泛型在java中有很重要的地位，在面向对象编程及各种设计模式中有非常广泛的应用。

什么是泛型？为什么要使用泛型？

泛型，即“参数化类型”。一提到参数，最熟悉的就是定义方法时有形参，然后调用此方法时传递实参。那么参数化类型怎么理解呢？顾名思义，就是将类型由原来的具体的类型参数化，类似于方法中的变量参数，此时类型也定义成参数形式（可以称之为类型形参），然后在使用/调用时传入具体的类型（类型实参）。

泛型的本质是为了参数化类型（在不创建新的类型的情况下，通过泛型指定的不同类型来控制形参具体限制的类型）。也就是说在泛型使用过程中，操作的数据类型被指定为一个参数，这种参数类型可以用在类、接口和方法中，分别被称为泛型类、泛型接口、泛型方法。

泛型的使用

泛型有三种使用方式，分别为：泛型类、泛型接口、泛型方法

泛型类

泛型类型用于类的定义中，被称为泛型类。通过泛型可以完成对一组类的操作对外开放相同的接口。最典型的就是各种容器类，如：List、Set、Map。

泛型类的最基本写法（这么看可能会有点晕，会在下面的例子中详解）：

class 类名称 <泛型标识：可以随便写任意标识号，标识指定的泛型的类型>{

private 泛型标识 /\*（成员变量类型）\*/ var;

.....

}

}

一个最普通的泛型类：

//此处T可以随便写为任意标识，常见的如T、E、K、V等形式的参数常用于表示泛型

//在实例化泛型类时，必须指定T的具体类型

public class Generic<T>{

//key这个成员变量的类型为T,T的类型由外部指定

private T key;

public Generic(T key) { //泛型构造方法形参key的类型也为T，T的类型由外部指定

this.key = key;

}

public T getKey(){ //泛型方法getKey的返回值类型为T，T的类型由外部指定

return key;

}

}

//泛型的类型参数只能是基本类型（包括自定义类），不能是简单类型

//传入的实参类型需与泛型的类型参数类型相同，即为Integer.

Generic<Integer> genericInteger = new Generic<Integer>(123456);

//传入的实参类型需与泛型的类型参数类型相同，即为String.

Generic<String> genericString = new Generic<String>("key\_vlaue");

Log.d("泛型测试","key is " + genericInteger.getKey());

Log.d("泛型测试","key is " + genericString.getKey());