

ЦИКЛЫ В КОТЛИН

Массивы + циклы

Содержание

1. Циклы и прерывание
2. Массивы

1. Циклы и прерывание

В предыдущей лекции мы рассмотрели if else, when, is, as и ?. Теперь пора рассмотреть циклы. Начнем с цикла for. В джава мы писали `for (int i=0;i<10;i++)`, а в котлин будем писать так

```
for (i in 0..10) {  
    print(i)  
}
```

Помните мы уже говорили про range а..б когда проходили when. В котлин немного иначе пишем обратный цикл с иным шагом чем 1

В джава мы писали `for (int i=10;i>0;i-=2)` А в котлин это будет так:

```
for (i in 10 downTo 0 step 2) {  
    print(i)  
}
```

Да, вы не сможете написать range в обратном порядке, поэтому вот так. В одном месте в котлин удобно и коротко, а в другом не очень. Но вы привыкнете.

Хорошо, а что насчет массива? Во-первых он инициализируется иначе в котлин.

```
val array = arrayOf("a", "b", "c")  
for (i in array.indices) {  
    print(array[i])  
}
```

Тип массива указывать не обязательно, можно просто передать данные. Посмотрите как мы проходим циклом по индексам – вместо того чтобы писать вот так:

```
val array = arrayOf("a", "b", "c")  
for (i in 0 until array.size) {  
    print(array[i])  
}
```

Запомните, а лучше напишите код и запустите – если написать `in 0..array.size` то последний

элемент будет с индексом 4. Для этого есть ключевое слово `until` чтобы не выходить за рамки т.е. брать исключительно, а не включительно. Но если вам нужны индексы массива, то первый способ лучше. А если вам не нужен в цикле индекс, то можно написать как и в джава цикл `foreach`

```
val array = arrayOf("a", "b", "c")
for (item in array) {
    print(item)
}
```

Хорошо, а что если вам нужен как элемент, так и индекс? Можно написать цикл с индексом и брать элемент, но в котлин продумали более элегантный способ.

```
val array = arrayOf("a", "b", "c")
for ((index, value) in array.withIndex()) {
    println("индекс $index значение $value")
}
```

Мы создаем пару из индекса и значения и проходим по массиву с индексами. Заметьте как изящно в котлин решен вопрос с конкатенацией строк. Не нужно писать теперь “индекс “ + `index` + “ значение “ + `value`. Можно подставлять в строку значения из кода.

Что по поводу прерывания цикла то в котлин есть одна классная штука.

Например у вас есть цикл в цикле. И вы решаете его прервать, но написанный `break` прерывает лишь тот цикл, в котором он сам находится. Как прервать цикл снаружи? Нужно написать в булеан переменную что-то и тогда прервать цикл. Что-то типа этого

```
public static void main(String[] args) {
    for (int i = 0; i < 3; i++) {
        System.out.println("i: " + i);
        for (int j = 0; j < 3; j++) {
            System.out.println("j: " + j);
            if (j == 1) {
                break;
            }
        }
    }
}
```

JavaClass > main()

JavaClass x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java

i: 0
j: 0
j: 1
i: 1
j: 0
j: 1
i: 2
j: 0
j: 1

Process finished with exit code 0

Вот так в джава прерывание вложенного цикла. А теперь как прерываем цикл снаружи.



```
public static void main(String[] args) {
    boolean breakOuter = false;
    for (int i = 0; i < 3; i++) {
        if (breakOuter)
            break;
        System.out.println("i: " + i);
        for (int j = 0; j < 3; j++) {
            System.out.println("j: " + j);
            if (j == 1) {
                breakOuter = true;
                break;
            }
        }
    }
}
```

JavaClass > main()

