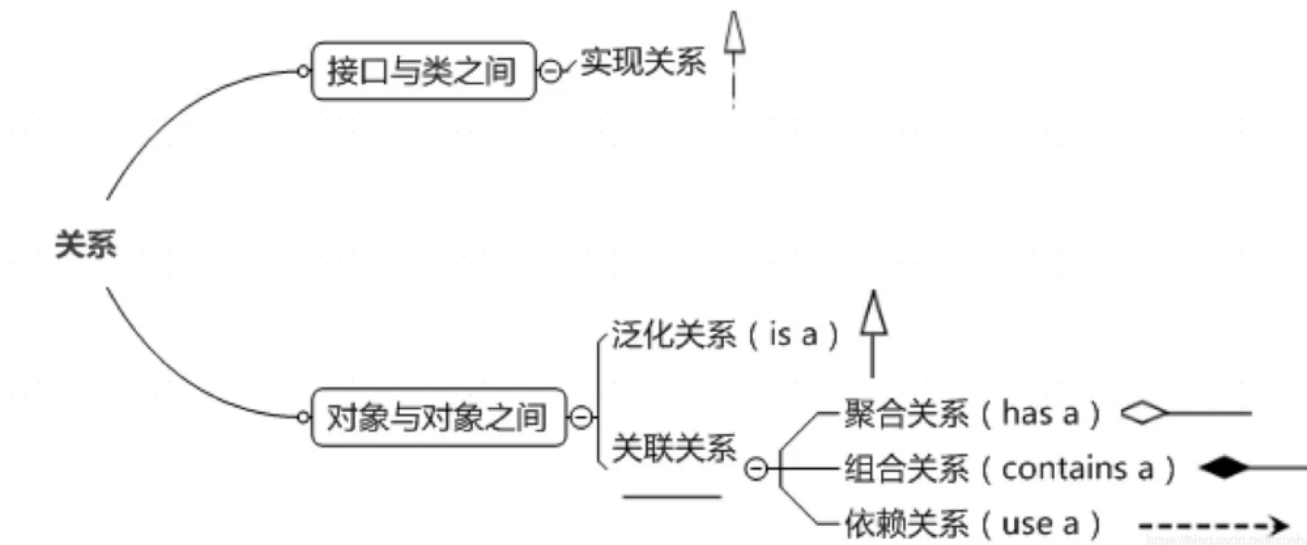


UML类图关系类别及表示方式

1 关系类别

在UML类图中，常见的关系有泛化（Generalization），实现（Realization），关联（Association），聚合（Aggregation），组合（Composition），依赖（Dependency）等



2 关系含义及表示方式（以C++代码为例）

1) 实现关系

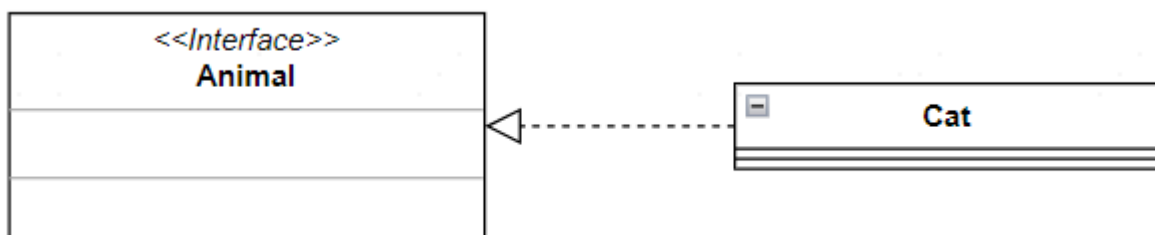
实现关系是指接口及其实现类之间的关系。

代码形式：使用纯虚函数实现抽象类作为接口，派生类为实现类：

```
class Animal
{
public:
    virtual void do_something() = 0;
};

class Cat : public Animal
{
public:
    void do_something() override
    {
    }
};
```

UML类图表示：



2) 泛化关系（继承关系 “is a”）

泛化关系是指对象与对象之间的继承关系。如果两个对象之间拥有“is a”的关系，那两者之间就存在继承关系。

代码形式：使用纯虚函数实现抽象类作为接口，派生类为实现类：

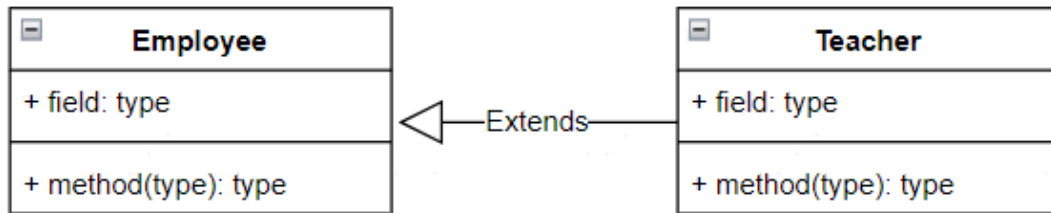
```

class Employee
{
};

class Teacher : public Employee
{
};

