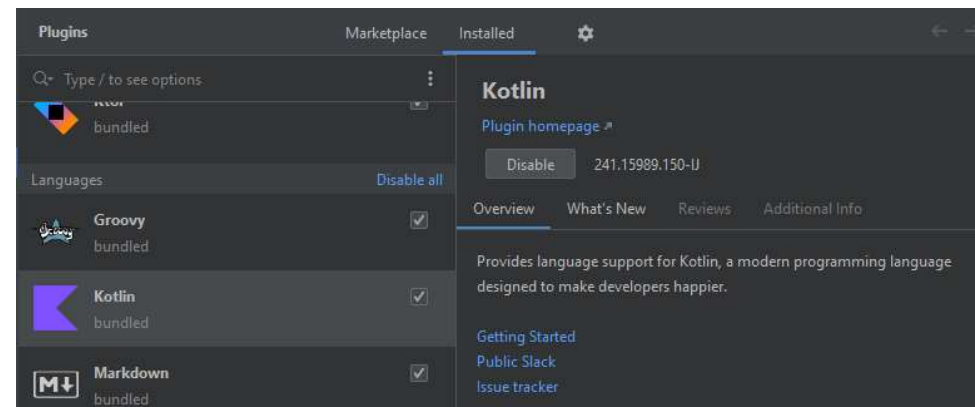
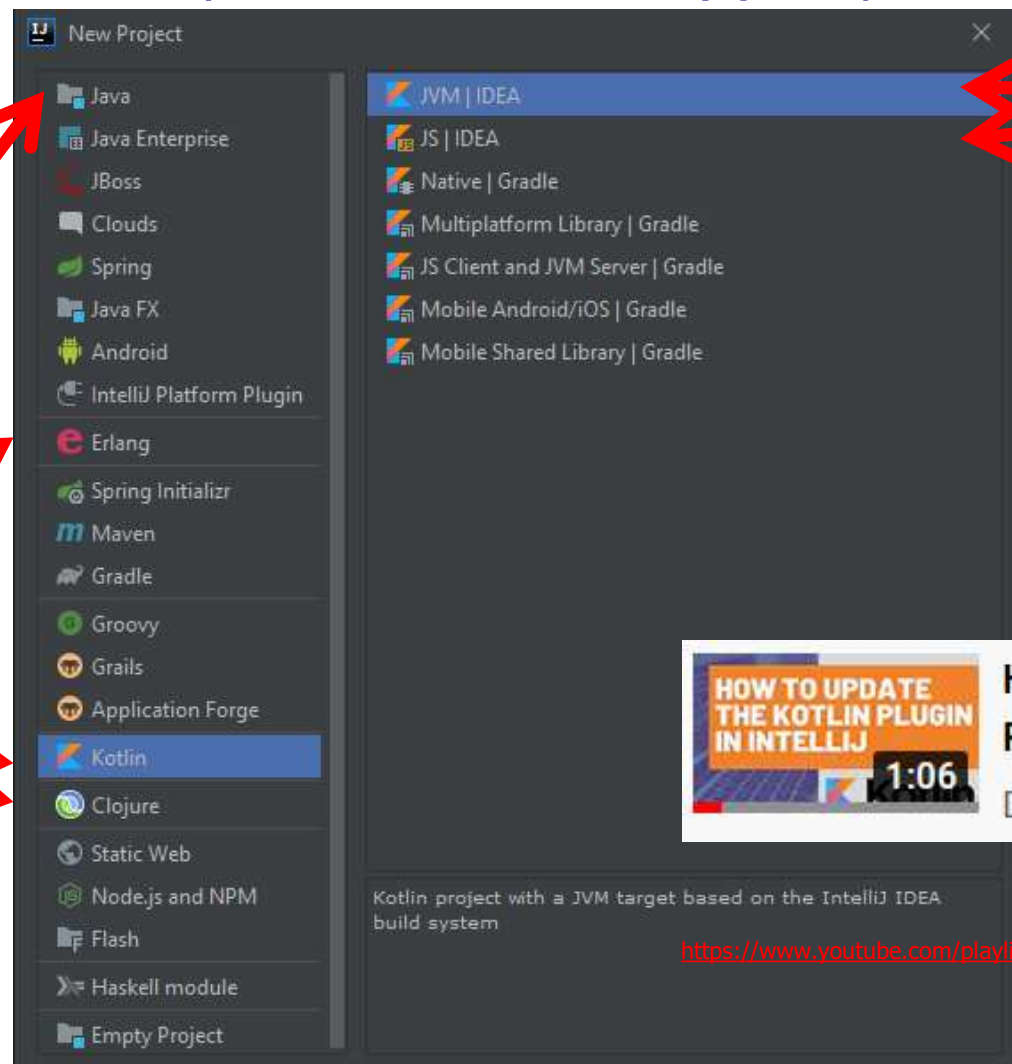


