如：List、Set、Map。

```
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;
  .....

  }
}
```

```
//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类类型（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());

12-27 09:20:04.432 13063-13063/? D/泛型测试: key is 123456
12-27 09:20:04.432 13063-13063/? D/泛型测试: key is key_vlaue
```

在使用泛型的时候如果传入泛型实参，则会根据传入的泛型实参做相应的限制，此时泛型才会起到本应起到的限制作用。如果**不传入泛型类型实参**的话，在泛型类中使用泛型的方法或成员变量定义的类型可以为**任何的类型**。

注意：

a. 泛型的类型参数只能是类类型，不能是简单类型。

a. 不能对确切的泛型类型使用instanceof操作。如下面的操作是非法的，编译时会出错。

```
if(ex_num instanceof Generic<Number>){
}
```

有界泛型

需要指定泛型的类型范围。有界类型就是在类型参数部分指定extends或super关键字，这里的extends也含有implements的功能，分别用上限或下限来限制类型范围，从而限制泛型的类型边界。

例如：

<T extends Animal>//限定T是Animal的子类

<T super Dog >//限定T是Dog的超类

多个限定时我们可以使用&来进行分割，这时关键词只能使用extends。与多重继承类似，这里只可以有一个类，其他都是接口。

```
<T extends Object&Comparable&Serializable>
```