1、软件是指：

**指令的集合**（计算机程序），通过执行这些指令可以满足预期的特征、功能和性能需求；

**数据结构**，使得程序可以合理利用信息；

**软件描述信息（文档）**，它以硬拷贝和虚拟形式存在，用来描述程序操作和使用。

软件 = 程序+数据+文档 程序 = 算法+数据结构

2、软件开发过程中为什么要编写文档？

1、便于对软件开发的管理和维护

2、便于各种人员的交流

3、软件的特点：

1、复杂性

软件实体比任何由人类创造的其他实体更复杂

难以理解，使维护过程变得十分困难

2、一致性

软件必须适应已有的技术和系统，随接口的不同而改变

复杂性来自保持与其他接口的一致

3、可变性

软件经常会遭受到持续的变更压力

软件很容易修改

修改会带来副作用，造成故障率的升高

4、不可见性

软件是客观世界空间和计算机空间中的一种逻辑实体，不具有物理的形体特征。

“需要做什么”

4、**软件危机**是指在计算机软件的**开发和维护**过程中所遇到的一系列严重问题。

5、**软件过程的目的**是为各种人员提供一个**公共的框架**，以便用相同的语言进行交流。

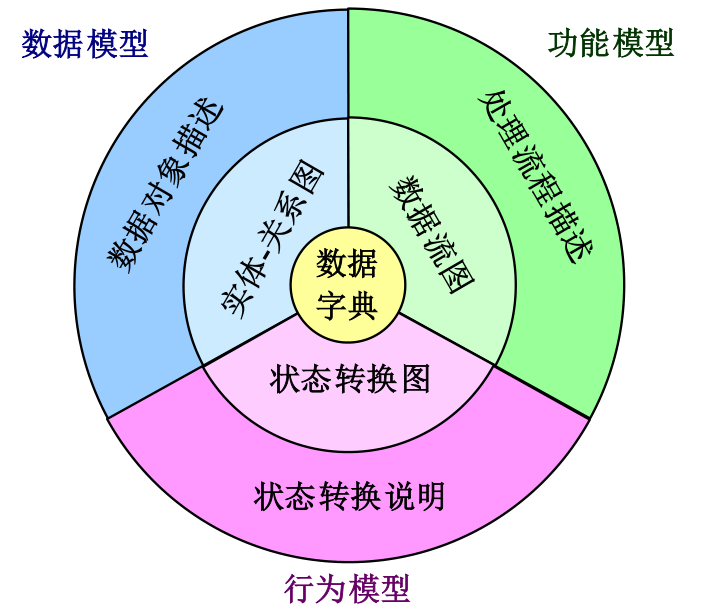
6、**测试**是使用测试用例来执行系统，检验系统的正确性。

7、**软件过程模型**是一个软件过程的抽象表示（路线图）。

8、**用例图**是对交互的一种简单概述，可以使用文本、表格或者顺序图对其补充说明。

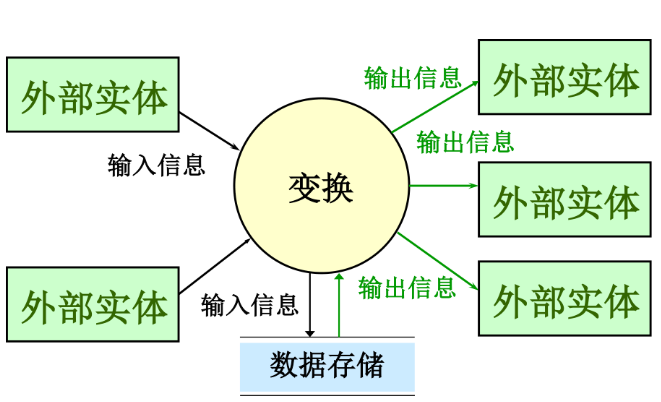
9、**验证需求的正确性：**一致性（Consistency）、完整性（Completeness）、现实性（Realism）、有效性（Validity）。

10、**结构化分析（Structured Analysis, SA）**用**抽象模型**的概念，按照软件内部**数据传递、变换**的关系，**自顶向下逐层功能分解**的系统分析方法来定义系统的需求。

