Kotlin for Java Developers

what every java developer should know about kotlin
Today I would like to share with you

- Why I care about Kotlin?
- 20 features I love
- 10 features you need to be aware of
- Ecosystem (platforms, tools, documentation, courses)
Next JVM language?

- Statically typed programming language for multi platform applications
  - concise
  - safe
  - interoperable with Java
  - built together with tooling support
  - open source under Apache 2.0 license
- Version 1.0 released in 2015, current version 1.1.2-2 (April 2017)
- Google announced first-class support for Kotlin on Android
- Pivotal will introduce Kotlin support in Spring Framework 5.0
- Easy to learn if you know Java
Kotlin basics

class Foo(val x : String, var y: Int) {
    fun sayHello() = print("Hello")
}

fun main(args: Array<String>) : Unit {
    print("Hello world!")

    val foo = Foo("JUG ZG", 14)
    foo.sayHello()
}
Features I love
Extension functions and properties

```kotlin
import java.io.File
import java.math.BigDecimal

fun Int.toBigDecimal() = BigDecimal.valueOf(this.toLong())

val Int.bd : BigDecimal
    get() = BigDecimal.valueOf(this.toLong())

fun String.reverse() = StringBuilder(this).reverse().toString()

fun main(args: Array<String>) {
    print(1.bd + 2.toBigDecimal())
    print("String to reverse").reverse()
    val lines = File("test").bufferedReader().lines()
    print(lines)
}
```
Data classes

- equals() / hashCode() pair
- toString() “Account[email=test]”
- componentN() functions in their order of declaration
- copy() function

```java
import java.util.Objects;

public class AccountJava {
    private String email;

    public AccountJava(String email) {
        this.email = email;
    }

    public String getEmail() {
        return email;
    }

    public void setEmail(String email) {
        this.email = email;
    }

    @Override
    public boolean equals(Object o) {
        if (this == o)
            return true;
        if (o == null || getClass() != o.getClass())
            return false;
        AccountJava that = (AccountJava) o;
        return Objects.equals(email, that.email);
    }

    @Override
    public int hashCode() {
        return Objects.hash(email);
    }

    @Override
    public String toString() {
        return "AccountJava\n\n            email=\n\n        \" + email + \"";
    }
}
```
String templates

```kotlin
fun main(args: Array<String>) {
    val name = "JUG Zielona Góra!"
    val message = "Hello: $name"
    print(message)

    val s = "abc"
    print("String length is ${if (args.isNotEmpty()) args[0].length else s.length}"))
```
Null safety

- Types defines nullability
  - Platform types
- Safe calls
- Elvis operator
- !!. operator
- Safe casts

```kotlin
fun main(args: Array<String>) {
    // Non null type
    var a: String = "abc"
    //a = null // compilation error

    // Nullable type
    var b: String? = "abc"
    b = null // ok

    // Safe calls
    val l = if (b != null) b.length else -1
    val b?.length

    // Elvis operator
    val account: Account? = Account("email", null)
    print(account?.externalId ?: "Not a number")

    // The !! operator
    b!!.length

    // Safe casts - return null if cast was not successful
    val aInt: Int? = a as? Int
}
```
fun getScore(score: Int) = when (score) {
    9, 10 -> "Excellent"
    in 6..8 -> "Good"
    4, 5 -> "Ok"
    in 1..3 -> "Fail"
    else -> "Fail"
}

fun whenShowcase(x : Any?) {
    val validNumbers = listOf(21, 22, 23)
    val result = when (x) {
        0, 1 -> "x == 0 or x == 1"
        is String -> x.startsWith("prefix")
        in 1..10 -> "x is in the range"
        in validNumbers -> "x is valid"
        !in 10..20 -> {
            "x is outside the range"
        }
        else -> "Else branch"
    } print(result)
}

public class ScoreSwitch {
    String getScore(int score) {
        String grade;
        switch (score) {
            case 10:
                grade = "Excellent";
                break;
            case 9:
                grade = "Excellent";
                break;
            case 8:
            case 7:
            case 6:
                grade = "Good";
                break;
            case 5:
            case 4:
                grade = "Ok";
                break;
            case 3:
            case 2:
            case 1:
                grade = "Fail";
                break;
            default:
                grade = "Fail";
        }
        return grade;
    }
}
Operator overloading

- +, -, *, /, %, ...
  - a + b -> \texttt{a.plus(b)}
  - a..b -> \texttt{a.rangeTo(b)}
- \texttt{in}, \texttt{!in}
  - a.contains(b)
- Indexed access [ ]
  - a[i] -> \texttt{a.get(i)}
  - a[i] = b -> \texttt{a.set(i, b)}
- Invoke
  - a(i, j) -> \texttt{a.invoke(i, j)}
- a == b
  - a?.equals(b) ?: (b === null)
- a > b, a < b, a >= b, a <= b -> \texttt{a.compareTo(b)}
Operator overloading (1)

```scala
// + on BigDecimal
println(123.bd + 234.bd)

// []
val list = array0f(1, 2, 3)
list[0] = list[2]

// in, !in
println(4 !in list)
println("f" in "a"..'z')

// invoke
val f = { a: Int, b: Int => a + b }
println(f.invoke(2,3))
val f1 = f
println(f1(2,3))

// ==, !==
val account1 = Account("test", "foo")
val account2 = Account("test", "foo")
println(account1 == account2)
println(account1 !== account2)

// comparison
println("1" > "2")
```
Default and named parameters
```kotlin
fun smartFun(arg: Any) {
    when (arg) {
        is Int -> print(arg + 1)
        is String -> print(arg.length + 1)
        is IntArray -> print(arg.sum())
    }
}

fun smartCasts(arg: Any?) {
    if (arg is String) return
    println("$arg is String and its length is ${arg.length}"")
    if (arg is CharSequence && arg.length > 4) {
        println(arg.length)
    }
}

fun main(args: Array<String>) {
    smartFun(1)
    smartFun("JUG Zielona Góra")
    smartFun(intArrayOf(1, 2, 3, 4, 5))

    val x: Int? = null
    val y = if (x != null) {
        x + 4
    } else {
        0
    }
}
```
Destructuring objects