# Kotlin



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



Kotlin Plugin in  
IntelliJ

File/Settings/Plugins  
MarketPlace/Kotlin

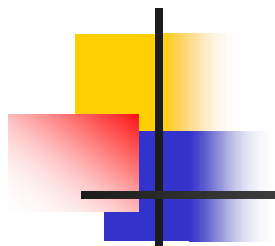
Plugin od JetBrains



How to Update the Kotlin  
Plugin in IntelliJ

Donn Felker - Freelancing for ...

<https://www.youtube.com/playlist?list=PLVUm4IewkTXqwzuRXZisWg7shMTiQhUtz>



# Kotlin



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

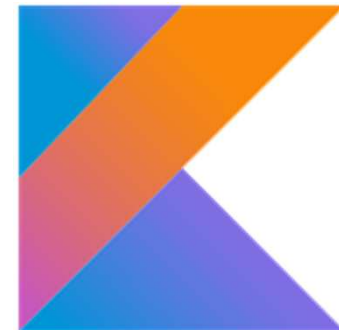
It is really hard to find one project that covers all the things that are new in Android Development, so I decided to write one. In this article we will use the following:



Rýchly nadhľad  
nad vlastnosťami  
jazyka Kotlin, dotyk  
s prvými aplikáciami

# Literatúra

serióznejšie čítanie

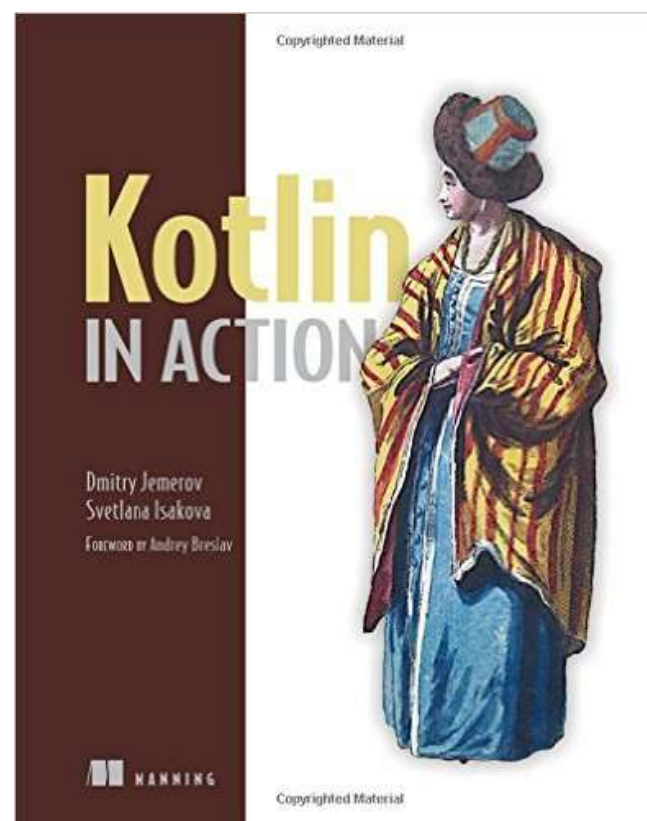
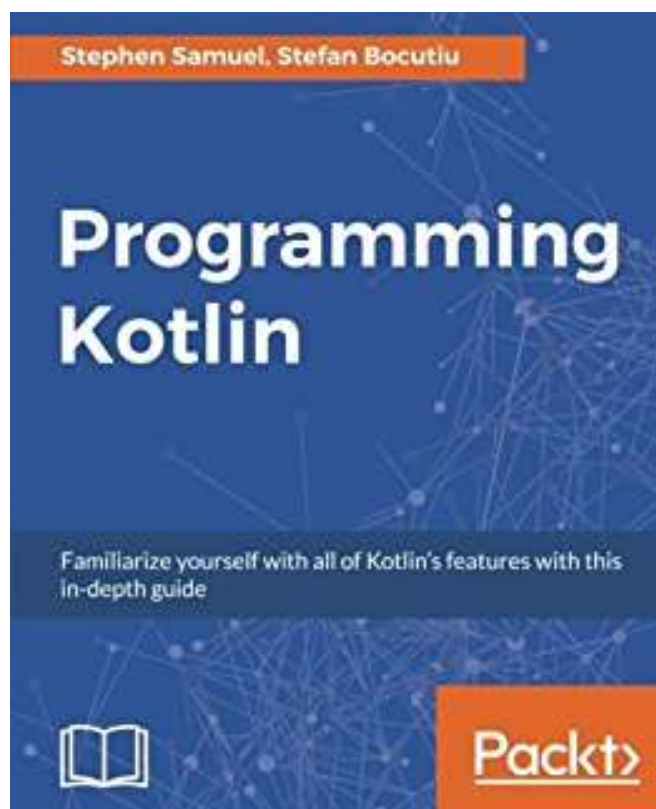