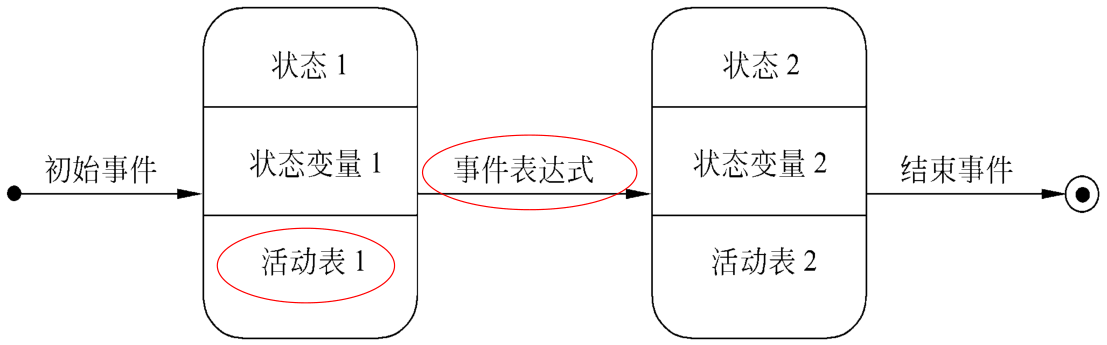


11、**数据模型**包含三种相互关联的信息：**数据对象**、**描述数据对象的属性**及**数据对象彼此间相互连接的关系**。最常用的表示概念性数据模型的方法，是实体联系方法（Entity-Relationship Approach）。ER图（Entity-Relationship Diagram）描述现实世界中的实体，而不涉及这些实体在系统中的实现方法。

12、功能模型（DFD）：



13、行为模型（状态转换图）：



常用的3种标准事件：

①entry事件指定进入该状态的动作；

②exit事件指定退出该状态的动作；

③do事件则指定在该状态下的动作。

14、数据字典（DD）：是对所有与系统相关的数据元素的一个有组织的列表，以及**精确的、严格的**定义。

15、OOA(Object-Oriented Analysis)面向对象的分析、OOD(Object-Oriented Design)面向对象的设计、OOP (Object-Oriented Program)面向对象的程序设计

消息传递是对象与外部世界关联的唯一途径。

16、基于场景的模型：用例图、顺序图

类模型：类图、协作图

行为模型：状态图、顺序图

这三种模型从不同侧面描述了对系统的需求。在面向对象的分析（OOA）阶段，这三种模型是必不可少的。

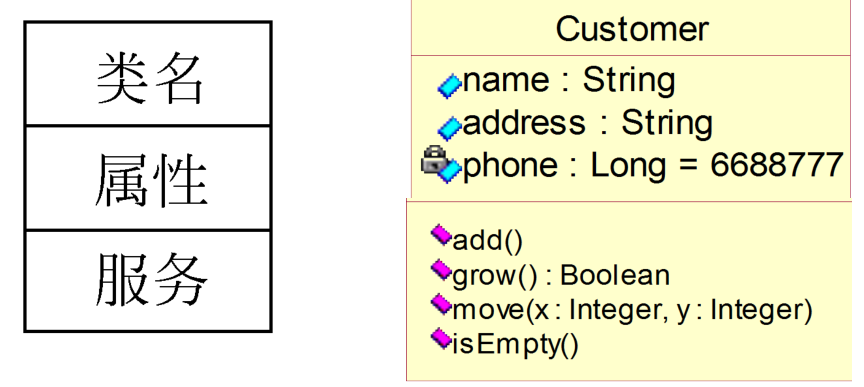
17、UML(Unified Modeling Language) 是一个通用的标准建模语言，可以对任何具有静态结构和动态行为的系统进行可视化建模。UML不是一个开发过程，也不是一个方法，但允许任何一种开发过程和面向对象方法使用它。

