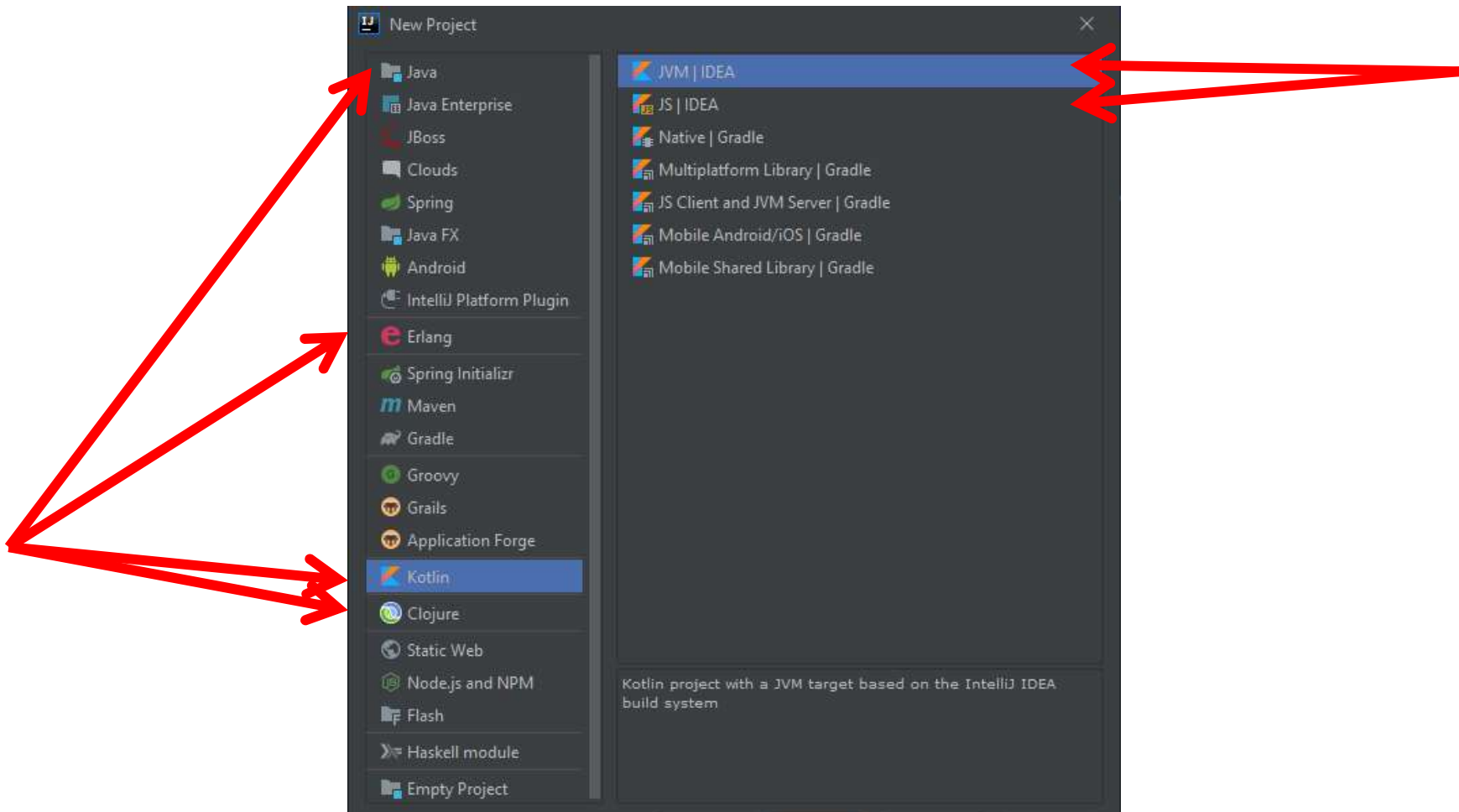
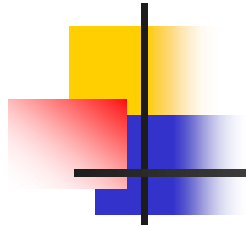


