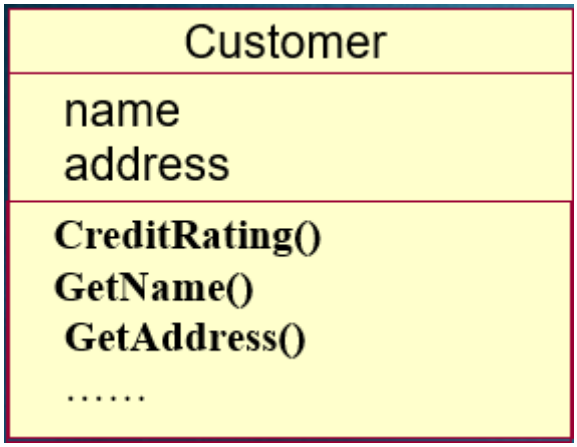
UML notes 04 Class Diagram

collected by wxb

1 Intro of Class

- class: 拥有相同属性、操作、关系的对象的集合
- 分类
  - entity: class 实体类
- control class 控制类
  - boundary class 边界类
- 类的表示：name + attributes + operations

  属性的表示语法：`[visibility] name [multiplicity] [:type] [= initial value] [{property}]`



- visibility

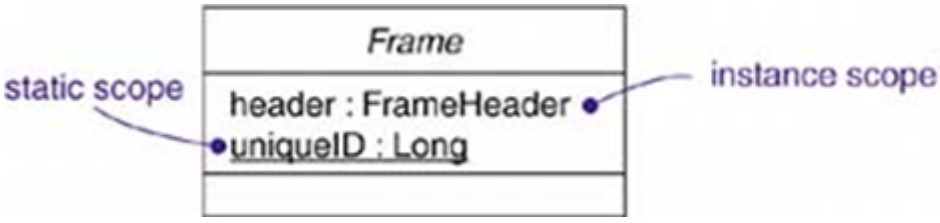
  +: public

  #: protected

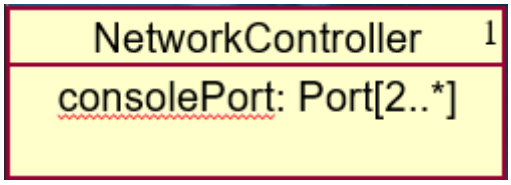
  -: private

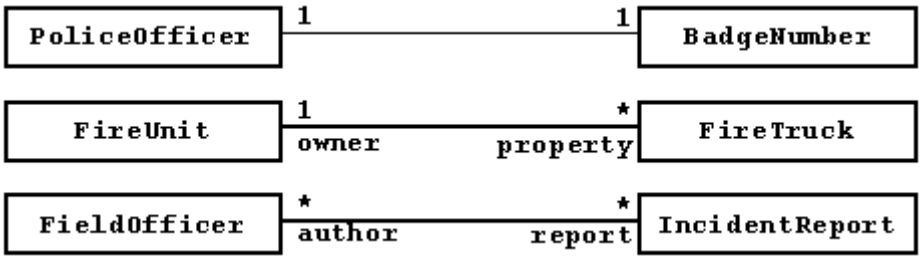
  ~: package

- scope



- multiplicity: 表示一个类可能有的实例数量





- polymorphic: 多态，子类会覆盖父类中的同名操作
- name: string

  simple name: 单独的名字

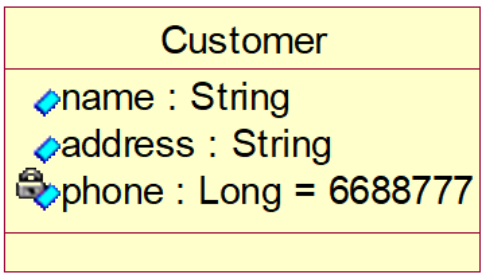
  qualified name: 加上package作为前缀，如 java::awt::Rectangle

  通常首字母大写
- attribute

  数量无限制，0到无限

  `[visibility] name [: type] [multiplicity] [=init value]`

  `\+ name : String [0...2] = "Fine"`



- operation

  数量无限制，0到无限

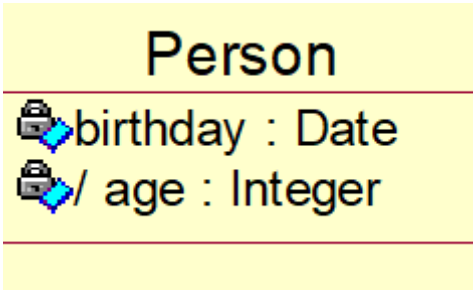
  格式 `[visibility] name [(param list)] [: return type] [property-string {, }]`

  如 `+ getNameById (id: int): String {guarded}`

  参数格式：`[direction] name :type [=default value]` 如 `in name: String ="Fine"`
- derived attribution 衍生属性