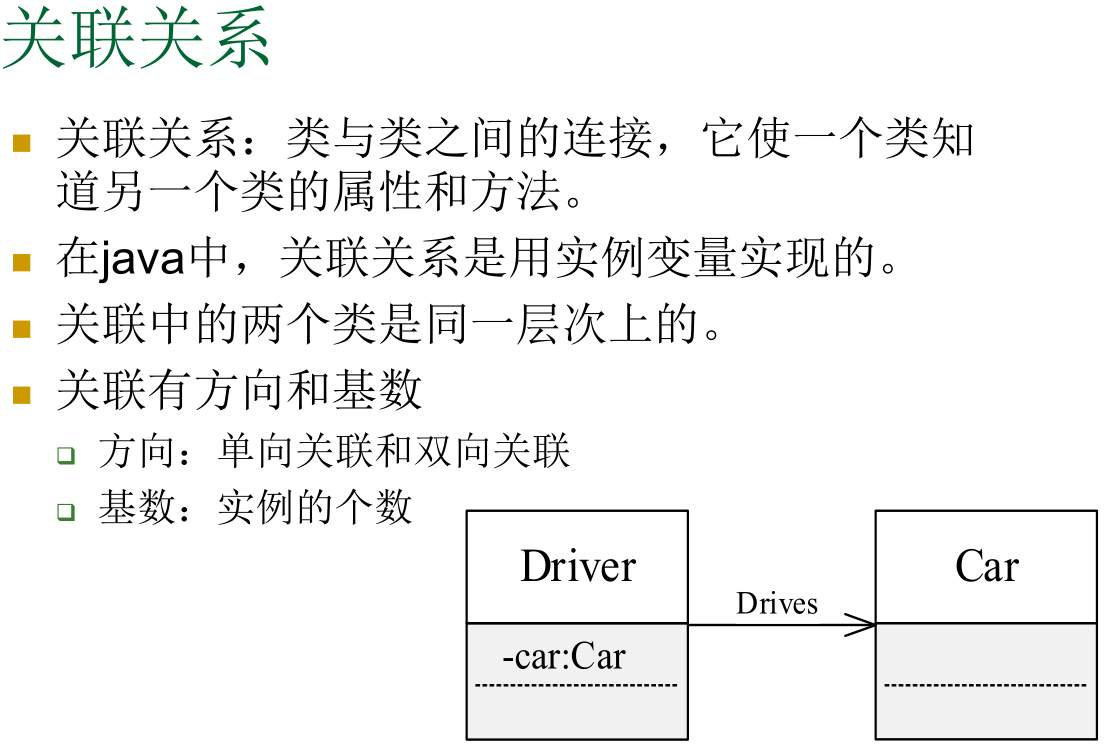
UML提供的**用例图**是进行需求分析和建立功能模型的强有力工具。

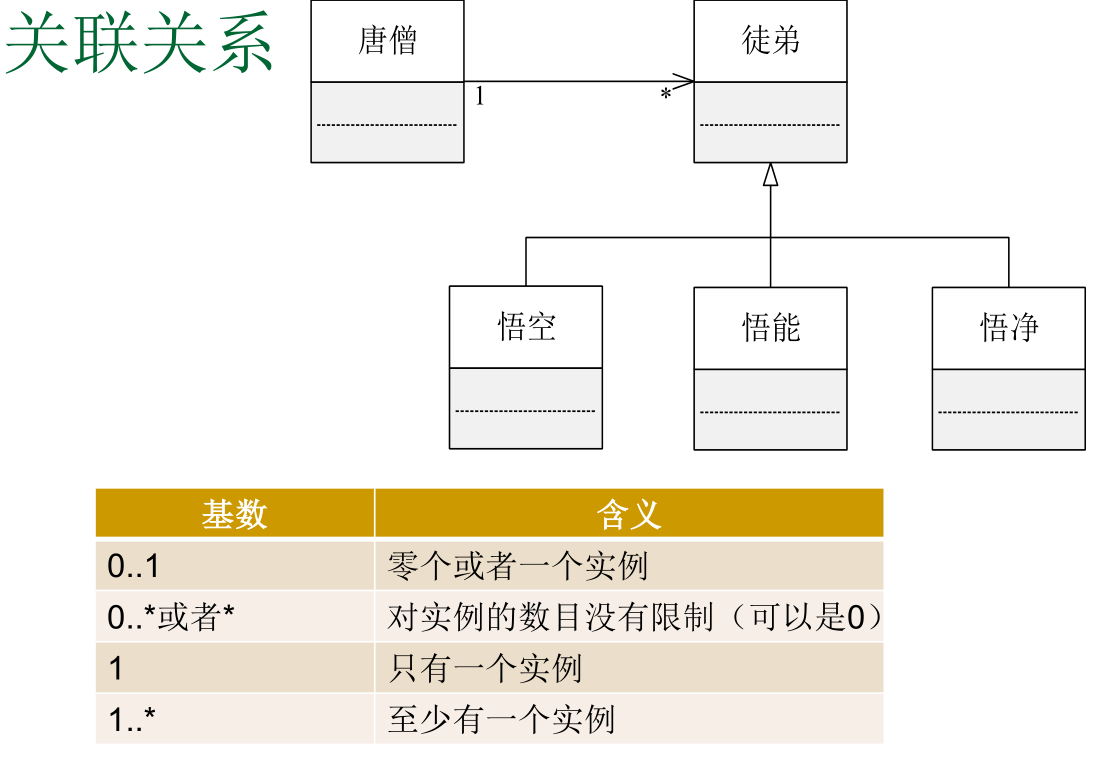