# Kotlin

Peter Borovanský, KAI, I-18, borovan(a)ii.fmph.uniba.sk





# Kotlin



## Modern Android development with Kotlin (September 2017) Part 1

It is really hard to find one project that covers all the things that are new in Android Development, so I decided to write one. In this article we will use the following:



<https://proandroiddev.com/modern-android-development-with-kotlin-september-2017-part-1-f976483f7bd6>

# Literatúra

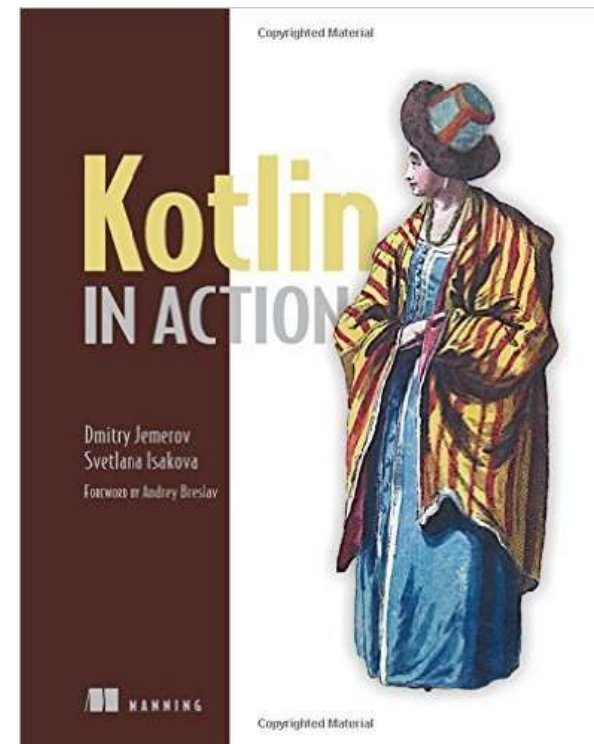
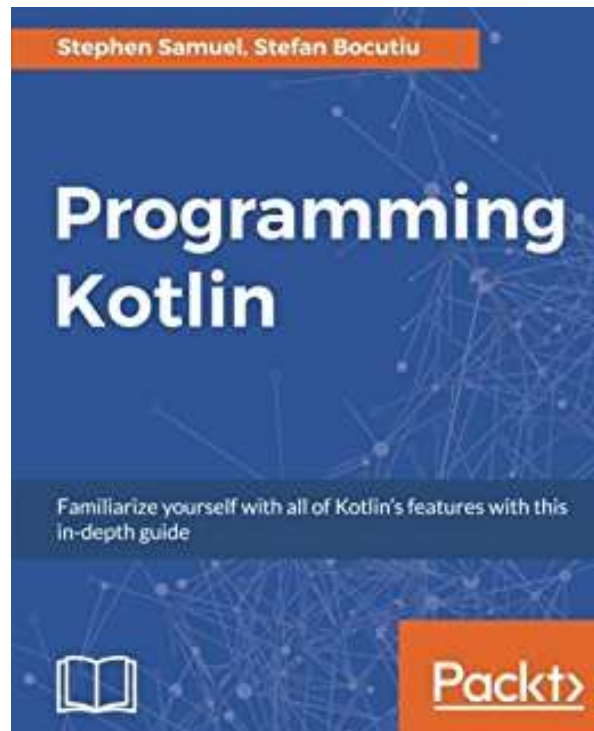


- Kotlin in Action

[https://github.com/panxl6/Kotlin-in-action/blob/master/ebook/Kotlin in Action v12 MEAP.pdf](https://github.com/panxl6/Kotlin-in-action/blob/master/ebook/Kotlin%20in%20Action%20v12%20MEAP.pdf)

