彻底掌握Kotlin

集合

- 集合可以方便你处理一组数据，也可以作为值参传给函数，和我们学过的其他变量类型一样，List、Set和Map类型的变量也分为两类，只读和可变。

List集合

List创建与元素获取

- getOrElse是一个安全索引取值函数，它需要两个参数，第一个是索引值，第二个是能提供默认值的lambda表达式，如果索引值不存在的话，可用来代替异常。
- getOrNull是Kotlin提供的另一个安全索引取值函数，它返回null结果，而不是抛出异常。

```
1  fun main() {
2      val list:List<String> = listOf("Jason", "Jack", "Jacky")
3      //list[4]
4      println(list.getOrElse(index: 4) { "Unknown" })
5      println(list.getOrNull(index: 4))
6      println(list.getOrNull(index: 4) ?: "Unknown")
7  }
```

可变列表

- 在Kotlin中，支持内容修改的列表叫可变列表，要创建可变列表，可以使用mutableListOf函数。List还支持使用toList和toMutableList函数动态实现只读列表和可变列表的相互转换。

```
fun main() {
    val mutableList: MutableList<String> = mutableListOf("Jason", "Jack", "Jacky")
    mutableList.add("Jimmy")
    mutableList.remove( element: "Jack")
    println(mutableList)

    listOf("Jason", "Jack", "Jacky").toMutableList()
    mutableListOf("Jason", "Jack", "Jacky").toList()
}
```

mutator函数

- 能修改可变列表的函数有个统一的名字：mutator函数
- 添加元素运算符与删除元素运算符（还记得C++中的运算符重载吗？）
- 基于lambda表达式指定的条件删除元素

```
1  fun main() {
2      val mutableList: MutableList<String> = mutableListOf("Jason", "Jack", "Jacky")
3      mutableList += "Jimmy"
4      println(mutableList)
5
6      mutableList -= "Jason"
7      println(mutableList)
8
9      mutableList.removeIf { it.contains( other: "Jack") }
10     println(mutableList)
11 }
```

集合遍历

- for in 遍历
- forEach 遍历
- forEachIndexed 遍历时要获取索引

```
1  fun main() {
2      val list : List<String> = listOf("Jason", "Jack", "Jacky")
3      for (s : String in list) {
4          println(s)
5      }
6      list.forEach{ it: String
7          println(it)
8      }
9      list.forEachIndexed{ index, item ->
10         println("$index, $item")
11     }
12 }
```

解构

- 通过_符号过滤不想要的元素

```
1  fun main() {
2      val list: List<String> = listOf("Jason", "Jack", "Jacky")
3      val (origin: String, _: String, proxy: String) = list
4  }
```

Set集合

Set创建与元素获取

- 通过setOf创建set集合，使用elementAt函数读取集合中的元素。

```kotlin
fun main() {
    val set:Set<String> = setOf("Kotlin", "Java", "Scala")
    //没有这种写法
    //set[3]
    set.elementAt( index: 2)
}
```

可变集合

- 通过mutableSetOf创建可变的set集合

```
1  fun main() {
2      val mutableSet: MutableSet<String> = mutableSetOf("Kotlin", "Java", "Scala")
3      mutableSet += "Groovy"
4  }
```

集合转换

- 把List转换成Set，去掉重复元素
- 快捷函数

```
fun main() {
    val list:List<String> = listOf("Jason", "Jack", "Jacky", "Jacky")
        .toSet()
        .toList()
    println(list)

    println(listOf("Jason", "Jack", "Jacky", "Jacky").distinct())
}
```

数组类型

- Kotlin提供各种Array，虽然是引用类型，但可以编译成Java基本数据类型。

数组类型	创建函数
IntArray	intArrayOf
DoubleArray	doubleArrayOf
LongArray	longArrayOf
ShortArray	shortArrayOf
ByteArray	byteArrayOf
FloatArray	floatArrayOf
BooleanArray	booleanArrayOf
Array	arrayOf

ANDROID

Map集合

Map的创建

- to看上去像关键字，但事实上，它是个省略了点号和参数的特殊函数，to函数将它左边和右边的值转化成一对Pair。

```
1  fun main() {
2      val map: Map<String, Int> = mapOf("Jack" to 20, "Jason" to 18, "Jacky" to 30)
3      println(map)
4
5      mapOf(Pair("Jack",20), Pair("Jason",18))
6  }
```

读取Map的值

- [ ]取值运算符，读取键对应的值，如果键不存在就返回null
- getValue，读取键对应的值，如果键不存在就抛出异常
- getOrElse，读取键对应的值，或者使用匿名函数返回默认值
- getOrDefault，读取键对应的值，或者返回默认值

```
1  fun main() {
2      val map: Map<String, Int> = mapOf("Jack" to 20, "Jason" to 18, "Jacky" to 30)
3      println(map["Jack"])
4      println(map.getValue( key: "Jack"))
5      println(map.getOrElse( key: "Rose"){"unknown"})
6      println(map.getOrDefault( key: "Rose", defaultValue: 0))
7  }
```

遍历

- forEach遍历Map

```
fun main() {
    val map : Map<String, Int> = mapOf("Jack" to 20, "Jason" to 18, "Jacky" to 30)
    map.forEach{ it: Map.Entry<String, Int>
        println("${it.key},${it.value}")
    }
    map.forEach { (key: String, value: Int) ->
        println("${key},${value}")
    }
}
```

可变集合

- 通过mutableMapOf创建可变的Map
- getOrPut 键值不存在，就添加并返回结果，否则就返回已有键对应的值

```
fun main() {
    val mutableMap: MutableMap<String, Int> = mutableMapOf("Jack" to 20, "Jason" to 18)
    mutableMap += "Jimmy" to 30
    mutableMap.put("Jimmy", 31)

    println(mutableMap.getOrPut( key: "Jimmy"){18})
    mutableMap.getOrPut( key: "Rose"){18}
    println(mutableMap)
}
```