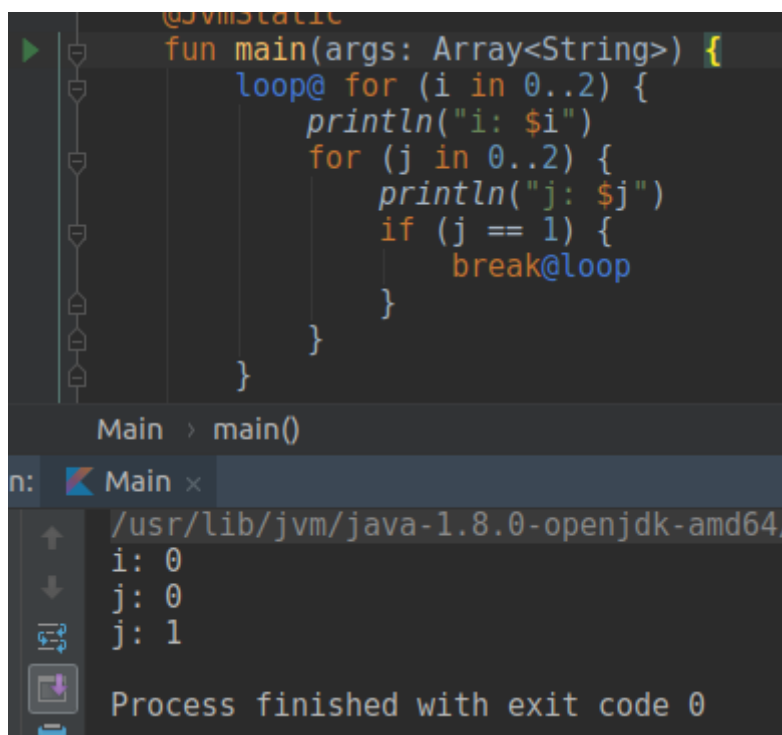
un: JavaClass x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

i: 0
j: 0
j: 1

Process finished with exit code 0

То же самое в котлин делается намного проще. Посмотрите на это.



```
@JvmStatic
fun main(args: Array<String>) {
    loop@ for (i in 0..2) {
        println("i: $i")
        for (j in 0..2) {
            println("j: $j")
            if (j == 1) {
                break@loop
            }
        }
    }
}
```

Main > main()

n: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/

i: 0
j: 0
j: 1

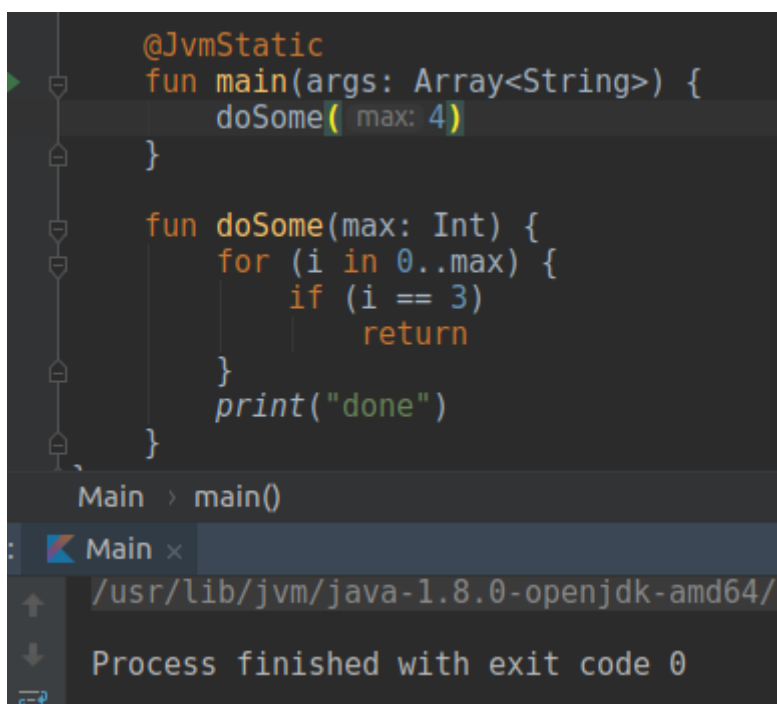
Process finished with exit code 0

Мы указываем какой цикл прервать через механизм аннотации.

