不合格;

（四）因伤病需要休学，经学校动员后仍不办理休学手续；（

（五）经学校指定医院确诊患有疾病，或意外伤残无法继续在校学习;

（六）未请假离校连续 2 周末参加学校规定的教学活动;

（七）超过学校规定期限未注册而又无正当事由;

（八）本人要求退学。

学生在规定的学习年限（4 年制 3～6 年，5 年制 4～7 年）内修完本专业培养计划规定的全部教学环节，取得注册专业规定的毕业学分，准予毕业，发给毕业证书。

2、**分析出状态**

(1)入学状态

(2)正常学习状态

(3)休学状态

(4)复学状态

(5)退学状态

(6)毕业状态

3、**找出主要状态之间的迁移**

（1）入学状态至正常学习状态：当复查合格时，进行注册，进入正常学习状态

（2）正常学习状态至休学状态：当传染病，其他原因时，进行休学，进入休学状态

（3）休学状态至复学状态：休学满一年后且身体健康时，进行复学，进入复学状态

（4）复学状态至正常学习状态：当复学教务核准时，进行注册，进入正常学习状态

（4）正常学习状态至退学状态：当自愿等情况时，进行退学，进入退学状态

（5）正常学习状态至毕业状态：当修满学分时，进行离校手续办理，进入毕业状态

4、**画图**