- underscore for unused variables (1.1)
- destructuring in lambdas (1.1)
Lambda and closures

fun up(arg: String): String = arg.toUpperCase()

fun main(args: Array<String>) {
    val countries = listOf("Poland", "Germany", "France", "Italy")

    // function getting String and returning String
    val toUpperCase: (String) -> String = ::up

    println(countries.map { element: String -> up(element) })
    println(countries.map { it -> up(it) })
    println(countries.map { toUpperCase.invoke(it) })
    println(countries.map(toUpperCase))
    println(countries.map { toUpperCase })

    var count = 0
    countries.forEach { count += it.length }
    println(count)

    countries.map { print("$it "); if (it == "Germany") return }  
    countries.map mapping@ { print(it); if (it == "Germany") return@mapping }  
    countries.map { print(it); if (it == "Germany") return@map }  
}
Expressions and statements

- if and when are expressions, not statements
  - val length = if (a is String) a.length else -1
  - val action = when (test) {
    in 0..5 -> OPEN
    else -> CLOSE
  }

- assignment is a statement, not an expression
  - if (a = b) does not compile
  - while ((line = bufferedReader.readLine()) != null) does not compile
Packages and source code structure

- packages
- import allows to import classes, functions, *
- type aliases
- multiple classes in one file
- arbitrary file names
- arbitrary directory structure
- visibility modifiers: private, protected, internal, public
Other languages have all these features

- Null safety
- No checked exceptions
- Extension functions
- Function types and lambdas
- Default and named parameters
- Properties
- Operator overloading
- Smart casts
- Data classes
- Immutable collections
- Enhanced switch-case
- String templates
- Ranges
- Infix notation

- Inline functions
- Coroutines (async/await)
- Great standard library
- Sealed classes
- Delegated and lazy properties
- Class delegation
- Singletons
- Nested functions
- Object decomposition
- Top-level functions
- Reified generics
- Raw strings
- 100% interoperable with Java 6
- And more...
Compile and run with Java code

- you can mix Java and Kotlin code in one project
- experiment with new language without breaking or rewriting the whole application
- small memory footprint of the Kotlin standard library
Understand decisions
What you need to know

- final by default
- platform types and nullability
- no primitives, no implicit widening conversions for numbers
- bytecode
- function names - conventions
- standard library
Final by default

- all classes, methods are final by default
  - tedious opening via ‘open’ keyword
  - interference with AOP (CGLIB), workarounds as compiler plugins ‘kotlin-spring’, ‘all-open’
- ‘override’ is a required keyword, not an annotation
- designing for inheritance
Platform types and nullability (!)

- any reference in Java may be null
- types of Java declarations are called platform types
- can be assigned to nullable or non-null type
- compiler, tools refers to them using as $T!$ which means $T$ or $T?$
- nullability annotations (JSR-305, Android, Lombok, JetBrains, Eclipse)
Bytecode generation

- Kotlin generates Java 6 or Java 8 bytecode
- on JVM lambdas does not use ‘invokedynamic’
Function naming conventions

- `a()` -> invoke
- `[]` -> `a.set`, `a.get`
- `==`, `!=` -> `equals`
- `for (element in container)` -> `uses iterator()`
- `in` -> `a.contains(b)`, `!in`
- infix notation `1.shl(2)` -> `1 shl 2`
Standard library

- **kotlin-runtime and kotlin-stdlib -> kotlin-stdlib**
  - < 1MB jar (JVM)
- **Kotlin classes and extension functions to Java classes**
  - kotlin
  - kotlin.collections
  - kotlin.comparisons
  - kotlin.concurrent
  - kotlin.io
  - kotlin.streams
  - kotlin.text
- kotlin.jvm
- kotlin.js
Platforms and tooling
Platforms and tooling

● JVM
  a. Java 6 and 8
● Android
● JavaScript (ES5.1)
  a. compatible with module systems like AMD, CommonJS
● native (LLVM)
  a. LLVM is used to compile Kotlin into native code
  b. technology preview for iOS, linux, MAC, (windows in the work)
● IntelliJ IDEA (Java to Kotlin converter), Eclipse
● Gradle, Maven, Ant
Where to start?

- Try online [https://try.kotlinlang.org/](https://try.kotlinlang.org/)
- Kotlin is Awesome! [https://kotlin.link/](https://kotlin.link/)
- This presentation
- And code examples
What will be your next JVM language?

https://en.wikipedia.org/wiki/List_of_JVM_languages