  值可以由模型中的其他属性计算得到

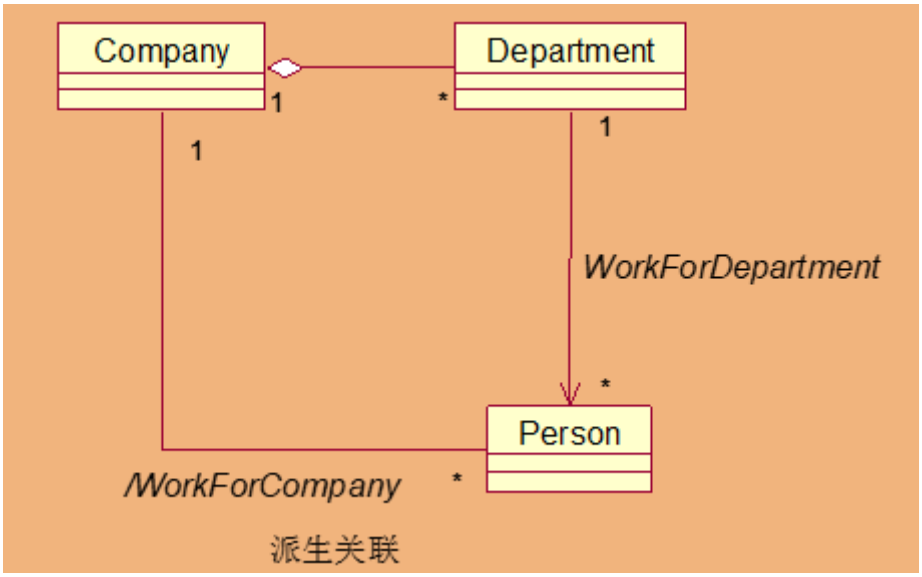
  表示：要在 name 前加上 /



- derived association 衍生关系

  由模型中已有的关系衍生得到

  表示：要在关系描述前加上 /



衍生属性和衍生关系不会生成代码

2 类中的关系

2.1 Inheritance 继承

generalization

- 子类会继承父类的属性和方法
- 子类可以增加属性、行为，重写父类的行为
- 父类可以用的地方子类都可以用

2.2 dependency 依赖

using

- 一个类使用另一个类，因此被使用的类的变更会影响到使用类，反之则不会
- 表示方法：点线，从依赖方指向被依赖方
- model-level 关系，不是run-time关系

realizing

- 接口由提供其实现的一组分类器实现。
- 协作实例包含实现用例指定的行为所需的对象和消息

2.3 association 关联

- 用来表示对象之间的关系
- 包括：关系名称 + 关系角色 + 数量关系
- 使用 association class 进行关联
- 使用 qualifier，可以区分不同的对象，比如一个三角形的三个顶点
- navigation, 用箭头指
- visibility
- constraints

Constraint Name	Description
ordered	Specifies that the set of **objects** at one end of an association are in an explicit order
set	The objects are unique with no duplicates
bag	The objects are nonunique, may be duplicates
ordered set	The objects are unique and ordered
list or sequence	The objects are ordered, may be duplicates
readonly	A link, once added from an object on the opposite end of the association, may not be modified or deleted. The default in the absence of this constraint is unlimited changeability

aggregation: has-a

- 表示包含关系，the aggregate 包含 the parts，一个part可以从属于多个 aggregate，并且可以独立存在
- 一个聚合不能涉及两个以上的类
- 不需要有相同的生命周期

