Java泛型

java泛型的应用可以提高的代码的复用性，同时泛型提供了类型检查，减少了数据的类型转换，同时保证了类型安全。

## 泛型的命名规范

为了与java关键字区别开来，java泛型参数只是使用一个大写字母来定义。各种常用泛型参数的意义如下：

* E — Element，常用在java Collection里，如：List<E>, Iterator<E>, Set<E>
* K, V — Key，Value，代表Map的键值对
* N — Number，数字
* T — Type，类型，如String，Integer等等
* S,U,V etc. - 2nd, 3rd, 4th 类型，和T的用法一样

### 基本泛型

***泛型类：***

Class Generator<E> {

E obj = new E();

}

***泛型接口：***

public interface List <E> {

void add(E x);

Iterator<E> iterator();

}

public interface Iterator<E> {

E next();

boolean hasNext();

}

***泛型方法：***

有时候我们并不希望整个类都被泛型化，这时可以只在某个方法上应用泛型。因为构造函数是一种特殊的方法，因此也可以在构造函数上应用泛型。

#### 泛型界限

有时候，你会希望泛型类型只能是某一部分类型，比如操作数据的时候，你会希望是Number或其子类类型。这个想法其实就是给泛型参数添加一个界限。其定义形式为：

<T extends BoundingType>

此定义表示T应该是BoundingType的子类型（subtype）。T和BoundingType可以是类，也可以是接口。另外注意的是，此处的“extends”表示的子类型，不等同于继承。

一个泛型参数可以有多重限制范围，使用“&”分隔。且限制范围中之多有一个类。如果用一个类作为限定，它必须是限定列表中的第一个。

如果说泛型方法是一个有用的工具，那泛参的界限就应该这个工具的灵魂，为这个工具添加了一些“行为准则”。

## 泛型类与子类型

无论类A与类B是否存在关联，MyClass<A>与MyClass<B>都没有任何关联，其共同的父类的是Object；

在泛型里，也存在子类型，前提是其泛型参数的限制并没有改变，可以认为泛参没有改变，其实就是从原来的类或接口来判断泛型的子类型：

如果 MyClassA extends MyClassB

那么 MyClassA<T> extends MyClassB<T> && MyClassA<T, V> extends MyClassB<T>