- Kotlin in Action

<https://www.manning.com/books/kotlin-in-action>

- Programming in Kotlin

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

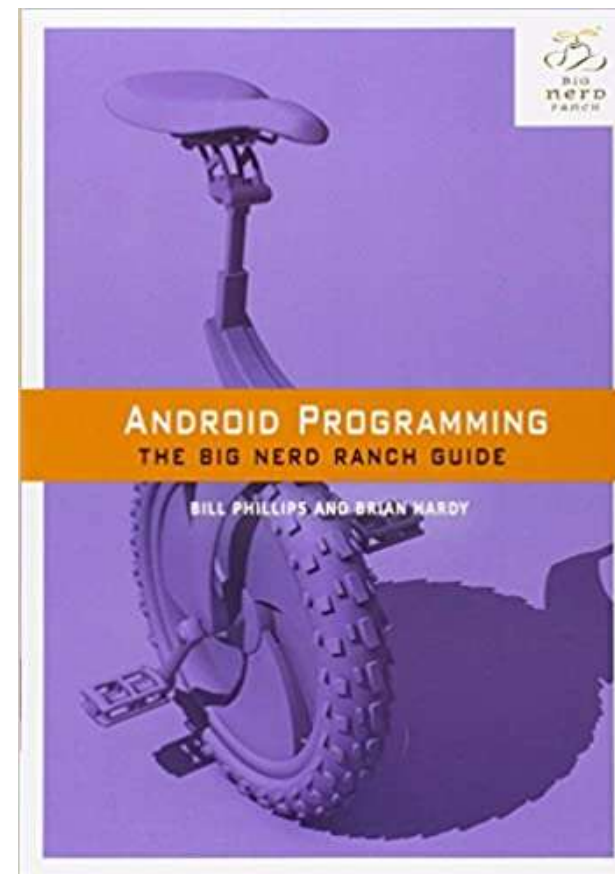
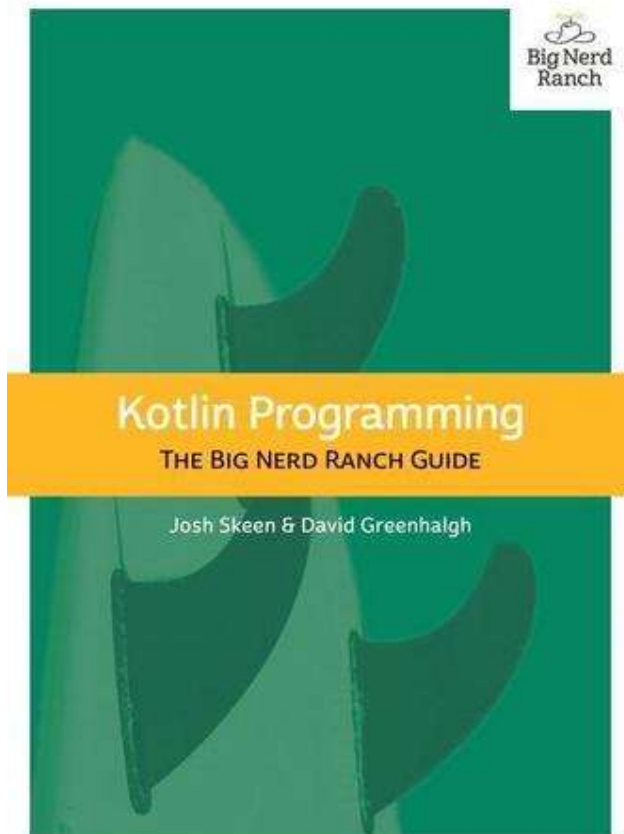