Еще одна классная штука в циклах в котлин, что вы можете выйти из метода внутри цикла прервав его. Давайте объясню на примере. В джава мы бы написали такой метод

```
public void soSome(int max) {  
    boolean successful = true;  
    for (int i = 0; i < max; i++) {  
        if (i == 3) {  
            successful = false;  
            break;  
        }  
    }  
    if (successful) {  
        System.out.println("all done");  
    }  
}
```

Мы проходим по циклу и при каком-то условии выходим из цикла и если это произошло не должны продолжать работу метода совсем. Как видите опять лишняя булеан переменная. В котлин можно выйти из метода сразу.



```
@JvmStatic  
fun main(args: Array<String>) {  
    doSome(max: 4)  
}  
  
fun doSome(max: Int) {  
    for (i in 0..max) {  
        if (i == 3)  
            return  
    }  
    print("done")  
}
```

Main > main()

Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/t

Process finished with exit code 0

Опять же, зачем вам это знать- во-первых чтобы использовать. Не обязательно прерывать код с помощью выброса ошибки. Во-вторых чтобы знать и при чтении чужого кода понимать что происходит. Вам не нужно гадать : return выходит из цикла или из метода, вы теперь знаете наверняка.

Теперь, а что если вам нужно все же выйти из цикла? И чтобы читающий точно знал об этом сразу и не гадал. Для этого давайте опять же использовать аннотации лейблы для циклов

Но перед этим давайте я вам покажу как еще можно пройти циклом по массиву

```

fun doSome(max: Int, array: Array<Int>) {
    array.forEach { it: Int
        if (it == max)
            return@forEach
        }
    print("done")
}

```

Вместо того, чтобы писать `for (it in array)` можно вызвать метод `forEach` у самого массив (или списка) и внутри использовать `it`. Как видите теперь можно выходить из цикла с помощью `return` и однозначно обозначив что выходим из цикла. Посмотрите на вывод в консоль.

```

@JvmStatic
fun main(args: Array<String>) {
    doSome( max: 2, arrayOf(1, 2, 3, 4, 5, 6))
}

```

object Main
Main > main()

Process finished with exit code 0

Если бы мы выходили из метода, то на втором повторе вышли бы из него и не увидели бы логирования в консоль.

Что же касается цикла `while` то он точно такой же как и в джава. Ничего особенного там нет. За исключением если у вас 2 вложенных и вам надо прервать внешний, тогда пишете через лейбл аннотацию.

```

fun main(args: Array<String>) {
    var x = 0
    loop@ while (true) {
        println("main")
        while (true) {
            println("Inner")
            if (x == 2)
                break@loop
            x++
        }
    }
}

```

Main > main()

Process finished with exit code 0

2. Массивы

Рассмотрим пару интересных вещей относительно массивов. Вы можете с легкостью заполнять массив без использования цикла. Например вам нужен массив из чисел от 1 до 5. Это можно сделать вот так

```
@JvmStatic
fun main(args: Array<String>) {
    val array = Array( size: 5) { i -> i + 1 }
    array.forEach { print(it) }
}
```

Main > main()

Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java .
12345
Process finished with exit code 0

Когда мы создаем массив то задаем ему длину как и в джава, но потом можем задать значения с помощью лямбды – брать число и добавить к нему 1. Можете создать массив из квадратов просто перемножив *i* само на себя.

