链滴

学习笔记 | 泛型

作者：ellenbboe

原文链接：https://ld246.com/article/1565438077041

来源网站：链滴

泛型

泛型适用的是**类**类型,而不能是**简单类型**

泛型类型在逻辑上看以看成是多个不同的类型，实际上都是**相同**的基本类型。

List<Integer>与List<Boolean>的Class是一样的,所以说类型相同

泛型标识<泛型标识>可以声明在类和方法

?代表的某一种类型,而不是多种,是实际的参数,而不是像T的形式参数

泛型的使用

泛型类

```
public class GenericsClass<T> {
    private T key;
    public GenericsClass(T key)
    {
        this.key = key;
    }
    public T getKey() {
        return key;
    }
}
```

表明T是泛型类型

使用

```
GenericsClass<Integer> a = new GenericsClass<Integer>(123); //显示声明
GenericsClass a = new GenericsClass(123);
```

GenericsClass<?>可以匹配任何类型,但是只能使用Object的方法

泛型接口

```
public interface GenericsInterface<T> {
    public T hello();
}
```

实现接口

可以还是返回泛型

```
public class GenericsImpl<Anything> implements GenericsInterface {

    @Override

    public Anything hello() {
```

```
        return null;
    }
}
```

或者返回具体类

```
public class GenericsImpl implements GenericsInterface<Object> {
    @Override
    public Object hello() {
        return null;
    }
}
```

泛型方法

```
public class GenericsMethod<T> {
private T key;

    public T hello()

    {
        return key;
    }
    public <T> T hello(GenericsClass<T> genericsClass)
    {
        T a = genericsClass.getKey();
        return a;
    }
}
```

只要方法有<泛型标识>,那么即使方法中声明的泛型T与类声明的泛型T是一个字母,但是他们是两个不的泛型,所以说**泛型方法能使方法独立于类而产生变化**

类中定义使用泛型的静态方法，需要添加额外的泛型声明

```
public static <T> void show(T t){}
```

无论何时，如果你能做到，你就该尽量使用泛型方法。也就是说，如果使用泛型方法将整个类泛型化，

那么就应该使用泛型方法。另外对于一个static的方法而已，无法访问泛型类型的参数。

所以如果static方法要使用泛型能力，就必须使其成为泛型方法。

泛型的上下边界

```
GenericsClass<T extends Number>
```

表示GenericsClass可以声明为GenericsClass<Number及其子类>

GenericsClass<T super Number>

表示GenericsClass可以声明为GenericsClass<Number及其父类>

集合与泛型

List

普通的集合定义,可以存放任何数据

List<Object>

泛型限制Object,但是List<Integer>却不能赋值给List<Object>,可以赋值给List<?>

List<?>

泛型通配符,可以删除,赋值但是不能添加,

因为?代表着某一类型,而这个类型我们是未知的,所以我们添加不了

List<? extends T>

泛型通配符,可以删除,赋值T以及T的子类List,但是不能添加,Get可以返回T以及T的父类的对象

不管赋值什么,这个List都是T的子类,所以get的时候不管List引用了什么,都能向上转型为T

List<? super T>

泛型通配符,可以删除,赋值T以及T的父类List,添加T以及T的子类,Get可以返回Object对象

因为?一定是T或者T的父类,所以我们只要添加T或者T的子类的对象,这些对象都能向上转型为T或者T父类,返回的就是T或者T的父类

我的疑问与解答

有个 List<Object> 里有个 String 类型,List 可以赋值给 List<? super Number>

因为 Object 是 Number 的父类, 而 String 类型 被看成了 Object 对象

```
List<Object> a1 = new ArrayList();
a1.add("String");
List<? super Number> c = a1;
System.out.println(c.get(0).getClass());

List<Object> a1 = new ArrayList();
a1.add("String");
List<? super Number> c = a1;
System.out.println(c.get(0).getClass());`
```

还是得到了 class java.lang.String -.-

但是实际上String类型即使被当做Object类型,它还是String类型,只是不能转换为Number类型而已,以getclass出来的还是java.lang.string octopus