# Literatúra

for nerds



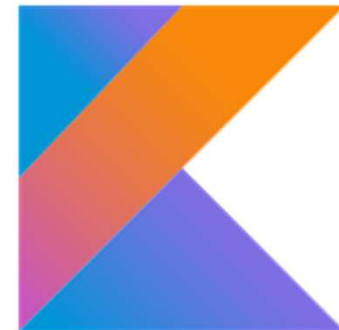
- Kotlin Programming – The Big Nerd Ranch Guide  
<https://www.amazon.com/Kotlin-Programming-Nerd-Ranch-Guide/dp/0135161630>
- Android Programming: The Big Nerd Ranch Guide (4th Edition)  
<https://www.bignerdranch.com/books/android-programming-the-big-nerd-ranch-guide-4th/>





# Literatúra

nežný úvod

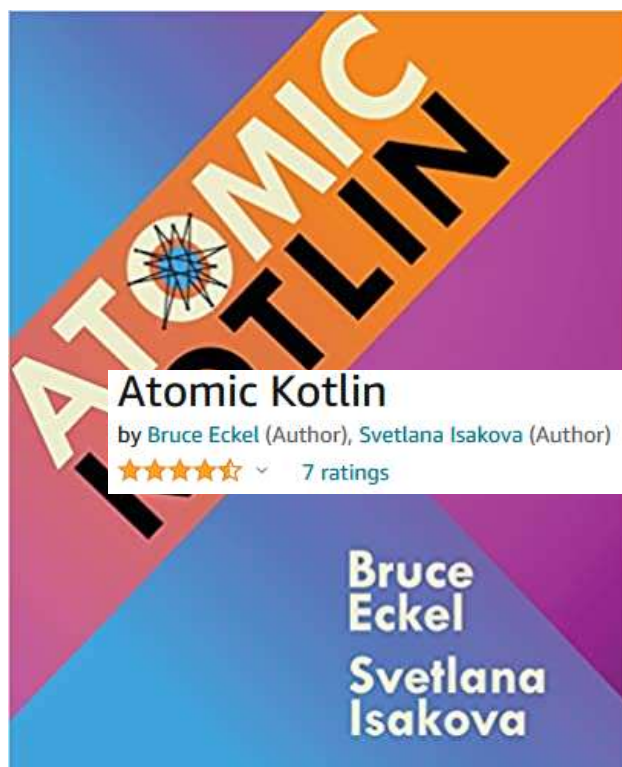


Bruce Eckel, Svetlana Isakova: Atomic Kotlin - ideálne pre začiatočníkov

<https://www.amazon.com/Atomic-Kotlin-Bruce-Eckel/dp/0981872557>

Marcin Moskala: Effective Kotlin – Best Practices - ideálne pre pokročilejších

<https://www.amazon.com/Effective-Kotlin-practices-Marcin-Moskala/dp/8395452837>



## Individual training

If you are an individual looking to upgrade your skill set, with our workshops you'll have an incredible chance to do it.

