面向对象方法与C++程序设计

第4章　　继承

大连理工大学
主讲人-赵小薇

继承中成员同名



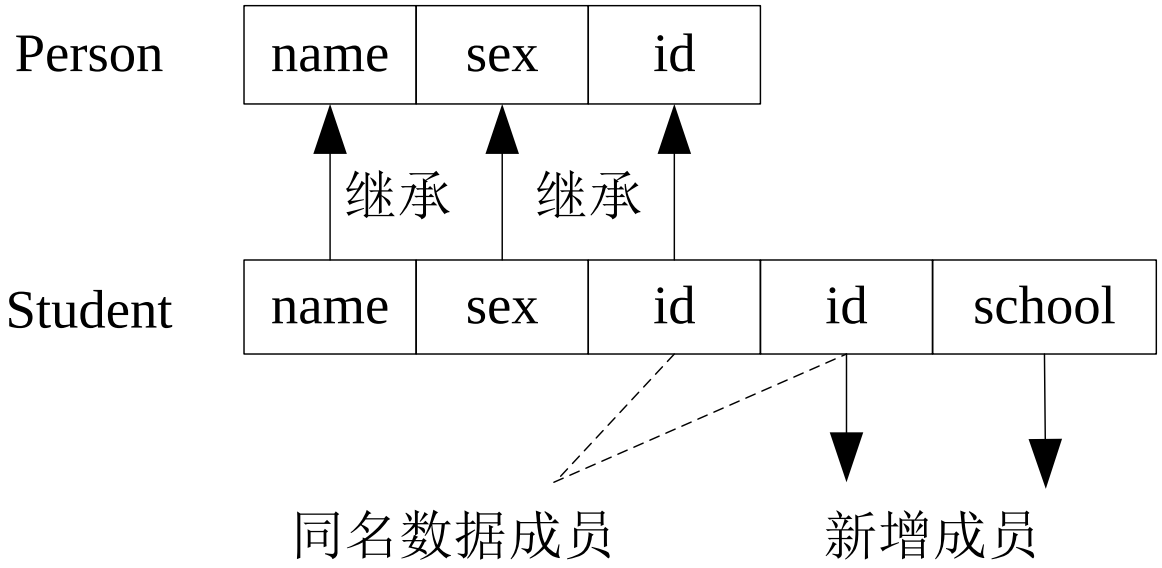
- 有两种情况：
- 基类成员与派生类成员同名
- 多重继承时不同基类成员同名

基类成员与派生类成员同名



```
class Person{                          // 基类
protected:
    char name[10];                     // 姓名
    char sex;                          // 性别
    int id;                            // 身份证
};
class Student : public Person{         // 派生类
private:    int id;                    // 学号
        char school[10];               // 学校
public:         void test(){           //测试函数
            id=123;                    // 究竟访问哪一个？
        }   };
```

