# WildcardType 泛型表达式

通配符表达式，泛型表达式，也可以说是，限定性的泛型，形如：? extends classA、？super classB。

## 源码

public interface WildcardType extends Type {

//获得泛型表达式上界（上限）

Type[] getUpperBounds();

//获得泛型表达式下界（下限）

Type[] getLowerBounds();

}

## 概述

WildcardType，通配符表达式，Type子接口，但是在Java中并没有WildcardType类型

## 源码详解

1.getUpperBounds   
获取泛型表达式上界，根据API的注释提示：现阶段通配符表达式仅仅接受一个上边界或者下边界，这个和定义类型变量时候可以指定多个上边界是不一样。但是API说了，为了保持扩展性，这里返回值类型写成了数组形式。实际上现在返回的数组的大小就是1，通配符?指定多个上边界或者下边界现在是会编译出错的（jdk1.7是这样的，至于7及以后就不知道了）。   
2.getLowerBounds   
获取泛型表达式下界。

public <T> void test(List<? extends classA > a){}Method method = Main.class.getMethod("test",List.class);

Type[] upperBounds = null;

Type[] lowerBounds = null;

Type[] types = method.getGenericParameterTypes();

for(Type type : types){

Type[] actualTypeArgument = ((ParameterizedType)type).getActualTypeArguments();

for(Type t : actualTypeArgument){

WildcardType wildcardType = (WildcardType) t;

lowerBounds = wildcardType.getLowerBounds();

upperBounds = wildcardType.getUpperBounds();

System.out.println("通配符表达式类型是："+ wildcardType);

if(upperBounds.length != 0){

System.out.println("表达式上边界："+Arrays.asList(upperBounds));

}

if(lowerBounds.length != 0){

System.out.println("表达式下边界："+Arrays.asList(lowerBounds));

}

}

}

//输出结果

通配符表达式类型是：? extends com.fcc.test.classA

表达式上边界：[class com.fcc.test.classA]