·用例模型描述的是外部行为者（actor）所理解的系统功能。

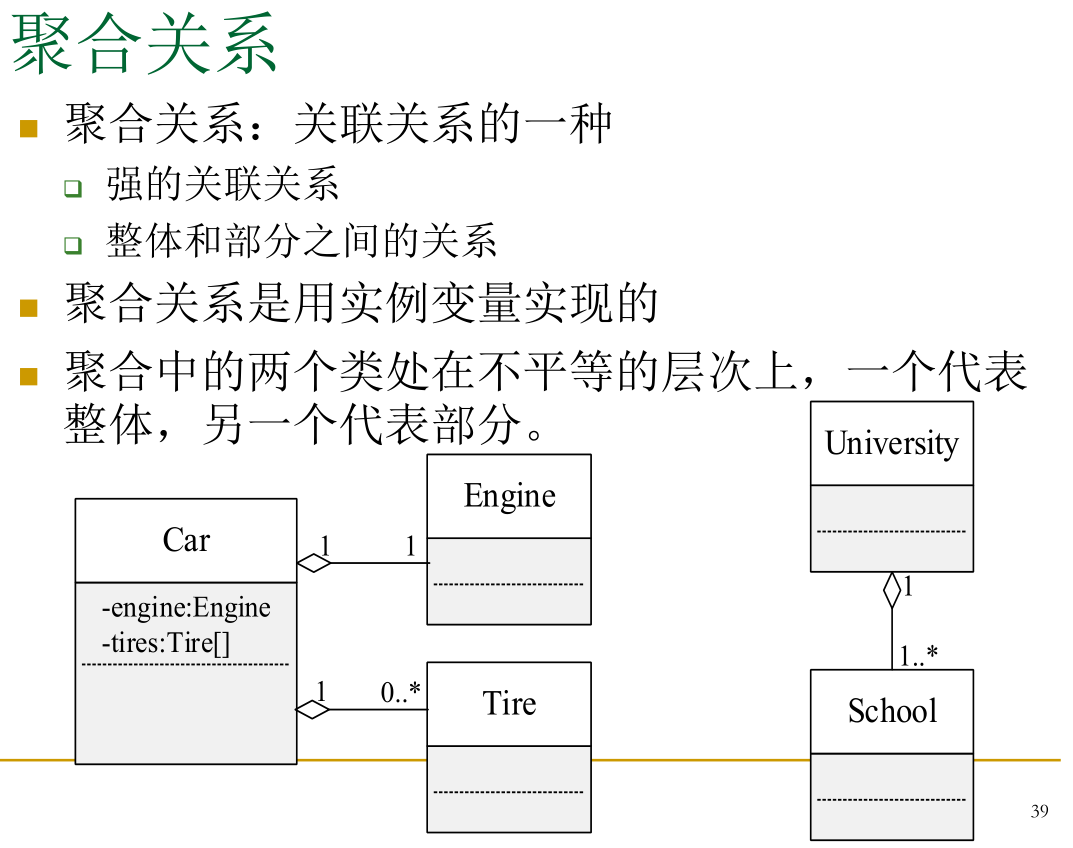
·用例模型的建立是系统开发者和用户反复讨论的结果。

