程序设计模式作业一

2020年2月15日　13:36


第一次作业，UML基础

作业要求：

1、 什么是用例？它有什么作用？

2、 一个用例图包括哪些元素？这些元素间的关系有哪些类型？试解释这些关系，并举例说明（画出用例图）。

3、 什么类？UML类图包括哪些部分？

4、 类之间有哪些关系？请分别举例说明。

- Q1：什么是用例？它有什么作用？（PPT chapter01 P27）

A1：

a. 用例是UML中重要的概念，是软件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。每个用例提供一个或多个场景，该场景说明系统是如何和最终用户或其他系统互动，从而获得明确业务目标。在UML文档中，用例的定义是：在不展现一个系统或子系统内部结构的情况下，对系统或子系统的某个连贯的功能单元的定义和描述。

b. 用例的用途在于帮助开发团队一种可视化的方式理解系统的功能需求。从用户角度描述系统的静态使用情况，用于建立需求模型。所使用的用例图主要描述角色以及角色与用例的连接关系。说明的是谁要使用系统，以及他们使用该系统可以做什么。这样通过模型元素以及元素之间的关系展示了一个外部用户能够观察到的系统功能模型图。

- Q2：一个用例图包括什么元素？这些元素间的关系有某种类型？试解释这些关系，并说明说明（画出用例图）。（PPT chapter01 P29）

A2：

a. 用例图中包含的元素有：参与者、用例、子系统、关系。

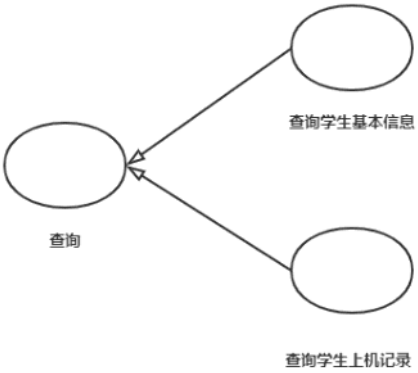
b. 用例图中包含的元素间的关系有：泛化、包含、扩展、关联。

c. 下面画出用例图对以上关系进行相关说明：

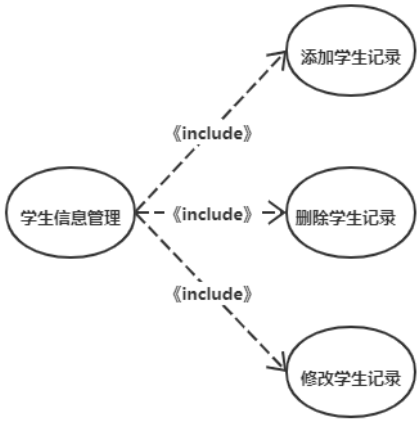
i. 泛化：泛化就是通常理解的继承关系。

一个用例可以被特别列举为一个或多个子用例，这被称为用例泛化，泛化关系在类间也有。子用例从父用例处继承行为和属性，还可以添加行为或覆盖、改变已继承的行为。

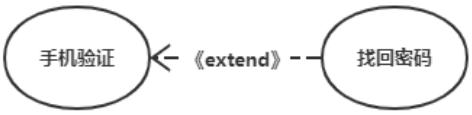
E.G. 在机房收费系统中可以这样应用



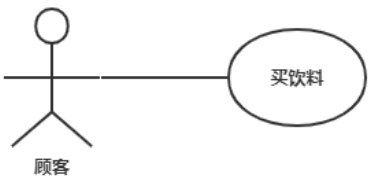
ii. 包含：包含关系用来把一个较为复杂用例所表示的功能分解成较小的步骤。包含关系对典型的应用就是复用，也就是情景。
E.G. 学生管理系统



iii. 扩展：是指用例功能的延伸，相当于为基础用例提供一个附加功能
E.G. 用户身份验证



iv. 关联：表示参与者与用例之间的关系，任何一方都可以发送或接收消息
E.G. 自动售货机



- Q3：什么是类？UML类图包括哪些部分？（PPT chapter01 P34）
  A3：
  a. 类封装了数据和行为，是面向对象的重要组成部分，他是具有相同属 性、操作、关系的对象集合的总称。
  在系统中每个类具有一定的职责，职责值得是类所担任的任务，即类要完成什么样的功能，承担什么样的义务。一个类可以有很多种职责，设计得好的类一般只有一种职责，在定义类的时候，可将类的职责分解为类的属性和操作。

类的属性即类的数据职责，累的操作即类的行为职责

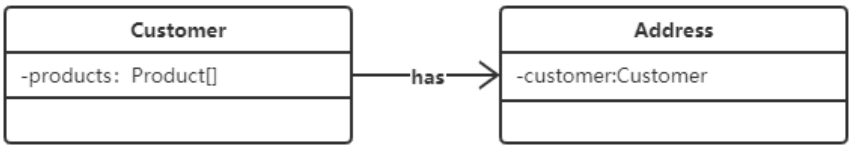
b. UML类图中一般是由三个部分组成：
   i. 类名：类的名字，类名是一个字符串
   ii. 属性：属性是类的性质，即类的成员变量。类可以有任意多个属性，也可以没有属性
   iii. 操作：操作是类的任意一个实例对象都可以使用的行为，操作是类成员的方法