Registration for the best Kotlin **OPEN WORKSHOPS** with **Marcin Moskala** is already open: 🗨️



### Kotlin Coroutines

A workshop focused on advanced practical skills like generics, reflection, annotation processing, and KSP, practiced on implementing projects like custom mocking library, object serialization, dependency injection.

**Dates:** 26-27th of October 2023

**Times:** 9:00-17:00 UTC+2

**Fee:** 300 euros

[More Info](#)



### Kotlin for Developers

Focused on the Kotlin JVM ecosystem, the training prepares for general programming and backend development (e.g. in Spring and Ktor).

**Dates:** 22-24th of November 2023

**Times:** 9:00-17:00 UTC+1

**Fee:** 400 euros

[More Info](#)

# Literatúra

ideálne pre „youtuberov“



<https://www.youtube.com/playlist?list=PLVUm4IewkTXqwzuRXZisWg7shM>



Search



## The Kotlin Programming Language Course for Beginners

134 videos • 32,965 views • Last updated on 19 Mar 2021



In this course, you will learn the Kotlin programming language from the ground up. Over 9 hours of content, 130+ lessons.

This playlist contains all 134 lessons. If you prefer to watch this as a single 9+ hour-long single video, you can do so here:

<https://www.youtube.com/watch?v=wuiT4...>

Topics include, but are not limited to:

- 1 **HOW TO INSTALL THE INTELLIJ IDE FOR KOTLIN DEVELOPMENT**  
Donn Felker - Freelancing for Software Developers
- 2 **HOW TO CREATE A KOTLIN FILE IN INTELLIJ**  
Donn Felker - Freelancing for Software Developers
- 3 **HOW TO UPDATE THE KOTLIN PLUGIN IN INTELLIJ**  
Donn Felker - Freelancing for Software Developers
- 4 **"Hello World" in Kotlin - Your First Kotlin Program**  
Donn Felker - Freelancing for Software Developers
- 5 **HOW TO CREATE VARIABLES IN KOTLIN**  
Donn Felker - Freelancing for Software Developers
- 6 **HOW TO CREATE READ ONLY VARIABLES**  
Donn Felker - Freelancing for Software Developers

How to Update the Kotlin Plugin in IntelliJ



# Android Studio Essentials -



Developing Android  
Apps Using Android Studio  
2022.3.1 and Kotlin,

Neil Smyth



|   |     |
|---|-----|
| 11. An Introduction to Kotlin .....                     | 91  |
| 11.1 What is Kotlin? .....                              | 91  |
| 11.2 Kotlin and Java .....                              | 91  |
| 11.3 Converting from Java to Kotlin .....               | 91  |
| 11.4 Kotlin and Android Studio .....                    | 92  |
| 11.5 Experimenting with Kotlin .....                    | 92  |
| 11.6 Semi-colons in Kotlin .....                        | 93  |
| 11.7 Summary .....                                      | 93  |
| 12. Kotlin Data Types, Variables, and Nullability ..... | 95  |
| 12.1 Kotlin Data Types .....                            | 95  |
| 12.1.1 Integer Data Types .....                         | 96  |
| 12.1.2 Floating-Point Data Types .....                  | 96  |
| 12.1.3 Boolean Data Type .....                          | 96  |
| 12.1.4 Character Data Type .....                        | 96  |
| 12.1.5 String Data Type .....                           | 96  |
| 12.1.6 Escape Sequences .....                           | 97  |
| 12.2 Mutable Variables .....                            | 98  |
| 12.3 Immutable Variables .....                          | 98  |
| 12.4 Declaring Mutable and Immutable Variables .....    | 98  |
| 12.5 Data Types are Objects .....                       | 98  |
| 12.6 Type Annotations and Type Inference .....          | 99  |
| 12.7 Nullable Type .....                                | 100 |
| 12.8 The Safe Call Operator .....                       | 100 |
| 12.9 Not-Null Assertion .....                           | 101 |
| 12.10 Nullable Types and the let Function .....         | 101 |
| 12.11 Late Initialization (lateinit) .....              | 102 |
| 12.12 The Elvis Operator .....                          | 103 |
| 12.13 Type Casting and Type Checking .....              | 103 |
| 12.14 Summary .....                                     | 104 |
| 13. Kotlin Operators and Expressions .....              | 105 |
| 13.1 Expression Syntax in Kotlin .....                  | 105 |
| 13.2 The Basic Assignment Operator .....                | 105 |
| 13.3 Kotlin Arithmetic Operators .....                  | 105 |
| 13.4 Augmented Assignment Operators .....               | 106 |
| 13.5 Increment and Decrement Operators .....            | 106 |
| 13.6 Equality Operators .....                           | 107 |
| 13.7 Boolean Logical Operators .....                    | 107 |
| 13.8 Range Operator .....                               | 108 |
| 13.9 Bitwise Operators .....                            | 108 |
| 13.9.1 Bitwise Inversion .....                          | 108 |
| 13.9.2 Bitwise AND .....                                | 109 |
| 13.9.3 Bitwise OR .....                                 | 109 |
| 13.9.4 Bitwise XOR .....                                | 109 |
| 13.9.5 Bitwise Left Shift .....                         | 110 |
| 13.9.6 Bitwise Right Shift .....                        | 110 |
| 13.10 Summary .....                                     | 111 |
| 14. Kotlin Control Flow .....                           | 113 |
| 14.1 Looping Control flow .....                         | 113 |
| 14.1.1 The Kotlin <i>for-in</i> Statement .....         | 113 |
| 14.1.2 The <i>while</i> Loop .....                      | 114 |
| 14.1.3 The <i>do ... while</i> loop .....               | 115 |

