# Java类型之参数化类型

## 源码

public interface ParameterizedType extends Type {

//1.获得<>中实际类型

Type[] getActualTypeArguments();

//2.获得<>前面实际类型

Type getRawType();

//3.如果这个类型是某个类型所属，获得这个所有者类型，否则返回null

Type getOwnerType();

}

Java类型Type详见：Type Java类型

## 概述

ParameterizedType，参数化类型，形如：Object<T, K>，即常说的泛型，是Type的子接口。

## 源码详解

1.getActualTypeArguments   
获得参数化类型中<>里的类型参数的类型，因为可能有多个类型参数，例如Map<K, V>，所以返回的是一个Type[]数组。   
【注意】无论<>中有几层<>嵌套，这个方法仅仅脱去最外层的<>，之后剩下的内容就作为这个方法的返回值，所以其返回值类型不一定。   
例如：   
1. List<ArrayList> a1;//这里返回的是，ArrayList，Class类型   
2. List<ArrayList<String>> a2;//这里返回的是ArrayList<String>，ParameterizedType类型   
3. List<T> a3;//返回的是T，TypeVariable类型   
4. List<? extends Number> a4; //返回的是WildcardType类型   
5. List<ArrayList<String>[]> a5;//GenericArrayType   
要注意，ArrayList与ArrayList<String>的不同。

public static void main(String[] args) throws Exception

{

Method method = new Main().getClass().getMethod("test", List.class);//这里的第二个参数，和getRawType()意义类似

Type[] types = method.getGenericParameterTypes();

ParameterizedType pType = (ParameterizedType) types[0];

Type type = pType.getActualTypeArguments()[0];

System.out.println(type);

//type是Type类型，但直接输出的不是具体Type的五种子类型，而是这五种子类型以及WildcardType具体表现形式

System.out.println(type.getClass().getName());

}

public void test(List<ArrayList<String>[]> a){

}//读者可将test中参数用相应类型代替

2.getRawType   
返回最外层<>前面那个类型，即Map<K ,V>的Map

Map<Integer, String> maps = new HashMap<>();

ParameterizedType pType = (ParameterizedType) maps.getClass().getGenericSuperclass();//获得HashMap的父类

System.out.println(pType.getRawType());//class java.util.AbstractMap

if(pType.getRawType() instanceof Class){

System.out.println("true");//true

}//注意类型（Type）与类（Class）的区别

3.getOwnerType   
获得这个类型的所有者的类型

public static void main(String[] args)

{

Method method = new GenericDemo().getClass().getMethod("applyMethod",Map.Entry.class);

Type[] types = method.getGenericParameterTypes();

ParameterizedType pType = (ParameterizedType)types[0];

//返回所有者类型，打印结果是interface java.util.Map

System.out.println(pType.getOwnerType());

}

public static <T,U> void applyMethod(Map.Entry<T,U> mapEntry){

}//Map接口就是Map.Entry的所有者