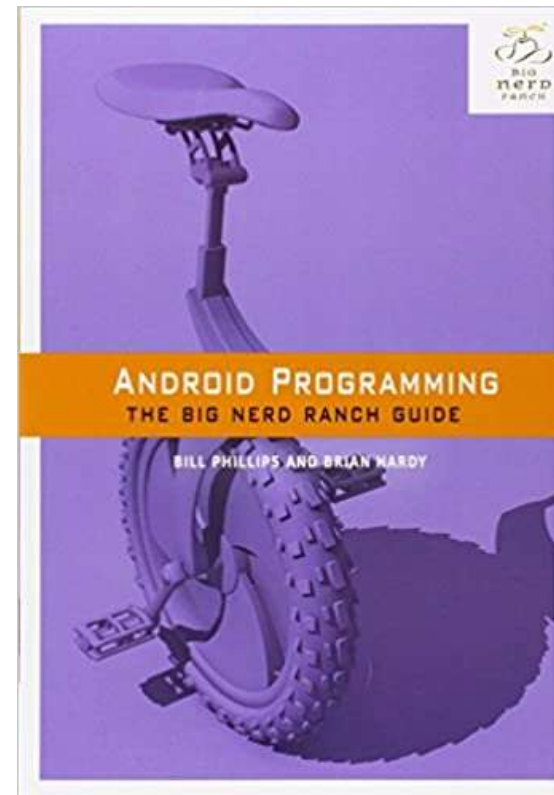
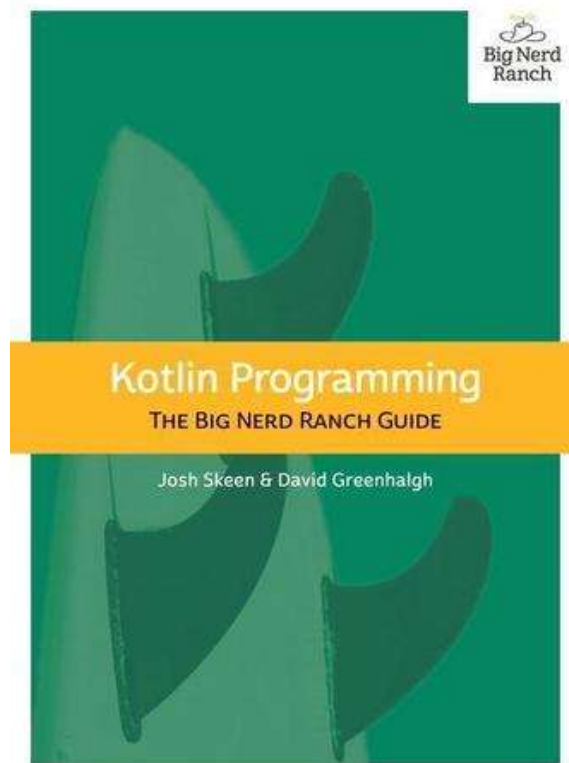
- Programming in Kotlin

<https://www.packtpub.com/application-development/programming-kotlin>



# Literatúra – for nerds

- Kotlin Programming – The Big Nerd Ranch Guide  
<https://www.megaknihy.sk/programovanie/20375234-kotlin-programming.html>
- Android Programming: The Big Nerd Ranch Guide (4th Edition)  
<https://www.bignerdranch.com/books/android-programming-the-big-nerd-ranch-guide-4th/>



# Literatúra



Swift is like Kotlin

- <https://kotlinlang.org/> Kotlin Playground (<https://play.kotlinlang.org/>)
- Swift is like Kotlin (<http://nilhcem.com/swift-is-like-kotlin/>)

## Swift

```
print("Hello, world!")
```

## Kotlin

```
println("Hello, world!")
```

prekladový slovník  
pre iOSákov

## Swift

```
var myVariable = 42  
myVariable = 50  
let myConstant = 42
```

## Kotlin

```
var myVariable = 42  
myVariable = 50  
val myConstant = 42
```

## Constants



# Kotlin Playground

<https://play.kotlinlang.org/>



https://try.kotlinlang.org/ is now obsolete

Kotlin in Action > Chapter 2 > 2.1 > 1\_HelloWorld.kt

Examples

Kotlin Koans 2/42

Kotlin in Action

Chapter 1

Chapter 2

2.1

1\_HelloWorld.kt

2\_Functions.kt

4\_1\_StringTemplate...

4\_2\_StringTemplate...

4\_3\_StringTemplate...

Save Save as Arguments

Program arguments

```
1 package ch02.ex1_1_HelloWorld
2
3 fun main(args: Array<String>) {
4     println("Hello, world!")
5 }
6
```

prog5 | Log out

Kotlin Koans > Collections > GroupBy

Examples

Kotlin Koans 22/42

Introduction

Conventions

Collections

Introduction

Filter map

All Any and other predi...