# Kotlin vs. Swift



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

## Constants

## Swift

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

## Kotlin

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



# Kotlin Playground

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



Home ⏪ ⓘ 🔒 [https://try.kotl.in/#/Kotlin in Action/Chapter 2/2.1/1\\_HelloWorld.kt](https://try.kotl.in/#/Kotlin%20in%20Action/Chapter%202/2.1/1_HelloWorld.kt)

🐦 prog5 | Log out

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 - 1

Pošli screenshot s Koans, dostaneš  
 $\text{Math.floor}(3 * \% / 100)$

# Čo sa naučíte na play.kotlinlang.org

## ▼ Introduction

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



## ► 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/>



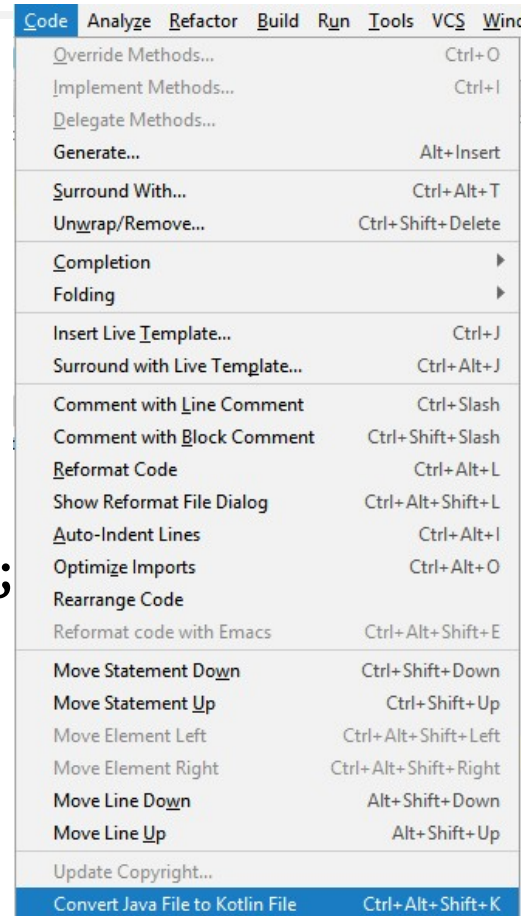
## Cvičenie - 1

Pošli screenshot s Koans, dostaneš  
`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

