"super"。它和"this"类似，但是也有不同的地方。   
1。表示调用父类的[构造函数](https://www.baidu.com/s?wd=%E6%9E%84%E9%80%A0%E5%87%BD%E6%95%B0&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3PyR1uAPBPH99n1R1PjFW0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EnHf1PW0Ynj0)。也是一个特殊语法，不是变量，没有什么类型。  
  
2。表示调用父类的方法（无视多态（重写））。可以在一个类的**非static成员**内部使用。比如super.method()。 Static方法由类调用,隶属于Class对象  
但是，注意，这个super.method()只是长得跟some\_var.method()一样，一个语法糖而已。实质上，"super"根本不是一个变量。  
  
为什么不是？因为如果是就坏了。java里面有一个金科玉律：**任何public非static函数的 调用 都是多态的。 动态绑定**  
所以，**如果**super是个变量，也指向当前对象，那么，不管super的静态类型是什么super.method()必然调用的是子类的那个版本，而不会是**我们期望的，静态（指不随多态变化）地选择父类的那个版本。**  
所以，你只要把super.xxx看成一个特殊的语法，比如理解为“super::xxx”就好了。  
  
既然super不是一个变量，那么不能把它象一个普通变量那样"=="，或者传递给某个函数就看起来很正常了，是么？何况，你其实也用不着它，有this你这些要求就都可以办到了。  
  
3。super的另外一个作用是**调用父类的protected函数**。**只有通过"super"这个魔咒，我们才能操作父类的protected成员，别无它法**。Super.method()