FlatMap

Max min

Sort

Sum

GroupBy

Cvičenie - 2

Pošli screenshot s Koans, dostaneš  
Math.floor(koans/14),  
resp.  
Math.floor(3\*% /100)

# Čo sa naučíte na play.kotlinlang.org

Progress: 30%

## ▼ Introduction

- ✓ Hello, world!
- ✓ Named arguments
- ✓ Default arguments
- ✓ Lambdas
- ✓ Strings
- ✓ Data classes
- ✓ Nullable types
- ✓ Smart casts
- ✓ Extension functions
- ✓ Object expressions
- ✓ SAM conversions
- ✓ [Extensions on collect](#)

## ► Introduction

## ▼ Conventions

- ✓ Comparison
- ✓ In range
- ✓ Range to
- ✓ For loop
- ✓ Operators overloading
- ✓ Destructuring declarat
- ✓ [Invoke](#)

Progress: 48%

## ► Introduction

## ► Conventions

## ▼ Collections

- ✓ Introduction
- ✓ Filter map
- ✓ All Any and other predicates
- ✓ FlatMap
- ✓ Max min
- ✓ Sort
- ✓ Sum
- ✓ GroupBy
- ✓ Partition
- ✓ Fold
- ✓ Compound tasks
- ✓ [Get used to new style](#)

TestShop.kt

Shop.kt

Progress: 78%

<https://play.kotlinlang.org/koans/>



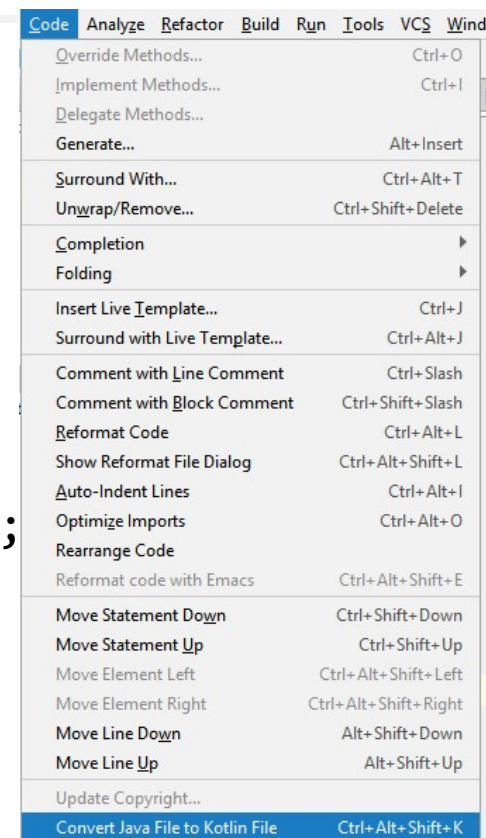
## Cvičenie - 2

Pošli screenshot s Koans, dostaneš  
Math.floor(koans/14),  
resp.  
Math.floor(3\*% /100)

# Java -> Kotlin

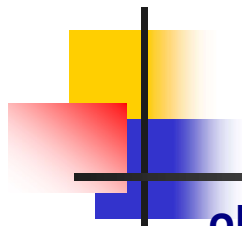
„klasický“ Java kód pre Fibonacciho s memoizáciou

```
public class fib {  
    static Integer[] table = new Integer[100];  
    private static int fib(int n) {  
        Integer result = table[n];  
        if (result == null) {  
            if (n < 2)  
                result = 1;  
            else  
                result = fib(n - 2) + fib(n - 1);  
            table[n] = result;  
        }  
        return result;  
    }  
    public static void main(String[] args) {  
        for(int i = 0; i<20; i++)  
            System.out.println("fib(" + i + ")=" + fib(i));  
    }  
}
```



Automatická konverzia do Kotlinu





# Java -> Kotlin

výsledok automatickej konverzie

Čo nás prekvapilo

```
object fib {  
    internal var table = arrayOfNulls<Int>(100)  
    private fun fib(n: Int): Int {  
        var result: Int? = table[n]  
        if (result == null) {  
            if (n < 2)  
                result = 1  
            else  
                result = fib(n - 2) + fib(n - 1)  
            table[n] = result  
        }  
        return result  
    }  
    @JvmStatic fun main(args: Array<String>) {  
        for (i in 0..19)  
            println("fib(" + i + ")=" + fib(i))  
    }  
}
```