- Q4：类之间有哪些关系？请分别举例说明。（PPT chapter01 P53）

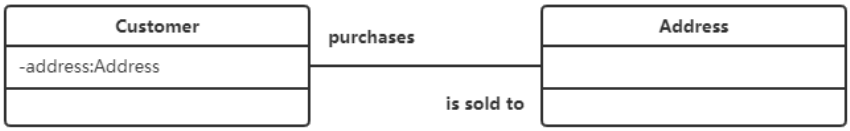
  A4：

  a. 关联关系：是类与类之间最常见的一种关系，它是一种结构化关系，用于表示一类对象与另一类对象之间有关系。如果B类作为A类的一个属性存在，称A关联于B。有三种类型：单向关联，双向关联，自关联

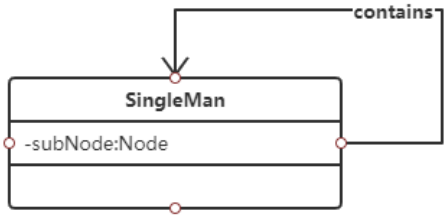
     E.G.



单向关联

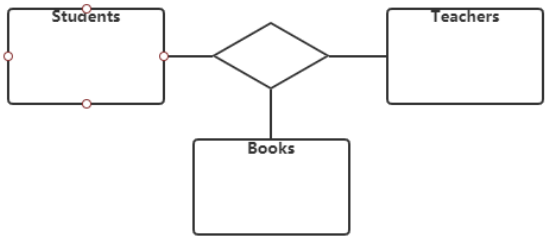


双向关联



自关联

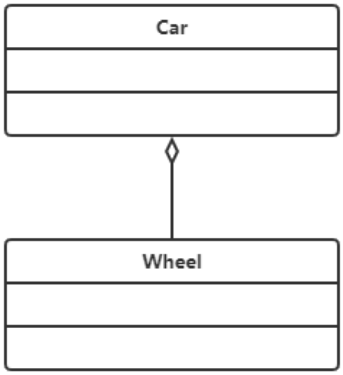
扩展：多维关联



多维关联

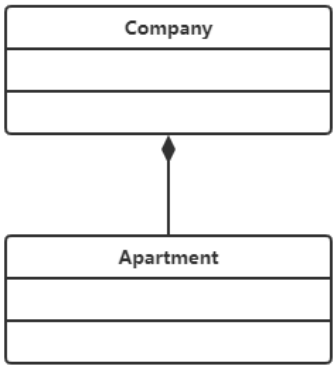
  b. 聚合关系：是一种关联关系，是弱关联关系。二者是一种Has-a的关系，是一种整体与部分的关键。二者可以单独存在

     E.G.



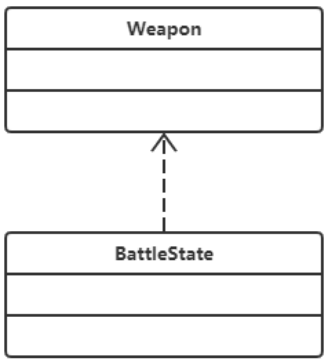
聚合

c. 组合关系：是强关联关系，二者是Contains-a的关系，也是一种整体与部分的关系。其中B是A的部件，不能单独存在
E.G.



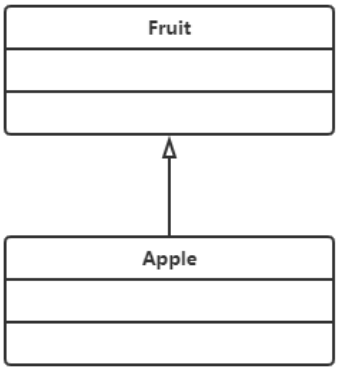
组合

d. 依赖关系：若B类作为参数被A类在一个方法中使用，则A依赖B
E.G.



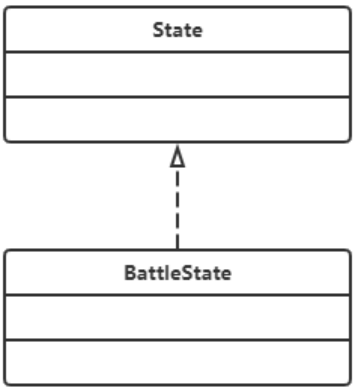
依赖

e. 泛化关系（继承）：描述父类与子类的关系，类B是类A的子类
E.G.



泛化

f. 实现关系：类B实现了接口A的功能，类B是接口A的实现
E.G.



实现

- 各种关系之间的强弱顺序：
  - 泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