类继承与对象组合是实现类复用的两种最常用的技术。

一：继承

　　继承是Is a 的关系，比如说Student继承Person,则说明Student is a Person。继承的优点是子类可以重写父类的方法来方便地实现对父类的扩展。

　　继承的缺点有以下几点：

　　1：**父类的内部细节对子类是可见的。**

　　2：子类从父类继承的方法在编译时就确定下来了，所以**无法在运行期间改变从父类继承的方法的行为。**

　　3：**子类与父类是一种高耦合，违背了面向对象思想。**

      4 ：**继承关系最大的弱点是打破了封装**，子类能够访问父类的实现细节，子类与父类之间紧密耦合，子类缺乏独立性，从而影响了子类的可维护性。

      5：**不支持动态继承。在运行时，子类无法选择不同的父类。**

二：组合

      1：**不破坏封装，**整体类与局部类之间松耦合，彼此相对独立。

      2：**具有较好的可扩展性。**

      3：**支持动态组合**。在运行时，整体对象可以选择不同类型的局部对象。

**组合是has a的关系   
继承是is a的关系**

引用一句老话应该更能分清继承和组合的区别：组合可以被说成“我请了个老头在我家里干活” ，继承则是“我父亲在家里帮我干活"。

总结

1.除非考虑使用多态，否则优先使用组合。  
2.要实现类似”多重继承“的设计的时候，使用组合。  
3.要考虑多态又要考虑实现“多重继承”的时候，使用组合+接口。

最后，随便写个例子吧哈~

要实现的目标：鸟（Bird）和狼（Wolf）都是动物（Animal），动物都有心跳（beat()），会呼吸（beat()），但是鸟会fly（fly()），狼会奔跑（run()），用java程序实现以上描述。

InheritTest.java 使用继承方式实现目标

CompositeTest.java 使用组合方式实现目标

**[java]** [view plain](https://blog.csdn.net/ycjnx/article/details/40952969) [copy](https://blog.csdn.net/ycjnx/article/details/40952969)

1. //InheritTest.java 使用继承方式实现目标
2. **class** Animal{
3. **private** **void** beat(){
4. System.out.println("心脏跳动...");
5. }
6. **public** **void** breath(){
7. beat();
8. System.out.println("吸一口气，呼一口气，呼吸中...");
9. }
10. }
11. //继承Animal，直接复用父类的breath()方法
12. **class** Bird **extends** Animal{
13. //创建子类独有的方法fly()
14. **public** **void** fly(){
15. System.out.println("我是鸟，我在天空中自由的飞翔...");
16. }
17. }
18. //继承Animal，直接复用父类的breath()方法
19. **class** Wolf **extends** Animal{
20. //创建子类独有的方法run()
21. **public** **void** run(){
22. System.out.println("我是狼，我在草原上快速奔跑...");
23. }
24. }
25. **public** **class** InheritTest{
26. **public** **static** **void** main(String[] args){
27. //创建继承自Animal的Bird对象新实例b
28. Bird b=**new** Bird();
29. //新对象实例b可以breath()
30. b.breath();
31. //新对象实例b可以fly()
32. b.fly();
33. Wolf w=**new** Wolf();
34. w.breath();
35. w.run();
36. /\*
37. ---------- 运行Java程序 ----------
38. 心脏跳动...
39. 吸一口气，呼一口气，呼吸中...
40. 我是鸟，我在天空中自由的飞翔...
41. 心脏跳动...
42. 吸一口气，呼一口气，呼吸中...
43. 我是狼，我在草原上快速奔跑...
45. 输出完毕 (耗时 0 秒) - 正常终止
46. \*/
47. }
48. }
50. //CompositeTest.java  使用组合方式实现目标
51. **class** Animal{
52. **private** **void** beat(){
53. System.out.println("心脏跳动...");
54. }
55. **public** **void** breath(){
56. beat();
57. System.out.println("吸一口气，呼一口气，呼吸中...");
58. }
59. }
60. **class** Bird{
61. //定义一个Animal成员变量，以供组合之用
62. **private** Animal a;
63. //使用构造函数初始化成员变量
64. **public** Bird(Animal a){
65. **this**.a=a;
66. }
67. //通过调用成员变量的固有方法（a.breath()）使新类具有相同的功能（breath()）
68. **public** **void** breath(){
69. a.breath();
70. }
71. //为新类增加新的方法
72. **public** **void** fly(){
73. System.out.println("我是鸟，我在天空中自由的飞翔...");
74. }
75. }
76. **class** Wolf{
77. **private** Animal a;
78. **public** Wolf(Animal a){
79. **this**.a=a;
80. }
81. **public** **void** breath(){
82. a.breath();
83. }
84. **public** **void** run(){
85. System.out.println("我是狼，我在草原上快速奔跑...");
86. }
87. }
88. **public** **class** CompositeTest{
89. **public** **static** **void** main(String[] args){
90. //显式创建被组合的对象实例a1
91. Animal a1=**new** Animal();
92. //以a1为基础组合出新对象实例b
93. Bird b=**new** Bird(a1);
94. //新对象实例b可以breath()
95. b.breath();
96. //新对象实例b可以fly()
97. b.fly();
98. Animal a2=**new** Animal();
99. Wolf w=**new** Wolf(a2);
100. w.breath();
101. w.run();
102. /\*
103. ---------- 运行Java程序 ----------
104. 心脏跳动...
105. 吸一口气，呼一口气，呼吸中...
106. 我是鸟，我在天空中自由的飞翔...
107. 心脏跳动...
108. 吸一口气，呼一口气，呼吸中...
109. 我是狼，我在草原上快速奔跑...
111. 输出完毕 (耗时 0 秒) - 正常终止
112. \*/
113. }
114. }