基类与派生类定义了同名成员



- 派生类定义了与基类相同的成员，此时基类的同名成员在派生类内不可见，也就是派生类成员隐藏了同名的基类成员，这种现象称为**继承时的同名成员隐藏规则。**

类名限定符



- 基类成员与派生类成员同名，可以通过类名限定符“::”来解决。其语法为：

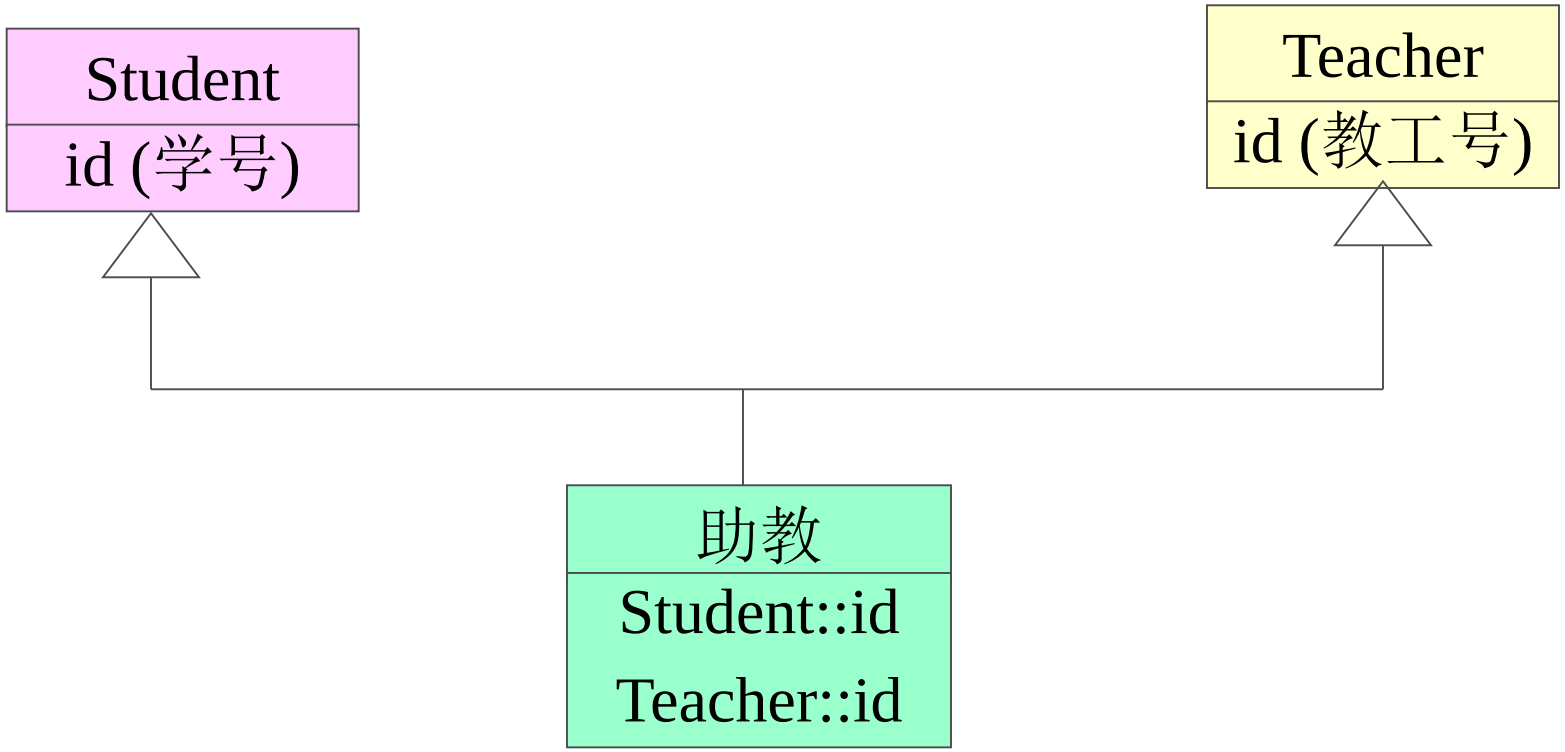
  类名::成员

```
class Student : public Person{        // 派生类
private:          int id;                          // 学号
                ……
public:           void test(){              //测试函数
                   id=123;                 // 访问派生类成员
                   Person::id =456; // 访问基类成员
                 }
 };
```