Или если вам нужен массив из 5 нулей, то просто делайте так

```
@JvmStatic
fun main(args: Array<String>) {
    val array = Array( size: 5) { 0 }
    array.forEach { print(it) }
}
```

Main

Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin
00000
Process finished with exit code 0

Но вы все так же можете заполнять массив как и в джава через индексы

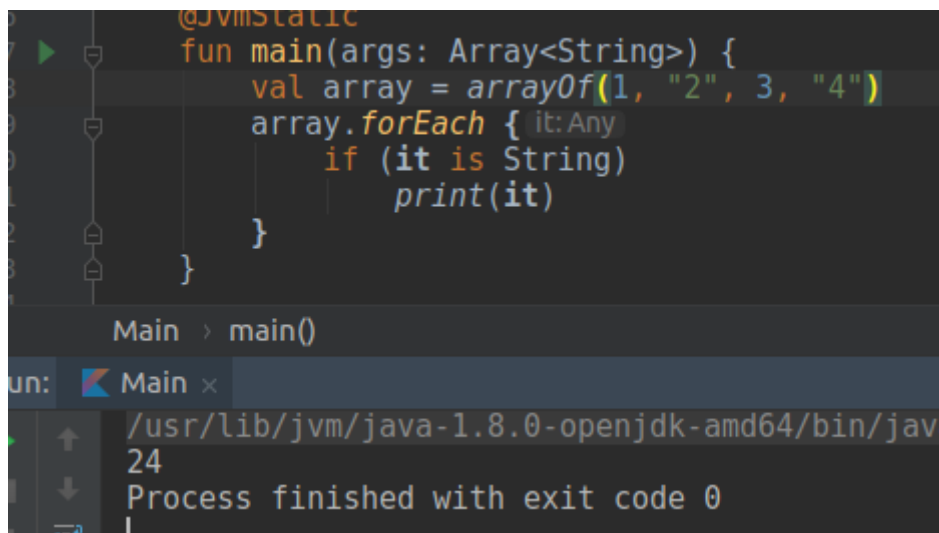
```
@JvmStatic
fun main(args: Array<String>) {
    val array = Array( size: 5) { 0 }
    array[2] = 1
    array.forEach { print(it) }
}
```

Main > main()

on: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/b
00100
Process finished with exit code 0

Еще одна забавная штука (которая на самом деле и в джава была) – массив любых видов, а не одного



```
@JvmStatic
fun main(args: Array<String>) {
    val array = arrayOf(1, "2", 3, "4")
    array.forEach { it: Any
        if (it is String)
            print(it)
    }
}
```

Main > main()

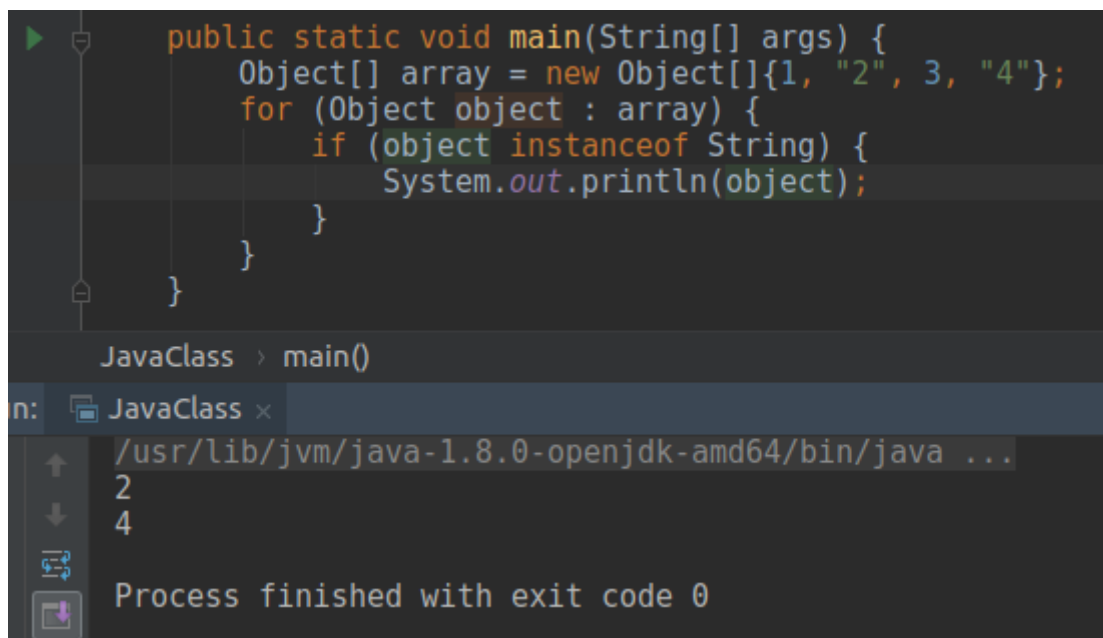
Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java