# Kotlinish verzia

```
import java.math.BigInteger

val table = mutableMapOf<Int, BigInteger>() // HashMap

fun fib(n: Int): BigInteger = table.getOrPut(n) {
    if (n <= 2)
        BigInteger.ONE
    else
        fib(n - 1) + fib(n - 2)
}

fun main() {
    println(fib(1024))
}
```



fibonacci 1024

NATURAL LANGUAGE MATH INPUT

EXTENDED KEYBOARD EXAMPLES UPLOAD RANDOM

4 506 699 633 677 819 813 104 383 235 728 886 049 367 860 596 218 604 830 803 023`  
149 600 030 645 708 721 396 248 792 609 141 030 396 244 873 266 580 345 011 219`  
530 209 367 425 581 019 871 067 646 094 200 262 285 202 346 655 868 899 711 089`  
246 778 413 354 004 103 631 553 925 405 243

Decimal approximation

4.5066996336778198131043832357288860493678605962186048308030... × 10<sup>213</sup>

More digits



# 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
```



# Cykly

---

```
fun main(args: Array<String>) {  
    for(a in args)  
        print("$a, ")  
  
    for (c in 'A'..'F')  
        println(Integer.toString(c.code()))  
  
    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)
```

```
[17, 2, 3, 5, 7, 11, 13]
class java.util.HashSet
[-1, 0, 1]
class java.util.ArrayList
{kral=13, eso=15, sedma=7, osma=8,
 dolnik=11, hornik=12}
class java.util.HashMap
```

```
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")
```

# Kotlin Notebook

The screenshot displays the IntelliJ IDEA interface with a Kotlin Notebook open. The notebook contains a Kotlin code snippet that creates a set, a list, and a map, and then iterates over the list to print elements from the set and map. The code is as follows:

```
In 5 1 val set = HashSet(2, 3, 5, 7, 11, 13, 17)
2 val list = ArrayListOf(-1, 0, 1)
3 val map = HashMapOf("sedma" to 7, "osma" to 8, "dolnik" to 11,
4 "hornik" to 12, "kral" to 13, "eso" to 15)
5 println(set); println(set.javaClass)
6 println(list); println(list.javaClass)
7 println(map); println(map.javaClass)
8
9 for(x in list)
10     for(y in set)
11         for((key, value) in map)
12             println("$x $y $key $value")
13
```

The output of the code execution is shown below:

```
1 2 osma 8
1 2 dolnik 11
1 2 hornik 12
1 3 kral 13
1 3 eso 15
1 3 sedma 7
1 3 osma 8
1 3 dolnik 11
1 3 hornik 12
1 5 kral 13
1 5 eso 15
1 5 sedma 7
1 5 osma 8
1 5 dolnik 11
1 5 hornik 12
```

The IDE interface also shows a project view on the left with various Kotlin files, a run console at the bottom, and a menu bar at the top with options like 'New', 'Cut', 'Copy', 'Paste', 'Find Usages', 'Find in Files...', 'Replace in Files...', 'Analyze', 'Refactor', and 'Clean Python Compiled Files'. The status bar at the bottom indicates the current file is 'KotlinNotebook.ipynb' and the encoding is 'UTF-8'.

# 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 a it

---

```
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")
}
```



# Funkcionály

---

```
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: APerson -> person.age ?: 0 }.age?: 0) } )
println(family.filter { it.age ==
    (family.map { it.age?:0 }.max()) } )
```

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() } 1.kt
```



# 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}))
```



# Collection vs. sequence

---

```
val collection = (-100..100)
  .filter {it % 2 == 0}
  .map { it * 2 }
  .map { it/it }
  .take(10)
println(collection)
java.lang.ArithmeticException
```

## Kolekcie:

- vyhodnocujú sa dravo -eager
- každá transformácia sa aplikuje na celú kolekciu
- vytvorí sa nová kolekcia
- dobré pre nevelké kolekcie

```
val sequence = (-100..100)
  .asSequence()
  .filter {it % 2 == 0}
  .map { it * 2 }
  .map { it/it }
  .take(10)
println(sequence.toList())
[1, 1, 1, 1, 1, 1, 1, 1, 1, 1]
```

## Sekvencie:

- vyhodnocujú sa lenivo -lazy
- každá transformácia sa aplikuje element-po-elemente
- nevytvorí sa nová kolekcia
- vhodné pre veľké kolekcie