# UML

## （1） 是什么

参考：

1】 <https://en.wikipedia.org/wiki/Unified_Modeling_Language>

2】 <https://time.geekbang.org/column/article/161575>

## （2） 关系及符号

### 泛化：Generalization/继承：Inheritance

参考：

1. <https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>
2. <https://www.jianshu.com/p/0cefd8137681>
3. <https://www.jianshu.com/p/daae959e9134>
4. <https://en.wikipedia.org/wiki/Class_diagram#Generalization/Inheritance>

#### 是什么

<https://en.wikipedia.org/wiki/Class_diagram#Generalization/Inheritance>

##### 定义：

It indicates that one of the two related classes (the ***subclass***) is considered to be a specialized form of the other (the ***super type***) and the superclass is considered a **Generalization** of the subclass. In practice：事实上, this means that any instance of the subtype is also an instance of the superclass.

An exemplary tree of generalizations of this form is found in [biological classification](https://en.wikipedia.org/wiki/Biological_classification): [humans](https://en.wikipedia.org/wiki/Human) are a subclass of [simian](https://en.wikipedia.org/wiki/Simian), which is a subclass of [mammal](https://en.wikipedia.org/wiki/Mammal), and so on.  The relationship is most easily understood by the phrase 'an A is a B' (a human is a mammal, a mammal is an animal).：在生物分类中可以找到一个典型的这种形式的概括树:人类是类人猿的一个子类，而类人猿又是哺乳动物的一个子类，等等。这种关系最容易理解的短语是“A是B”(人是哺乳动物，哺乳动物是动物)。

##### 别名：

The generalization relationship is also known as the [*inheritance*](https://en.wikipedia.org/wiki/Inheritance_(computer_science)) or *"is a"* relationship.

##### superclass和subtype：

The [*superclass*](https://en.wikipedia.org/wiki/Superclass_(computer_science)) (base class) in the generalization relationship is also known as the *"parent"*, *superclass*, *base class*, or *base type*.

The [*subtype*](https://en.wikipedia.org/wiki/Subtyping) in the specialization relationship is also known as the *"child"*, *subclass*, *derived class*, *derived type*, *inheriting class*, or *inheriting type*.

##### 和生物学的亲子关系区分：

Note that this relationship bears no resemblance to the biological parent–child relationship: the use of these terms is extremely common, but can be misleading.：请注意，这种关系与生物学上的亲子关系没有相似之处：这些术语的使用非常普遍，但可能会产生误导。

A is a type of B. For example, "an oak is a type of tree", "an automobile is a type of vehicle"：A是B的一种。例如，“橡树是一种树”，“汽车是一种交通工具”

##### 表示方式：

Generalization can only be shown on class diagrams and on [use case diagrams](https://en.wikipedia.org/wiki/Use_case_diagram).

#### 不同语言表示

在java中用extends表示该关系。

#### 符号表示

空心三角箭头+实线

例1：<https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>

表格

描述已自动生成

例2：

<https://en.wikipedia.org/wiki/Class_diagram#Generalization/Inheritance>

[图示

描述已自动生成](https://en.wikipedia.org/wiki/File:KP-UML-Generalization-20060325.svg)

Class diagram showing generalization between the superclass *Person* and the two subclasses *Student* and *Professor*

### 【2】 实现：Realization/Implementation

参考：

1. <https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>
2. <https://www.jianshu.com/p/0cefd8137681>
3. <https://www.jianshu.com/p/daae959e9134>
4. <https://en.wikipedia.org/wiki/Class_diagram#Realization/Implementation>

#### 是什么

<https://en.wikipedia.org/wiki/Class_diagram#Realization/Implementation>

##### 定义：

In UML modelling, a realization relationship is a relationship between two model elements, in which one model element (the client) realizes (implements or executes) the behavior that the other model element (the supplier) specifies.

#### 不同语言表现

在java中用implements表示该关系。

#### 符号表示

空心三角箭头+虚线

表格

中度可信度描述已自动生成

### 【3】 依赖：Dependency

参考：

1. <https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>
2. <https://www.jianshu.com/p/0cefd8137681>
3. <https://www.jianshu.com/p/daae959e9134>
4. <https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-dependency-relationships>
5. <https://en.wikipedia.org/wiki/Class_diagram>
6. <https://blog.csdn.net/K346K346/article/details/59582926>
7. <https://www.cnblogs.com/liuzhang/archive/2013/03/17/2964095.html>
8. <https://design-patterns.readthedocs.io/zh_CN/latest/read_uml.html>
9. <https://www.jianshu.com/p/ee0c26bcbf3d>
10. <https://www.cnblogs.com/xrq730/p/5533019.html>
11. <https://blog.csdn.net/lk1822791193/article/details/82851321>
12. <https://zhuanlan.zhihu.com/p/350793664>