Už nenájdete pôvodný zdroják

DÚ podobne vygenerované sa neuznajú



# if je výraz

---

- if je výraz

```
fun binCifSum(n : Int) : Int =  
    if (n <= 0) 0  
    else binCifSum(n/2) + if (n % 2 == 0) 0 else 1  
    else binCifSum(n/2) + (n % 2 == 0)
```

```
fun binCifSumClassic(n : Int) : Int {  
    if (n <= 0) return 0  
    else if (n % 2 == 0) return binCifSumClassic(n / 2)  
    else return 1 + binCifSumClassic(n / 2)  
}
```

```
fun main(args:Array<String>) : Unit {  
    for (n in 0..10)  
        println("binCifSum $n je ${binCifSum(n)}")  
}
```



# when je switch, tiež je to výraz

```
val kategoria =  
    if (vek < 6) "predskolsky"  
    else if (vek <= 11) "1.stupen"  
    else if (vek <= 18) "2.stupen"  
    else "mimo"  
  
val kategoria1 =  
    when (vek) {  
        in 0..5 -> "predskolsky"  
        in 5..11 -> "1.stupen"  
        in 12..18 -> "2.stupen"  
        else -> "mimo"  
    }  
  
var kategoria2 = "mimo"  
when (vek) {  
    in 0..5 -> kategoria2 = "predskolsky"  
    in 5..11 -> kategoria2 = "1.stupen"  
    in 12..18 -> kategoria2 = "2.stupen"  
}
```



# For/foreach cyklus

```
for (x in 1..10) println(x)                // 1, 2, ..., 10
for (x in (1..10).toList()) println(x)     // 1, 2, ..., 10
for (x in (10 downTo 1).toList()) println(x) // 10, 9, ..., 1
for (x in 10 downTo 1) println(x)          // 10, 9, ..., 1
for (x in 1 until 10) println(x)           // 1, 2, ..., 9
for (x in 1 until 10 step 2) println(x)    // 1, 3, 5, 7, 9
for (x in listOf(2,3,5,7,11,13)) println(x)

for (x in 'a'..'z') println(x)             // a, b, ..., z
for ((index, value) in ('a'..'z').withIndex())
    println("[\$index]=\$value")            // [0]=a, [1]=b,...

val map=mapOf(1 to "gula",2 to "zelen",3 to "zalud",4 to"srdce")
for ((key, value) in map) println("[\$key]=\$value")
// [1]=gula, [2]=zelen, [3]=zalud, [4]=srdce
```



```
fun main(args: Array<String>) {  
    for(a in args)  
        print("$a, ")  
  
    for (c in 'A'..'F')  
        println(Integer.toBinaryString(c.toInt()))  
  
    for (c in ' '..'z')  
        if (c in 'a'..'z' || c in 'A'..'Z')  
            print(c)  
  
    for (c in ' '..'z')  
        when (c) {  
            in '0'..'9' -> println("digit")  
            in 'a'..'z', in 'A'..'Z' -> println("letter")  
        }  
}
```





# Operátory porovnania

---

- podobne ako Java <=, <, >=, >, !=

ale

== je porovnanie hodnôt

=== je porovnanie referencií

```
val a = "kot"  
val b = "lin"  
val c = (a+b).trim()  
val d = "kotlin"  
println("c==d ${c==d}, c===d ${c===d}")
```

c==d true, c===d false



# Kolekcje

---

```
val set = hashSetOf(2, 3, 5, 7, 11, 13, 17)
val list = arrayListOf(-1, 0, 1)
val map = hashMapOf("sedma" to 7, "osma" to 8, "dolnik" to 11,
                    "hornik" to 12, "kral" to 13, "eso" to 15)
```

```
println(set)    println(set.javaClass)
println(list)   println(list.javaClass)
println(map)    println(map.javaClass)
```

```
for(x in list)                // cyklus cez list
    for(y in set)              // cyklus cez set
        for((key, value) in map) // cyklus cez map
            println("$x $y $key $value")
```