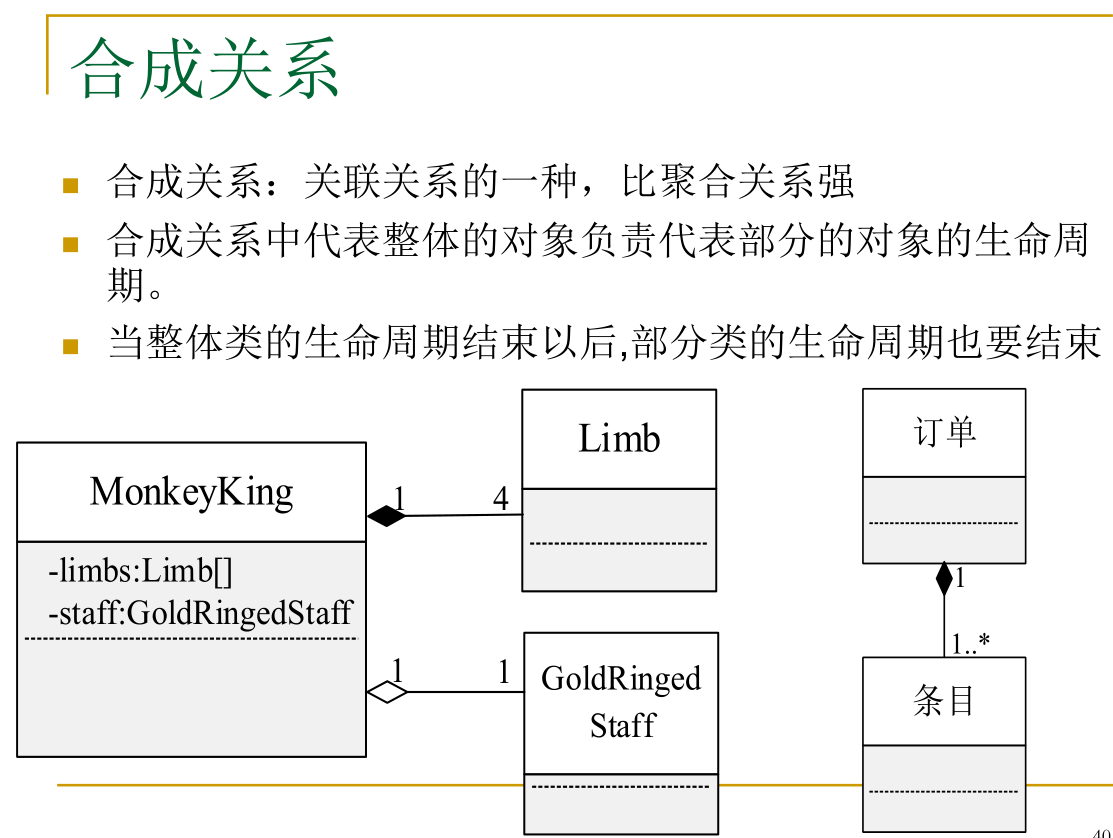
一幅用例图包含的模型元素有**系统**、**行为者**、**用例**及**用例之间的关系**。

