编译时类型和运行时类型：

Java的引用变量有两个类型，一个是编译时类型，一个是运行时类型，编译时类型由声明该变量时使用的类型决定，运行时类型由实际赋给该变量的对象决定。

如果编译时类型和运行时类型不一致，会出现所谓的多态。

因为子类其实是一种特殊的父类，因此java允许把**一个子类对象直接赋值给一个父类引用变量，无须任何类型转换，或者被称为向上转型**，由系统自动完成。

引用变量在编译阶段只能调用其编译时类型所具有的方法，但运行时则执行它运行时类型所具有的方法（意思是说：编写代码时，只能调用父类中具有的方法，如果子类重写了该方法，运行时实际调用的是运行时类型的该方法。程序在编译时，会在编译类型中检查是否具有所调用的方法，如果编写代码时，使用引用变量调用子类中的特有方法，或者调用重载了父类中的方法，而父类中找不到该方法，则会报编译错误），因此，编写Java代码时，引用变量只能调用声明该变量所用类里包含的方法。与方法不同的是，对象的属性则不具备多态性。通过引用变量来访问其包含的实例属性时，系统总是试图访问它编译时类所定义的属性，而不是它运行时所定义的属性。转换。  
——要访问子类中特有的方法和属性，在编写代码时，则必须进行类型转换。

 —— 以上摘自《疯狂Java讲义》

1.子类方法覆盖父类方法（ 子类重写父类中的方法，调用子类中的方法）

1. class Father{
2. public void method(){
3. System.out.println("父类方法："+this.getClass());
4. }
5. }
6. public class Son extends Father{
7. public void method(){
8. System.out.println("子类方法："+this.getClass());
9. }
10. public static void main(String[] args){
11. Father instance = new Son();
12. instance.method();
13. }
14. }

    运行结果：

    子类方法：class Son

    2. 子类没有重写父类中的方法，所以到父类中寻找相应的方法

* 1. class Father{
  2. public void method(){
  3. System.out.println("父类方法："+this.getClass());
  4. }
  5. }
  6. public class Son extends Father{
  7. public static void main(String[] args){
  8. Father instance = new Son();
  9. instance.method();
  10. }
  11. }
      1. 运行结果： 父类方法： class Son

    3.动态绑定只是针对对象的方法，对于属性无效。**因为属性不能被重写**。

* + - 1. class Father{
      2. private String name = "Father'name";
      3. }
      4. public class Son extends Father{
      5. public String name = "Son'name";
      6. public static void main(String[] args){
      7. Father instance = new Son();
      8. System.out.println(instance.name);
      9. }
      10. }
          1. 运行结果：Father'name这里还可以从另外一个方面来说明：如果将Father类的**[java]**view plaincopyprint?

public String name = "Father'name"; 修改为

private String name = "Father'name"; System.out.println(instance.name);

那么编译器将报错： 错误: Father中**private**name不可以访问

说明在System.out.println(instance.name);这行代码执行时，访问的是父类的 name属性，而该属性被声明为private，所以无法访问，因而报错！

    下面在分析一个例子：

1. class A
2. {
3. int count = 20;
4. }
5. class B extends A
6. {
7. int count = 200;
8. }
9. public class Test
10. {
11. public static void main(String[] args)
12. {
13. A a = new A();
14. System.out.println(a.count);
15. B b = new B();
16. System.out.println(b.count);
17. A ab = b; //向上转型
18. System.out.println(ab.count);
19. }
20. }
21. 运行结果 ： 20
22. 200
23. 20 结果分析：

    前两行的输出毫无疑问，问题在

    A ab = b;

    System.out.println(ab.count);

    的输出是20，而不是200；在这之间我们可以用

    System.out.println(ab == b);

    来进行简单的判断，结果输出为 true ，说明 ab 和 b 两个引用变量指向同一个实例，既然 ab 和 b 指向同一个实例，为什么输出的是20不是200呢？原因在于：

    1.对于 class A 和class B来说，class B是class A的子类，由于子类的变量并不会覆盖父类的变量，所以实际上在class B中是存在来两个count，在这分别记作 A.count 和B.count ；

    2.虽然在 class B中存在A.count 和B.count ，但是究竟输出那一个 count ，**取决于该引用变量的声明时类型（本文开头红色文字部分已经说明）**，此处 声明时类型 是 class A，所以输出 20 即A.count ，同理若改为 B ab = b ；则输出 200 即 B.count