多重继承的成员同名



- 多重继承时不同基类成员同名也可以用类名限定符“::”来解决。



继承中的同名成员访问



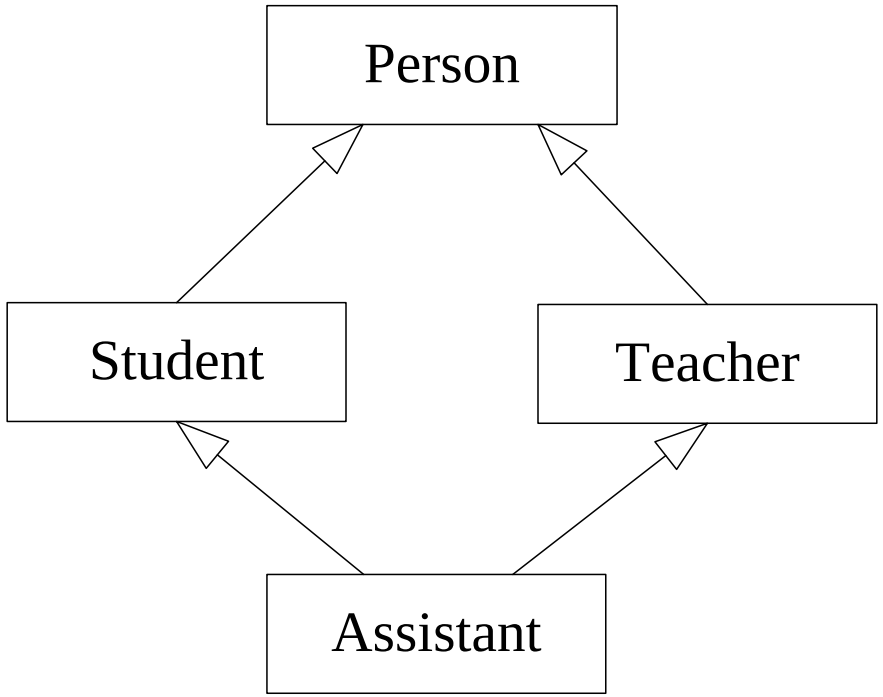
```
class Student
 {
   protected:
         int   id;
 };
```

```
class Teacher
 {
   protected :
         int   id;
 };
```

```
class Assistant: public Student, public Teacher
 {      public:
          void print(){
            cout<<id<<endl;     //error!访问是二义的
            cout<<Student::id<<endl;     //访问Student的id
            cout<<Teacher::id<<endl;     //访问Teacher的id
           }
 };
```

多重继承中引起的二义性

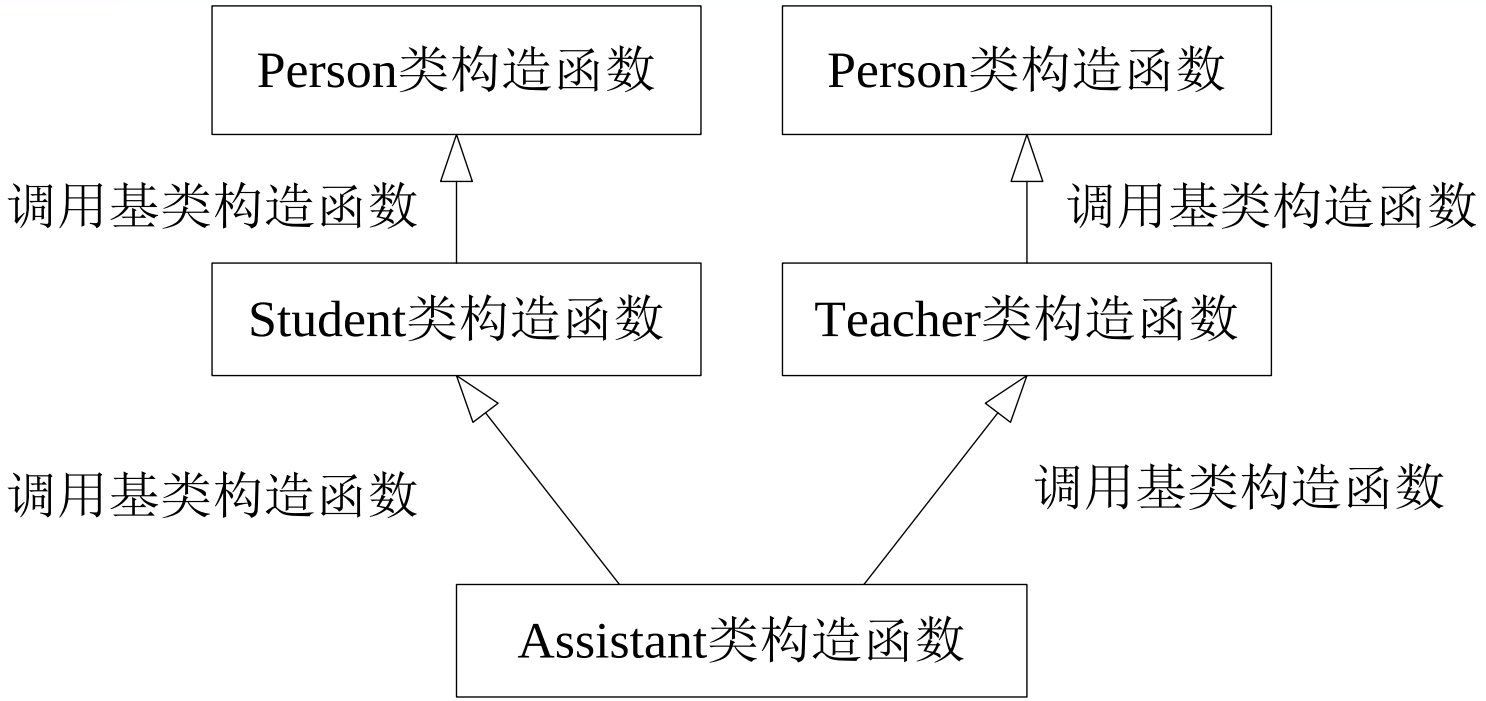
- 多个基类继承来的成员有同名现象，可以使用类名限定符“::”来进行区分。但是，有个前提：这些基类不是从一个共同的基类派生来的。如果不具备这个前提条件，在访问同名成员时会出现问题。