# Číselné funkcie, String template

```
fun fib(n: Int): Int {  
    return if (n < 2) 1 else fib(n-1) + fib(n-2)  
}  
  
fun fib1(n: Int): Int {  
    fun fib(n: Int, a : Int = 0, b : Int = 1): Int {  
        return if (n < 0) a else fib(n-1, b, a+b)  
    }  
    return fib(n)  
}  
  
fun main(args: Array<String>) {  
    val lst = listOf(1,2,3,4,5,6,7,8,9,10)  
    println(lst.map { n -> fib(n) })  
    println(lst.map { fib1(it) })  
    lst.forEach { println("fib($it) = ${fib1(it)}") }  
    for(i in 1..11) println("fib($i) = ${fib1(i)}")  
    println("Maximum: ${lst.map { fib(it) }.max()}")  
}
```



# Funkcie

---

```
val fcia = { x:Int, y : Int -> println("sucet $x+$y"); x+y}  
val proc = { x:Int, y : Int -> println("sucet $x+$y")}
```

```
println(fcia(12,7))  
proc(13,9)  
println({ x:Int -> x+1 }(2))  
; // inak neopochopí, že nejde o blok, ale lambda konštantu  
{ x:Int -> println(x)}(4)  
    // preto jasnejší zápis  
run {{ x:Int -> println(x)}(4)}
```

```
val delta = 5  
println(listOf(1,2,3)  
    .map { it + delta}    // x -> x + delta, clojure  
    .filter {it % 2 == 0} )
```

# Addams Kotlin family



```
data class Person(val first : String, val name: String,  
                  val age: Int? = null,  
                  val father : Person?, val mother : Person?)
```

Data class je class s predgenerovanými equals, hashCode, toString, copy

```
fun main(args: Array<String>) {  
    val father = Person("Gomez", "Addams", 156, null, null)  
    val mother = Person("Morticia", "Addams", 136, null, null)  
    val daughter = Person("Wednesday", "Addams", 46, father, mother)  
    val son = Person("Pugsley", "Addams", 36, father, mother)  
    val family = listOf( father, mother, daughter, son,  
                          Person("Fester", "Addams", 174, null, null), // uncle  
                          Person("Pubert", "Addams", null, null, null) // on the picture  
                        )  
    val oldest = family.maxBy { it.age ?: 0 }  
    println("The oldest is: $oldest")  
}
```





# Funkcie

---

```
println(family.map { it.first }) // mapToObj
println(family.filter { it.age?:0 > 100 } )
println(family.all { it.age?:0 < 100 } )
println(family.all { it.name == "Dracula" } )
println(family.groupBy { it.father } )
println(family.filter {
    it.age == family.maxBy { person: Person -> person.age?:0 }?:0 } )
```

Ak by .age bol Int, nie Int?

```
it.age == family.maxBy { person: Person -> person.age }?:0 } )
```

```
val numbers = mapOf(0 to "zero", 1 to "one")
for((father, persons) in family.groupBy { it.father })
    println("${persons.size} ma otca $father")
```

```
println(listOf("a", "aba", "b", "ba", "abba").groupBy { it.length })
println(listOf("a", "aba", "b", "ba", "abba").flatMap { it.toList() })
```



# Funkcie

---

```
class Book(val title: String, val authors: List<String>)
val books = listOf(
    Book("Action in Kotlin", listOf("Dmitry Jemerov", "Svetlana Isakova")),
    Book("Mort", listOf("Terry Pratchett")),
    Book("Good Omens", listOf("Terry Pratchett", "Neil Gaiman")),
    Book("Discworld", listOf("Terry Pratchett", "Paul Kidby")))
println(books.flatMap { it.authors }.toSet())

listOf(1, 2, 3, 4)
    .asSequence()
    .map { print("map($it) "); it * it }
    .filter { print("filter($it) "); it % 2 == 0 }
    .toList()

val nats = generateSequence(1) { it + 1 }
println(nats.takeWhile { it <= 100 }.sum())
println(nats.takeWhile { it <= 10 }.reduce({ x:Int, y : Int -> x*y}))
```



# Break point

pokračovanie niekedy na budúce

---