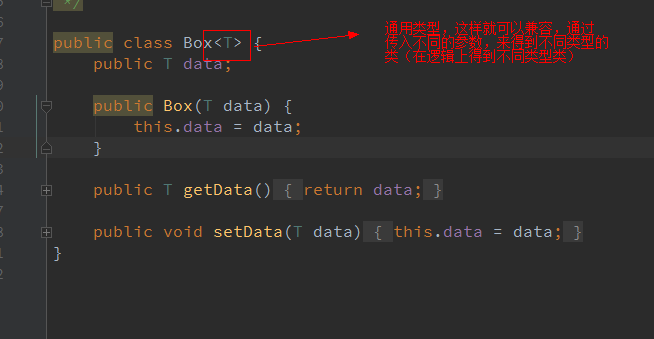
Java 泛型就是将原来的具体类型参数化，类似于方法中的变量参数，类型定义成参数形式，用的时候，再去传入具体类型。

比如一个String的list，我可能要定义一个StringList的类，来接受String；int的list需要定义IntString类，这样就太麻烦了。所以我们把list参数化，用形参表示，这样我们申明list的时候，用哪种类型，就传入那个类型。用String我们就传入List<String>.这样有公用性。从暂且看，泛型的功能就是兼容。

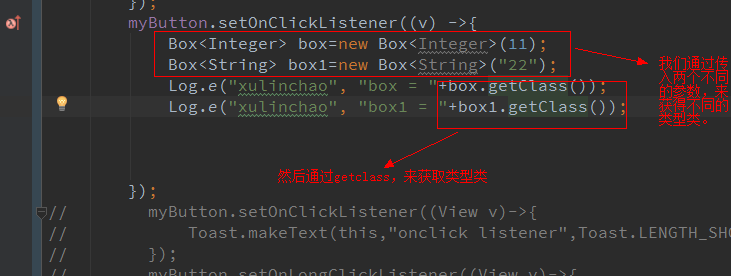
Java中getclass()是获得类型类，就是用class表示一个类型，例如用String类来表示String类型（就是字符串类型），注意这个类型类的获取是从.class字节码中的class对象，校验后会去掉传入的任何属性，同一个泛型所以都是一样的。

简单来说，泛型就是通用类型类，传入参数，就是具体类型类。

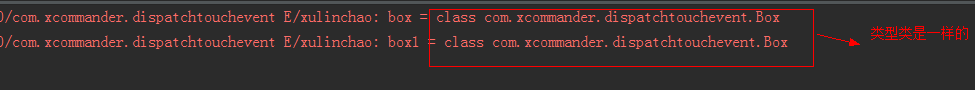
在使用泛型时，传入不同的参数，逻辑上就会得到不同类型的类（也就是类型类）如下图，

类型类是用来定义一个类型的。

通过上面的getclass可以知道，为什么说是逻辑上的类型类呢，看下图:



观察获取后的结果：

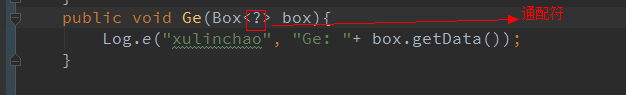


这个是为什么呢？

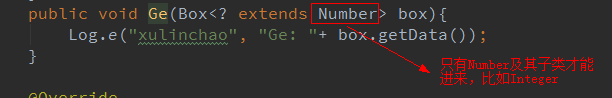
因为泛型传入参数，变成具体类型类，但是这个具体类型类，仅限于代码编译阶段，这个阶段会用传入的参数类型，来校验这个类的实例对象是否正确（传入的类型是否正确），如果发生错误，则编译不过；如果编译通过，则抹去传入的具体类型的信息。因为之前在编译阶段都校验过了。所以完全可以擦除类型。所以泛型的信息不回带到运行时。

现在说下类型通配符

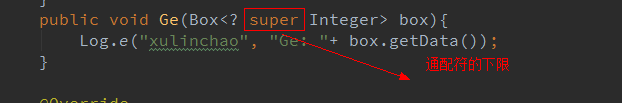
就是我一个方法，参数是一个类型类变量，为了让所有类型类都能匹配，我们用？表示，



如果我想设定通配符的上限，也就是说只允许一个类及其子类才允许，传进来。



如果我想设定通配符的下限，也就是说只有这个类及其父类才能进来，



这里说的进入，是指编译时通过校验，编译后就会擦除这些实际传过来的参数。