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

基类与派生类的转化



- 三种继承方式中，只有公有继承使派生类完整地保留了基类的所有特征，因此，公有继承时，一个派生类的对象可用于基类对象适用的地方，需要基类对象的任何地方都可以使用派生类对象替代。



- 假设存在以下的继承关系：

```
class A{
public:
        int i;
};
class B :public A{
public :
        int j;
};
```

基类A与派生类B间的赋值兼容关系



1　派生类对象可以直接赋值给基类对象。

```
A  a;    // 基类对象
B  b;    // 派生类对象
a = b;  // 直接赋值
```

- 注意，赋值后a的数据类型依然是A，因此不要企图使用a去访问派生类的成员，因此下面的语句是错误的：

```
a.j = 3;    // 错误，A中不具有j成员
```

2 派生类对象可以初始化基类的引用。

```
A & c = b;  //  c是基类的引用，使用派生类对象b初始化c
```

- 函数的形式参数如果是一个基类的引用，在实际调用该函数的时候，可以传递一个派生类对象来代替基类对象。

```
void function(A & c)          // 形式参数是基类A的引用
 {  cout<<c.i<<endl;  }       // 输出A的数据成员i
```

- 实际调用该函数时使用如下的方式：

```
function(b);                  //实际参数是派生类B的对象
```



3．派生类对象的地址可以赋值给基类的指针。

```
A * p = & b ;   //基类指针p指向派生类对象b
p->i = 5 ;      //使用基类指针对派生类对象的成员进行访问
```

或者

```
A *p = new B();
p->i = 5 ;
```

举例



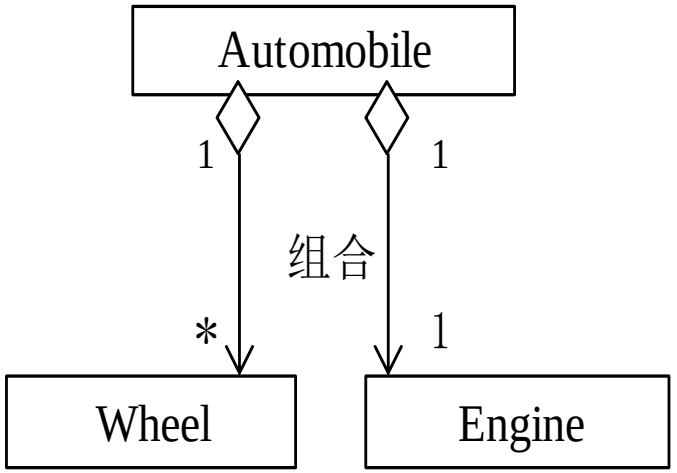
```
class Person{  …
public: void print(){ cout<<"人id：  "<<id<<endl; }
};
class Student : virtual public Person{…
public:  void print(){…}
};
class Teacher : virtual public Person{…
public:  void print(){…}
};
class Assistant:public Student,public Teacher{…
public:  void print(){…}
};
```

```
Person p(10001);                                    // 创建Person对象
Student s(20304,997,"dlut");                        // 创建Student对象
Teacher t(11100,7,"Professor");                     // 创建Teacher对象
Assistant a(40062,999,"dlut",18,"assistant",543.21f);   // Assistant对象
 Person * group[4];                                 // 声明Person的指针数组
 group[0]=&p;                                       // 指向Person对象
 group[1]=&s;                                       // 指向Student对象
group[2]=&t;                                        // 指向Teacher对象
group[3]=&a;                                        // 指向Assistant对象
for(int i=0;i<4;i++)
 group[i]->print();                                 // 调用print函数
```

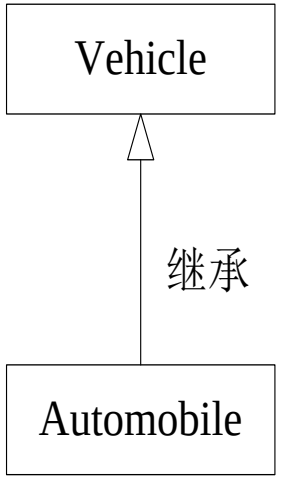
print()在实际运行中执行了原来基类中的print函数还是派生类特有的print()?

类与类的关系

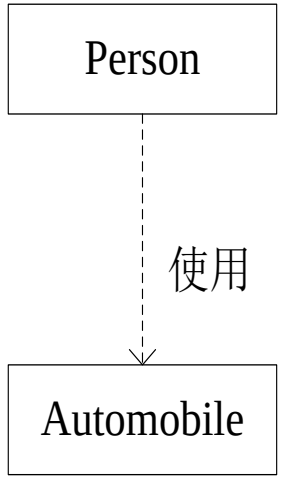
- C++的程序是由类组成的。类和类之间不是彼此孤立的，就像现实世界中的万事万物一样，相互之间存在各种各样的联系。类与类的关系可分为三种：



(a)



(b)



(c)

类的组合（has-a关系）



```
class Engine{                              // 发动机类
public:  void work();      };              // 发动机运转
class Wheel{                               // 车轮类
public:    void roll(); };                 // 车轮转动
class Automobile{                          // 汽车类
private:   Engine * engine;                // 拥有一个发动机
           Wheel* wheel;                   // 拥有四个车轮
public:    void move(){                    // 汽车行进
           engine->work();                 // 使用成员对象
      for(int i=0;i<4;i++) wheel[i]->roll(); // 使用成员对象
      }
 };
```

类的继承（is-a关系）



```
class Vehicle{                          // 交通工具类
protected:
        double weight;                  // 重量
         float speed;                   // 速度
public:
        void run();                     // 运行
};
class Automobile:public Vehicle{   // 汽车类：就是交工工具
private:  int  load;                      // 载客数
public:
    void move(){run();            // 可以直接调用基类成员
 }   };
```

类的使用关系(use-a关系)



```
class Person{
public:
    void buy(Automobile & auto){
        auto.move();                    // Person类使用汽车类
    }
};
void main(){
    Person person;
    Automobile auto;
    person.buy(auto);
 }
```

继承、组合和使用



- 继承、组合和使用是面向对象程序设计中常采用的三种类关系
- 类与类的“继承关系”是最紧密的关系，而“使用关系”是相对比较松散的关系。在实际的程序设计中，究竟使用哪一种关系，要看具体的实际需要，灵活使用。