#### 是什么

1. 是一种使用关系，表示类之间的调用关系。
2. 依赖总是单向的。尽量避免双向依赖
3. 依赖关系是五种关系中耦合最小的一种关系。类A要完成某个功能调用了类B，则类A依赖类B。
4. 可以简单的理解，就是一个类A使用到了另一个类B，而这种使用关系是具有临时性的、非常弱的，但是B类的变化会影响到A；
5. 他描述一个对象在运行期间会用到另一个对象的关系；

与关联关系不同的是，它是一种临时性的关系，通常在运行期间产生；并且随着运行时的变化，依赖关系也可能发生变化；

依赖关系除了临时知道对方外，还有“使用”对方的方法和属性；

1. 依赖关系强调的是\*\*使用上的关系 \*\*。
2. 它表示类与类之间的连接，表示一个类依赖于另外一个类，方向是单向的。（可以简单理解为：我吃饭用到筷子，所以在吃饭这件事上，我依赖于筷子，但是我不吃饭我就不依赖于它了！）

但是类B（筷子）的变化会影响类A（我）。（如果没有筷子，我吃饭就会受到影响）

#### 代码表现

一般为在**类A的某个方法（构造器，成员方法，类方法）中，返回值、形参、局部变量或对静态方法的调用**

1.A方法的参数是B（耦合）

2.A方法的参数是B的属性（紧耦合）

3.A方法中实例化了B

4.A方法的返回值是B

#### 符号表示

普通箭头+虚线

图形用户界面

中度可信度描述已自动生成

#### 实例

##### 1） 例1

In an e-commerce application, a Cart class depends on a Product class because the Cart class uses the Product class as a parameter for an add operation. In a class diagram, a dependency relationship points from the Cart class to the Product class. As the following figure illustrates, the Cart class is, therefore, the client, and the Product class is the supplier.

形状

描述已自动生成

The above relationship indicates that a change to the Product class might require a change to the Cart class.

##### 2） 例2

学生在学习生活中经常使用电脑，于是对电脑产生了依赖。以Student类和Computer类为例，用C++语言编码如下：

|  |
| --- |
| class Computer {  public:  static void start(){  cout<<"电脑正在启动"<<endl;  }  };  class Student {  public:  //返回值构成依赖  Computer& program();  //形参构成依赖  void program(Computer&);  void playGame() {  //局部变量构成依赖  Computer\* computer=new Computer;  ...  //静态方法调用构成依赖  Computer::star();  }  }; |

##### 3） 例3

比如某人要过河，需要借用一条船，此时人与船之间的关系就是依赖。人过河之后，与船的关系就解除了，因此是一种弱的连接。

##### 4） 例4

比如，驾驶员（Driver）开车，Driver类的drive()方法将车（Car）的对象作为一个参数传递，以便在drive()方法中能够调用car的move()方法，且驾驶员的drive()方法依赖车的move()方法，因此也可以说Driver依赖Car，Java代码为：

图形用户界面, 应用程序

描述已自动生成

##### 5） 例5

比如下图中的依赖关系，司机开车，要依赖车。如下图的UML类图中Driver 对Car的依赖就是通过函数参数的方式。

图示

中度可信度描述已自动生成

##### 5） 例6

依赖关系的例子有很多，比如：LocalDateTime的now方法根据时区ID创建LocalDateTime，这里说明LocalDateTime依赖ZoneId。

public static LocalDateTime now(ZoneId zone) {

return now(Clock.system(zone));

}

LocalDateTime依赖ZoneId的关系如下图所示：

图片包含 图形用户界面

描述已自动生成

### 【4】 关联：Association

参考：

1. <https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>
2. <https://www.jianshu.com/p/0cefd8137681>
3. <https://www.jianshu.com/p/daae959e9134>

#### 《1》 是什么

是一种拥有的关系，它使一个类知道另一个类的属性和方法。关联分为单

向关联和双向关联两种。

#### 《2》 不同语言表现

在 Java中，关联关系是通过使用**成员变量**来实现的。

#### 《3》 符号表示

普通箭头+实线。指向被拥有者。

有两种：单向和双向，

双向关联：可以有两个箭头或没有箭头（不推荐）；

单向关联：有一个箭头。

图片包含 图示

描述已自动生成

### 聚合：Aggregation

参考：

1. <https://www.jianshu.com/p/0cefd8137681>
2. <https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>
3. https://www.cnblogs.com/xrq730/p/5533019.html