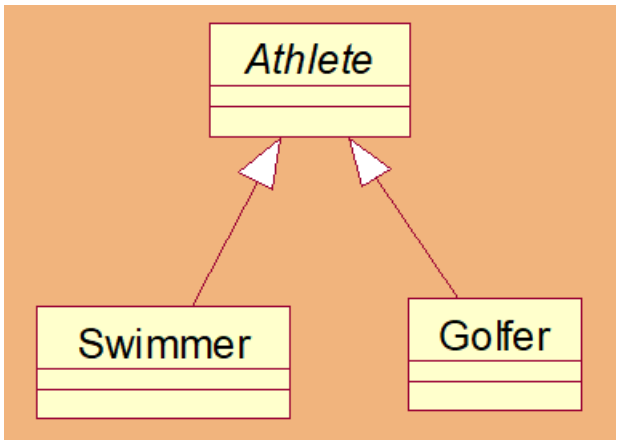
composition

- 生命周期需要等长

2.4 Abstract class & interface

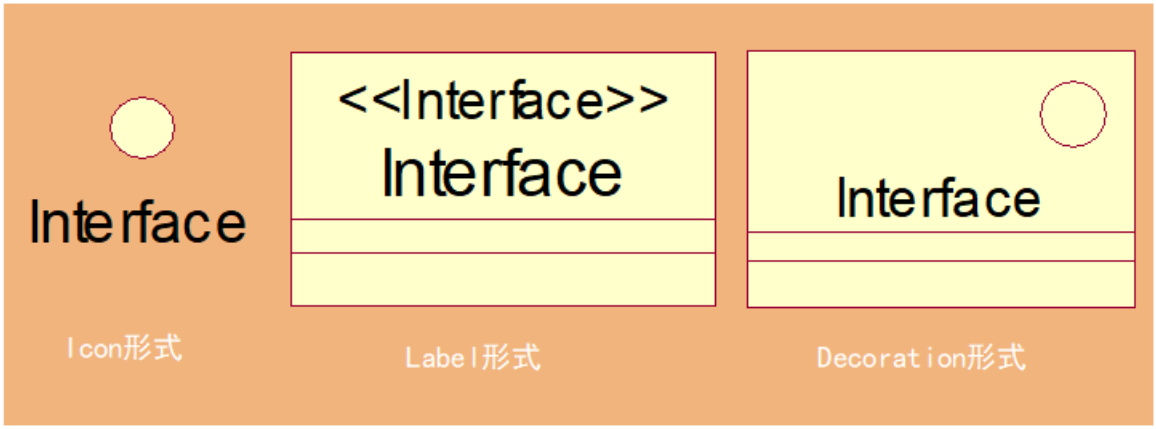
abstract class 抽象类

- 不能直接实例化
- 通常只有声明，没有实现
- 表示方法：斜体



interface 接口

- 类的一种原型
- 一组操作的集合，用来实现一个类或者组件要求的服务
- 接口中没有attribute，只有operation的声明



3 Class Diagram

- **stati**c view of system
- 展示了 packages, classes, interfaces, collaborations, dependency, generalization, relationship
- **3 level: conceptual/domain => specification => implementation**

3.1 CRC analysis method

Class, Responsibility, Collaboration

CAR Card

- name
- responsibility

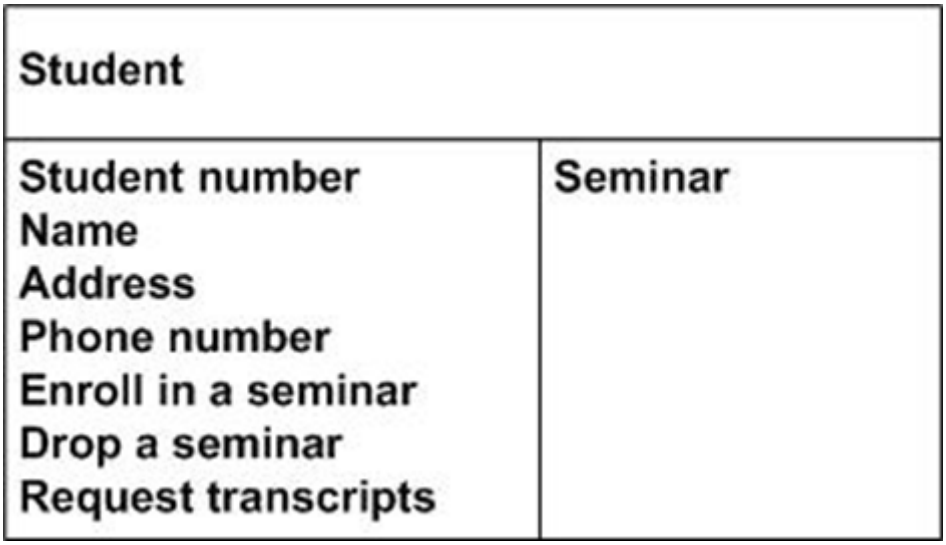
  这个类知道/有的，或者做的

  > 学生知道自己的名字、地址、电话号码，并且会选课、退课

- collaborators

  类需要交互或协作的对象

Class Name:	
Superclasses:	
Subclasses:	
Responsibilities:	Collaborators



3.2 Design principles of OO modeling

- **OCP: Open/Closed Principle**

  开放拓展，关闭修改

- **LSP: Liskov Substitution Principle**

  子类要能够继承父类的所有属性和方法

- **DIP: Dependency Inversion Principle**

  要保证依赖顺序不颠倒，从而保证底层模型可被重用

  High-level module 不能依赖于 low-level module，两者都要依赖于 abstractions

- **ISP: Interface Segregation Principle**

  用几个special的接口会比用一个general的接口要好

  接口不需要提供太多功能，只要提供一个service function就好

4 Object Diagram

- 展示了模型化系统特定时间 complete / partial 的视图
- 表示实例对象的属性以及之间的关系