举例



```
class Person
  {   protected:  int id; };    // 身份号码
class Student : public Person
 {  protected:    int id; };          // 学号
class Teacher : public Person
 {  protected:   int id; };          // 职工号
class Assistant: public Student,public Teacher{
public:   void test(){                  // 测试
            Student::id  = 1001;        // 正确：访问Student类的id
            Teacher::id = 101;          // 正确：访问Teacher类的id
           Person::id   =10001;         // 错误！为什么？
          }
};
```



- 内存中实际上存在两个基类Person的对象，一个是由Student这一派生分支产生，另一个是由Teacher的派生分支产生的。

```
Person::id   =10001; // 二义性错误
```

- 无法传递给系统足够的信息，系统无法知道该语句到底要访问哪一个id。

虚基类

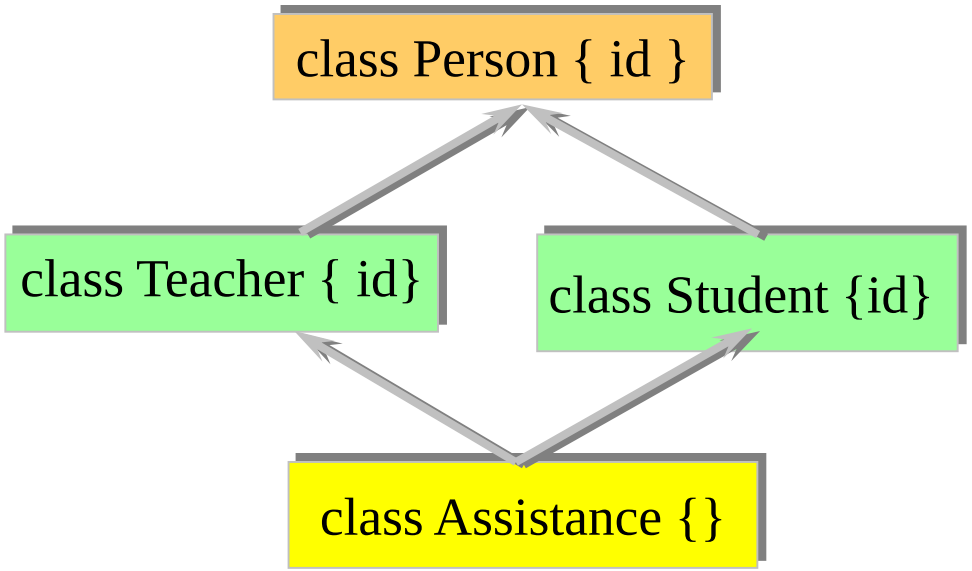


- C++语言允许程序中只建立公共基类的一个副本，将直接基类的共同基类设置为**虚基类**（virtual base class），这时从不同路径继承过来的该类成员在内存中只拥有一个副本。

- 虚基类的声明是在派生类继承基类时定义的，其语法形式如下：
- class 派生类名：virtual [继承方式] 基类名



```
class Person{   protected:        int id; };
class Teacher: virtual {   protected:      int id; };
class Student: virtual {   protected:      int id; };
class  Assistant : public Teacher , public Student {
public:
      void test(){
      Student::id  = 1001;          // 正确：访问Student类的id
      Teacher::id = 101;           // 正确：访问Teacher类的id
      Person::id   =10001;        // 正确：内存只有一个Person对象
    }
};
```


class Person { id }
class Teacher { id}
class Student {id}
class Assistance {}