#### 是什么

##### 概述

是整体与部分的关系，且部分可以离开整体而单独存在。如Car和Wheel是整体与部分的关系，Wheel离开Car仍然可以存在，并不随Car的创建而创建，销毁而销毁。

是一种弱拥有关系；

##### 和关联的关系

聚合关系是关联关系的一种。

* 关联关系所涉及的两个类处于同一个层次上；
* 聚合关系所涉及的两个类处于不平等的层次上(一个代表整体，一个代表部分)

关联与聚合仅仅从 Java 或 C++ 语法上是无法分辨的，必须考察所涉及的类之间的逻辑关系。

#### 不同语言表现

Java成员变量。**成员对象通常以构造方法、Setter方法的方式注入到整体对象之中**。

#### 符号表示

空心菱形+实线+普通箭头。菱形指向整体。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

图示

描述已自动生成

### 组合：Composition

参考：

1. <https://www.cnblogs.com/duanxz/archive/2012/06/13/2547801.html>
2. <https://www.jianshu.com/p/0cefd8137681>
3. <https://www.jianshu.com/p/daae959e9134>
4. <https://www.cnblogs.com/xrq730/p/5533019.html>
5. <https://zhuanlan.zhihu.com/p/93289356>
6. <https://www.huaweicloud.com/articles/4c7a5ce09a4b6e4d141a852ee40db7e1.html>

#### 《1》 是什么

##### 1） 概述

组合（Composition）关系也表示的是一种整体和部分的关系，但是在组合关系中整体对象可以控制成员对象的生命周期，一旦整体对象不存在，成员对象也不存在。

是整体与部分的关系，但部分不能离开整体而单独存在。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。

一种强的‘拥有’关系，体现了部分与整体的关系，它们拥有相同的生命周期。

##### 2） 和关联以及聚合的关系

1. 组合关系是关联关系的一种，是比聚合关系要强的关系，它要求普通的聚合关系中代表整体的对象负责代表部分的对象的生命周期。
2. 聚合是一种较弱形式的对象包含(一个对象包含另一个对象)关系。较强形式是组合(Composition).

聚合

很多大雁聚在一起变成雁群，也可以说雁群里有雁，所以是整体和个体之间的关系,即has-a的关系，此时整体与部分之间是可分离的，他们可以具有**各自**的生命周期。即雁群可以不存在了，但是孤雁仍然存在。

组合

一只鸟由两个翅膀组合而成，它体现的是contains-a关系，这种关系比聚合更强，也称为**强聚合**。他同样体现的是整体与部分间的关系，但此时整体和部分是不可分的，整体的生命周期结束也就意味着**部分**的生命周期结束。比如这只鸟死了，自然这两个翅膀也不存在了。

4.

文本

描述已自动生成

聚合与组合的对比：

（1）聚合关系没有组合紧密

学生不会因为班级的解散而无法存在，聚合关系的类具有不同的生命周期；而学生如果没有心脏将无法存活，组合关系的类具有相同的生命周期。

这个从构造函数可以看出。聚合类的构造函数中包含另一个类的实例作为参数，因为构造函数中传递另一个类的实例，因此学生可以脱离班级体独立存在。组合类的构造函数包含另一个类的实例化。因为在构造函数中进行实例化，因此两者紧密耦合在一起，同生同灭，学生不能脱离心脏而存在。

（2）信息的封装性不同。

在聚合关系中，客户端可以同时了解Classes类和Student类，因为他们是独立的。

在组合关系中，客户端只认识Student类，根本不知道Heart类的存在，因为心脏类被严密地封装在学生类中。

理解聚合与组合的区别，主要在于聚合的成员可独立，复合的成员必须依赖于整体才有意义。

#### 《2》 不同语言表现

Java成员变量。通常在**整体类的构造方法中直接实例化成员类**，这是因为组合关系的整体和部分是共生关系，如果通过外部注入，那么即使整体不存在，那么部分还是存在的，这就相当于变成了一种聚合关系了。

#### 《3》 符号表示

实心菱形+实线+普通箭头。箭头指向成员。

图形用户界面, 文本, 应用程序, 电子邮件

描述已自动生成

#### 《4》 实例

1） https://time.geekbang.org/column/article/219290?cid=100039001

图示

描述已自动生成

2）

<https://time.geekbang.org/column/article/204845>

文本

描述已自动生成

### 耦合度强弱

<https://blog.csdn.net/K346K346/article/details/59582926>

类关系涉及依赖、关联、聚合、组合和泛化这五种关系，耦合度依次递增。关于耦合度，可以简单地理解为当一个类发生变更时，对其他类造成的影响程度，影响越小则耦合度越弱，影响越大耦合度越强。