24

Process finished with exit code 0

В джава вы бы могли это сделать так



```
public static void main(String[] args) {
    Object[] array = new Object[]{1, "2", 3, "4"};
    for (Object object : array) {
        if (object instanceof String) {
            System.out.println(object);
        }
    }
}
```

JavaClass > main()

Run: JavaClass x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

2

4

Process finished with exit code 0

И еще одна функция у массивов в Котлин – добавить новый элемент и вернуть новый массив.



```
@JvmStatic
fun main(args: Array<String>) {
    val array = arrayOf(1, "2", 3, "4")
    array.plus(element: "new").forEach { it: Any
        if (it is String)
            print(it)
    }
}
```

Main

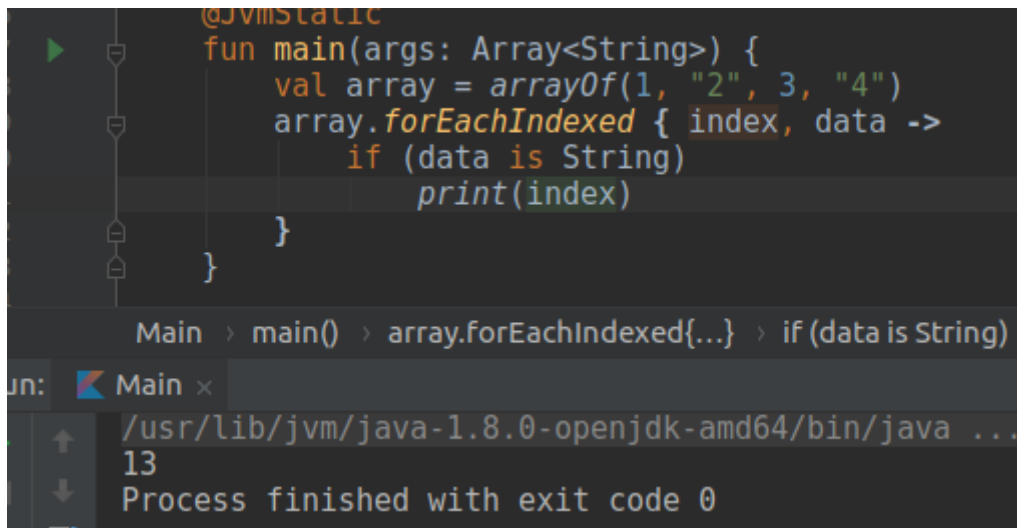
Run: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ..

24new

Process finished with exit code 0

И еще одно – можно пройти по массиву и получить как индекс так и сам элемент более простым способом. Если есть метод `forEach`, то должен быть метод `forEachIndexed`



```
@JvmStatic
fun main(args: Array<String>) {
    val array = arrayOf(1, "2", 3, "4")
    array.forEachIndexed { index, data ->
        if (data is String)
            print(index)
    }
}
```

Main > main() > array.forEachIndexed{...} > if (data is String)

un: Main x

/usr/lib/jvm/java-1.8.0-openjdk-amd64/bin/java ...

13

Process finished with exit code 0

Самое главное не путать где индекс, а где элемент. Для этого смотрите подсказки идеи когда пишете код.

Можете попрактиковаться в решении задач на циклы и массивы – просто найдите задачи в интернете (e.g. codingbat.com) если у нас не было таковых в разделе джава