JDK1.5的新特性

1. **ForEach循环**：**一般只用于迭代或遍历，只是为了简化书写和美化代码而已**。

在Iterable接口中定义了**forEach方法**。

1. **forEach语句的格式：**

**for（类型 变量 ： 数组或集合）{**

**}//注意：集合只能是Collection单列集合，Map集合不可以，因为Map集合不是单列集合。**

1. ① 例如：**遍历集合**：

List<String> list = new ArrayList<String>();

list.add("abc1");list.add("abc2");list.add("abc3");

**for (String str : list)** {

System.out.println(str);

}

② **遍历数组**：

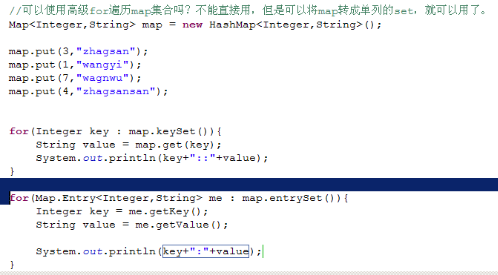
int[] a = {2,3,5,6,4};

for (int i : a) {

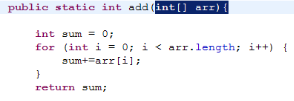
System.out.println(i);

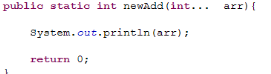
}//**注意i已经代码数组元素，不再是索引**。

1. **高级for循环（即forEach循环语句）与传统for循环的区别：**
2. **传统for循环可以控制循环的增量和判断条件，从而可以完成对语句执行很多次；而forEach语句只能完成一次遍历或者迭代；**
3. **forEach语句只是对传统for循环遍历的一种简化形式，forEach语句必须有遍历的目标，目标要么是数组，要么是Collection单列集合；而for语句比较随意，没有必要具有遍历目标，仅仅实现循环功能也可以；**
4. forEach语句只可以获取集合或者数组的每个元素，但是不能对每个元素进行删除、替换等操作；如果仅仅只是遍历获取元素，则可以利用forEach语句，但是如果对元素进行操作，建议使用传统for循环。
5. **高级for可以遍历Map集合吗？**

**不能直接使用，因为高级for只可以遍历Collection单列集合。如果先转成Set集合，就可以了。**

1. **函数可变参数：**
2. 格式：





1. **前者是必须先生成一个数组，才可以。**

**后者是直接传进去可变参数，然后自动会变成数组。**

1. **函数的可变参数**本质上是一个数组，但是接收的是数组的元素，自动将这些元素封装成数组。即使不传元素也可以执行，默认是一个空的数组。简化了调用者的书写。
2. 局限性：可变参数**只可以定义在参数列表的最后一个且只可以有一个**。

如**add （int a，int … arr） 正确**；**add(int … arr ; int a) 错误，因为把传进的数都封装到了数组中，后面a无值。**

1. 如**工具类Arrays中的asList方法**。



如：

**方式一：**

**List<String> list = new ArrayList<String>();**

**list.add("abc");**

**list.add("cde");**

**list.add("ghg");**

**方式二：**

**List<String> list2 = Arrays.asList("abc","cde","ghg");**

**System.out.println(list2+"\n"+list);**

1. **静态导入**：
2. 以前都是**导入的类**，现在引入了可以**导入类中的静态方法和静态属性**；
3. **导入的是类中的静态成员**：**静态方法和静态属性**。因此，导入方法之前，一定要判断这个方法是否是**静态方法**。

import static java.util.Collections.max;//导入**类Collections中的max方法**

**然后，就可以直接利用max方法。**

**max(list);//不再加Collection.**

import static java.util.Collections.\*;//导入的是**类Collections中的所有的静态成员**

**这样其中的所有的静态方法都可以直接利用了。**