<https://www.cnblogs.com/liuzhang/archive/2013/03/17/2964095.html>

 类与类之间由弱到强关系是:    **没关系->依赖->关联->聚合->组合。**

<https://zhuanlan.zhihu.com/p/93289356>

UML类图几种关系的耦合度由强到弱，泛化 = 实现 > 组合 > 聚合 > 关联 > 依赖

## （3） UML图分类

### 【1】 UML类图：Class Diagram

#### 是什么

参考：<https://en.wikipedia.org/wiki/Class_diagram>

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering), a **class diagram** in the [Unified Modeling Language](https://en.wikipedia.org/wiki/Unified_Modeling_Language) (UML) is a type of static structure diagram that describes the structure of a system by showing the system's [classes](https://en.wikipedia.org/wiki/Class_(computer_science)), their attributes, operations (or methods), and the relationships among objects.

#### 类的表示法

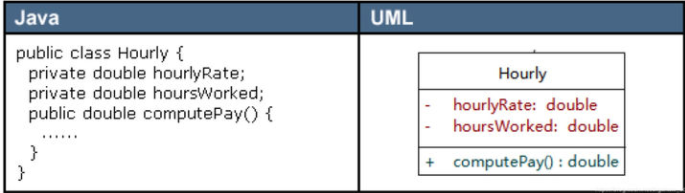
参考：

1. <https://en.wikipedia.org/wiki/Class_diagram>
2. <https://zhuanlan.zhihu.com/p/109655171>

##### 普通类

普通类在类图中用矩形框表示，矩形框分为三层：第一层是类名字。第二层是类的成员变量；第三层是类的方法。成员变量以及方法前的访问修饰符用符号来表示：

* “-”表示 private；
* 不带符号表示 default。
* “#”表示 protected；
* “+”表示 public；



##### 抽象类

抽象类在UML类图中同样用矩形框表示，但是抽象类的类名以及抽象方法的名字都用斜体字表示，如下图所示。

表格

描述已自动生成

##### 接口

接口在类图中也是用矩形框表示，但是与类的表示法不同的是，接口在类图中的第一层顶端用构造型 <<interface>>表示，下面是接口的名字，第二层是方法，如下图。

图形用户界面

中度可信度描述已自动生成

此外，接口还有另一种表示法，俗称棒棒糖表示法，如下。

图示

描述已自动生成这里唐老鸭实现讲人话接口。

##### 包

类和接口一般都出现在包中，UML类图中包的表示形式如下图所示。

表格

描述已自动生成

#### 关系及符号

见（2）

### 【2】 包图（并非正式的UML图）：Package Diagram

#### 《1》 是什么

##### 参1：

<https://blog.csdn.net/caozhangyingfei0109/article/details/8534191>

1、定义：包图是在 UML 中用类似于文件夹的符号表示的模型元素的组合

2、简要介绍：包图是一种维护和描述系统总体结构的模型的重要建模工具，通过对包中各个包以及包之间关系的描述，展现出系统的模块与模块之间的依赖关系。

包图的作用：包图可以描述需求，设计的高阶概况；包图通过合理规划自身功能反应系统的高层架构，在逻辑上将系统进行模块化分解；包图最终是组织源码的方式。

一个包图可以由任何一种ＵＭＬ图组成，通常是ＵＭＬ用例图或是ＵＭＬ类图。

包被描述成文件夹，可以用于ＵＭＬ任何一种的图上。

包图只是把某些类放在一个包中，因此可以看做是类图的一种。

##### 参2：

<https://blog.csdn.net/erlian1992/article/details/51193407>

包图是一种维护和描述系统总体结构的模型的重要建模工具，通过对图中各个包以及包之间关系的描述，展现出系统的模块与模块之间的依赖关系。

在UML的建模机制中，模型的组织是通过包（package）来实现的。包可以把所建立的各种模型（包括静态和动态）组织起来，形成各种功能或用途的模块，并可以控制包中元素的可见性和描述包之间的依赖关系。通过这种方式系统模型的实现者能够在高层把握系统的机构。

#### 《2》 实例

<https://blog.csdn.net/caozhangyingfei0109/article/details/8534191>

*机房收费系统*

图片包含 图示

描述已自动生成

## （3） UML工具

### 【1】 UMLet