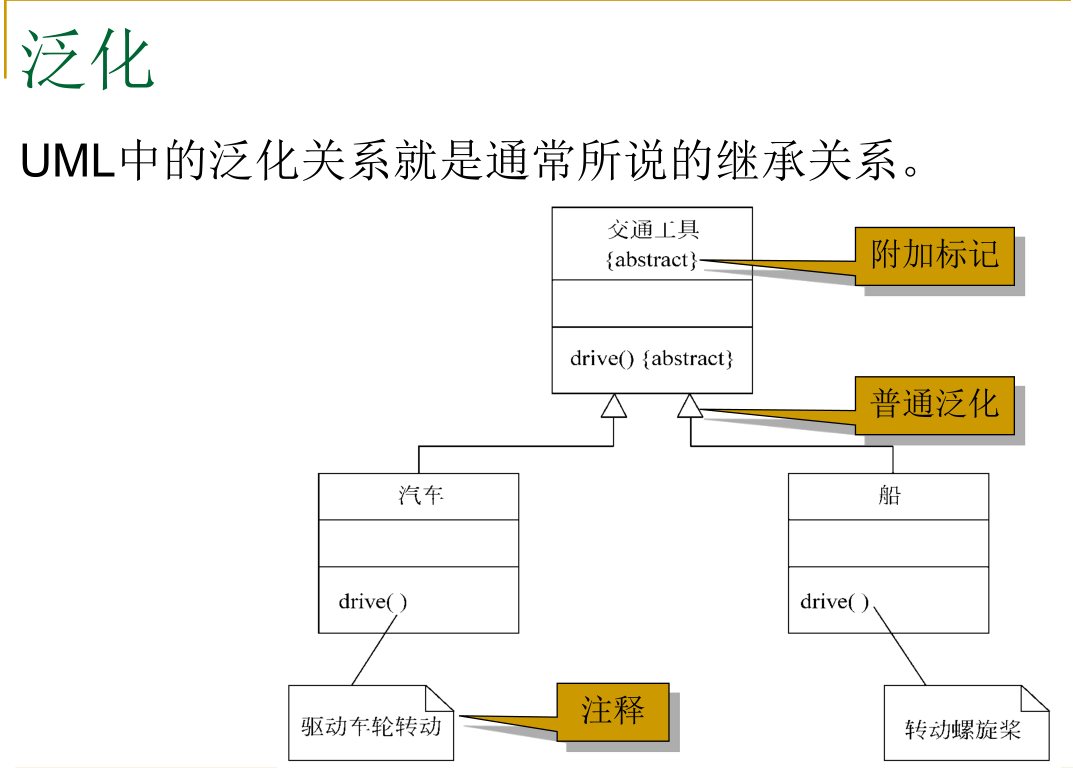


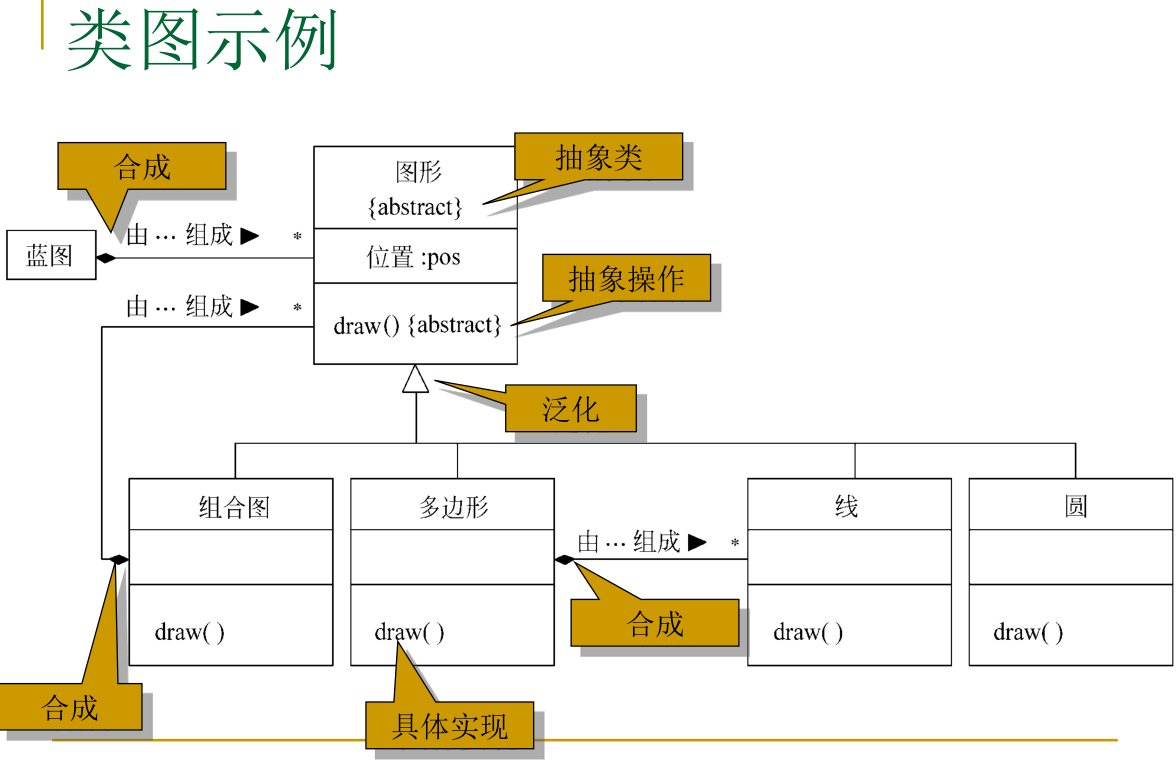


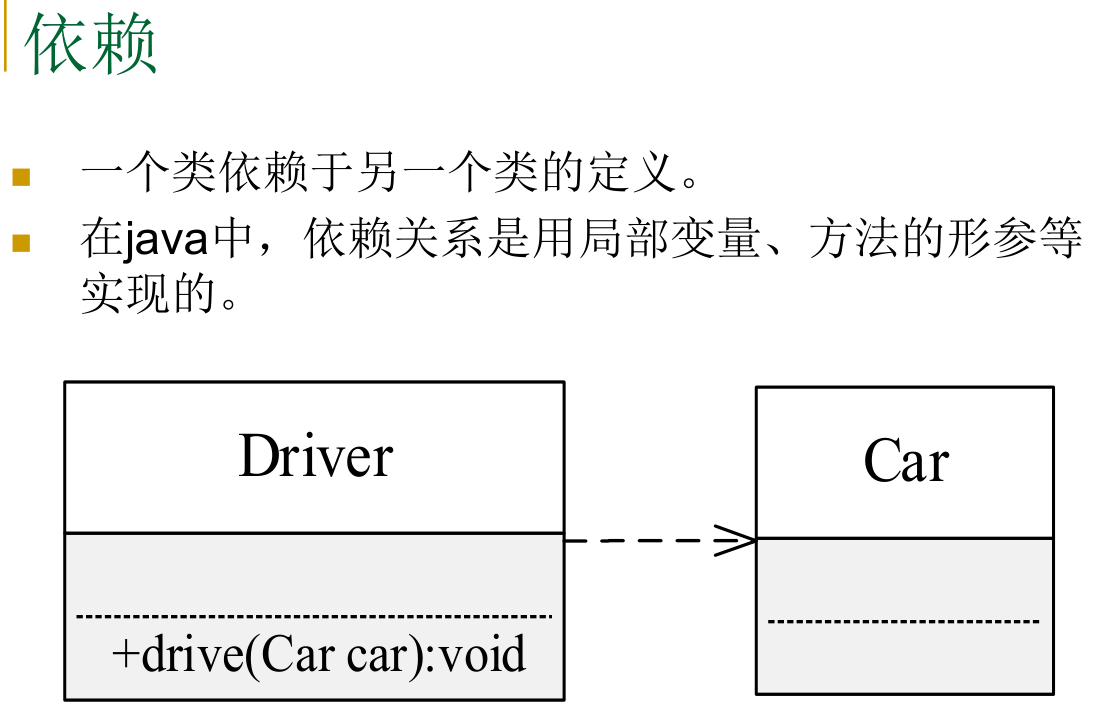












18、从技术的角度，面向对象方法将软件设计划分为**数据/类设计、体系结构设计、接口设计和构件设计**4部分。

19、从工程管理的角度，可以将软件设计分为**总体设计**和**详细设计**。

**总体设计：**将**软件需求**转化为**数据结构和软件的系统结构**。

**详细设计：**通过对结构表示进行细化，得到软件的详细的数据结构和算法。

20、设计原理：模块化(Modularity)、抽象(Abstraction)、逐步求精(Stepwise Refinement)、信息隐藏(Information Hiding)、模块独立。

模块独立程度的两个定性标准度量：耦合(Coupling)、内聚(Cohesion)

应该采取下述设计原则：

尽量使用数据耦合，少用控制耦合和特征耦合，限制公共环境耦合的范围，完全不用内容耦合。

模块设计追求**高内聚、低耦合、可重用**。

21、开闭原则：在设计一个构件时，应当让这个构件在不被修改（代码层）的前提下被扩展，即在不修改源代码的情况下改变这个模块的行为。

22、设计模式（Design pattern）是一套被反复使用、多数人知晓的、经过分类编目的、代码设计经验的总结。使用设计模式是为了可重用代码、让代码更容易被他人理解、保证代码可靠性。整个设计模式贯穿一个原理：**面对接口编程，而不是面对实现**。目标原则是：**降低耦合, 增强灵活性**。

23、黑盒测试(功能测试)、白盒测试(结构测试)、

* 环形复杂度*V*(G)计算方法
  + *V*(G)=流图中的区域数
  + *V*(G)=*E*-*N*+2，其中*E*是流图中的边数，*N*是结点数
  + *V*(G)=*P*+1，其中*P*是流图中判定结点的数目

24、