```

UML类图表示：



3) 关联关系 (Association)

关联关系 (Association) 是指对象和对象之间的连接，它使一个对象知道另一个对象的属性和方法。如果一个对象的类代码中，包含有另一个类的实例对象，那么这两个对象之间就是**关联关系**。关联关系有单向关联和双向关联。如果两个对象都知道（即可以调用）对方的公共属性和操作，那么二者就是双向关联。如果只有一个对象知道（即可以调用）另一个对象的公共属性和操作，那么就是单向关联。大多数关联都是单向关联，单向关联关系更容易建立和维护，有助于寻找可重用的类。

代码形式：使用纯虚函数实现抽象类作为接口，派生类为实现类：

```

class TimeCard
{
};

class Employee
{
private:
    TimeCard tc_;
};

```

UML类图表示（双向关联关系用带双箭头的实线或者无箭头的实线双线表示。单向关联用一个带箭头的实线表示，箭头指向被关联的对象）：



注：

- 数字：精确的数量
- *或者0..*：表示0到多个
- 0..1：表示0或者1个，在Java中经常用一个空引用来实现
- 1..*：表示1到多个

关联关系又分为**依赖关联**，**聚合关联**和**组合关联**三种类型。

a) 依赖关系 (Dependency)

依赖 (Dependency) 关系是一种弱关联关系。如果对象A用到对象B，但是和B的关系不是太明显的时候，就可以把这种关系看作是依赖关系。如果对象A依赖于对象B，则 A “use a” B。比如驾驶员和汽车的关系，驾驶员使用汽车，二者之间就是依赖关系。

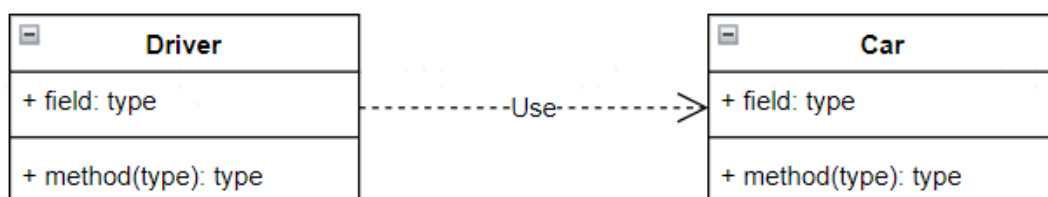
代码形式：

```
class Car
{
};

class Driver
{
private:
    void driverCar(Car c){

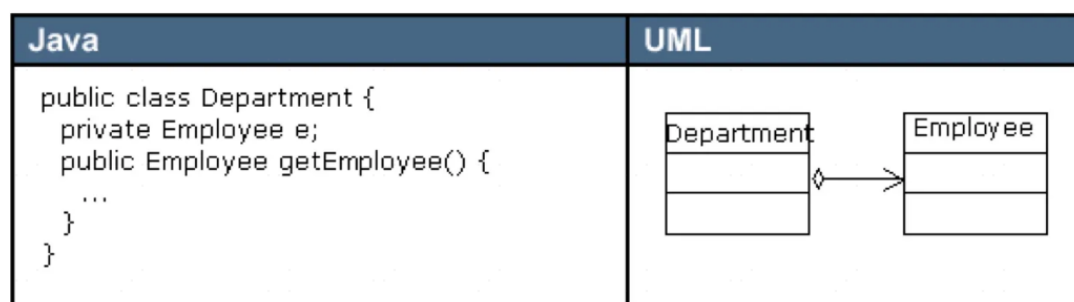
    }
};
```

UML类图表示（双向关联关系用带双箭头的实线或者无箭头的实线双线表示。单向关联用一个带箭头的实线表示，箭头指向被关联的对象）：



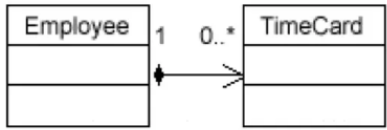
b) 聚合关系 (Aggregation)

聚合 (Aggregation) 是关联关系的一种特例，它体现的是整体与部分的拥有关系，即 “has a” 的关系。此时整体与部分之间是可分离的，它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，所以聚合关系也常称为共享关系。例如，公司部门与员工的关系，一个员工可以属于多个部门，一个部门撤消了，员工可以转到其它部门。



c) 组合关系 (Composition)

组合 (Composition) 也是关联关系的一种特例，它同样体现整体与部分间的包含关系，即 “contains a” 的关系。但此时整体与部分是不可分的，部分也不能给其它整体共享，作为整体的对象负责部分的对象的生命周期。这种关系比聚合更强，也称为强聚合。如果A组合B，则A需要知道B的生存周期，即可能A负责生成或者释放B，或者A通过某种途径知道B的生成和释放。

Java	UML
<pre> public class Employee { private TimeCard tc; public void maintainTimeCard() { ... } } </pre>	 <pre> classDiagram class Employee class TimeCard Employee "1" --> "0..*" TimeCard </pre>