

## 83-2、kotlin可变长参数的注意点

### (1) 定义

- 正常来说，主构造函数或普通函数只能一个一个的定义参数，kotlin中可以通过vararg定义可变长的参数，可以同时传多个参数，等同于java的...

### (2) 与java比较

- java中可变长参数只能声明在方法形参的末尾，需要先定义方法前面的形参，剩余的形参都是可变长参数。
- kotlin中由于可以使用具名参数，所以可变长参数可以声明在方法形参的任意位置。

复制代码

```
1
2 fun main(args:Array<String>){
3     printArray(3,"H","ello","world",arg2 =
4         "OK")
5 }
6 fun printArray(arg1:Int,vararg
7     array:String,arg2:String){
8     println("arg1=$arg1")
9     array.forEach(::print)
10    println("\narg2=$arg2")
11 }
12 //打印结果
13 arg1=3
14 Helloworld
```

```
15 arg2=OK
16
```

java >

### (3) 给方法的可变长参数传递的是一个数组时

- 需要使用spread操作符\*来标识数组，会将数组自动平铺展开到可变长参数中。
- 或者使用具名方式替换掉操作符\*也可以达到数组平铺到可变长参数中的效果。
- 唯一不足的是截止目前操作符，只支持将数组展开，不支持将集合展开。

复制代码

```
1
2 val s = arrayOf("H","ello"," world")
3   printArray(3,s,arg2 = "OK") ×
4   printArray(3,*s,arg2 = "OK") ✓
5   printArray(3,array = s,arg2 = "OK") ✓
6
7 fun printArray(arg1:Int,vararg
   array:String,arg2:String){
8     println("arg1=$arg1")
9     array.forEach(::print)
10    println("\narg2=$arg2")
11 }
```

java >

### (4) 直观演示

## 笔记目录



- 83-2、kotlin可变长参数的...
  - (1) 定义
  - (2) 与java比较
  - (3) 给方法的可变长参...
  - (4) 直观演示

- 使用具名方式传入数组实参，使用可变长形参接收，此时数组元素已经平铺到可变长参数中

```
15 fun main(args: Array<String>) { args: []
16     val s = arrayOf("H", "ello", " world") s: ["H", "ello", " world"]
17     printArray1(array = s) s: ["H", "ello", " world"]
18     printArray1(s)
19 }
new *
fun printArray1(vararg array:Any){
21
22 }
```

```
hangm *
fun main(args: Array<String>) {
    val s = arrayOf("H", "ello", " world")
    printArray1(array = s) array: ["H", "ello", " world"]
    printArray1(s)
}
new *
fun printArray1(vararg array:Any){ array: ["H", "ello", " world"] ~
}

array = {String[3]@1214} ["H", "ello", " world"] ~
> 0 = "H"
> 1 = "ello"
> 2 = " world"

Set value F2 Create renderer Add as inline
```

- 不使用具名方式，直接传入数组实参，使用可变长形参接收，此时可变长参数就是数组本身

```

15  ▶ fun main(args: Array<String>) {  args: []
16      val s = arrayOf("H", "ello", " world")  s: ["H", "ello", " world"]
17      printArray1(array = s)
18      printArray1(s)  s: ["H", "ello", " world"]
19  }
new *
20  fun printArray1(vararg array:Any){
21
22  }

```

```

15  ▶ fun main(args: Array<String>) {
16      val s = arrayOf("H", "ello", " world")
17      printArray1(array = s)  array: Object[1]@1220
18      printArray1(s)
19  }
new *
20  fun printArray1(vararg array:Any){  array: Object[1]@1220
21
22  }

```

array = {Object[1]@1220}

0 = {String[3]@1209} ["H", "ello", " world"]

- > 0 = "H"
- > 1 = "ello"
- > 2 = " world"

Set value F2   Create renderer   Add as

