Kotlin для JVM: просто, ясно, безопасно

Mikhail Glukhikh
mailto: Mikhail.Glukhikh@jetbrains.com
JetBrains, Senior Software Developer

JVM языки (год выпуска)

- Java (1995)
- Groovy (2003)
- Scala (2004)
- Clojure (2007)
- **Kotlin (2016)**
- Fantom (2005), Ceylon (2013), Gosu (2014)

Kotlin: характеристики + цели

- Более **ясный и краткий** язык, чем Java
  - Свойства
    - `class Point(val x: Double, val y: Double)`

Класс Point: аналог на Java

```
public class Point {
    private final double x;
    private final double y;

    public Point(double x, double y) {
        this.x = x;
        this.y = y;
    }
    public double getX() {
        return x;
    }
    public double getY() {
        return y;
    }
}
```

Kotlin: характеристики + цели

- Более **ясный и краткий** язык, чем Java
  - Функции-расширения
    - ```
      fun Array<*>.firstOrNull() =
              if (size >= 1) this[0] else null
      ```
      ```

Kotlin: характеристики + цели

- Более **ясный и краткий** язык, чем Java
  - Функции-расширения
    - ```
      fun Array<*>.firstOrNull() =
              if (size >= 1) this[0] else null
      ```
  - Вывод типов
    - ```
      val name /* : String */ = "Mikhail"
      ```

Kotlin: характеристики + цели

- Более **ясный и краткий** язык, чем Java
  - Функции-расширения
    - `fun Array<*>.firstOrNull() = if (size >= 1) this[0] else null`
  - Вывод типов
    - `val name /* : String */ = "Mikhail"`
  - Строковые шаблоны
    - `val greeting = "Hello, $name, I am glad to see you"`

Hello, Kotlin!

```
fun Array<*>.firstOrNull() =
        if (size >= 1) this[0] else null

fun main(args: Array<String>) {
    //                          Elvis vv
    val name = args.firstOrNull() ?: "Kotlin"
    println("Hello, $name, I am glad to see you")
}
```

Elvis operator

- Elvis-оператор

```
first ?: second // is equivalent to

if (first != null) first else second
```

Hello, Java!

```
public class Main {
    public static void main(String[] args) {
        String name = args.length > 0 ?
                      args[0] : "Java";
        System.out.println("Hello, " + name +
                ", I am glad to see you");
    }
}
```

Kotlin: характеристики + цели

- Более **безопасный** язык, чем Java
  - Защита от `NullPointerException`

Защита от NullPointerException

```
fun caller(x: Type? /* ? means "can be null" */) {
    x.foo()   // Error: x can be null
}
```

Защита от NullPointerException

```
fun caller(x: Type? /* ? means "can be null" */) {
    x.foo()   // Error: x can be null
    x?.foo()  // OK, safe call
}
```

Защита от NullPointerException

```
fun caller(x: Type? /* ? means "can be null" */) {
    x.foo()   // Error: x can be null
    x?.foo()  // OK, safe call

    x?.foo() ?: doIfNull() // OK: safe call + Elvis
}
```

Защита от NullPointerException

```
fun caller(x: Type? /* ? means "can be null" */) {
    x.foo()   // Error: x can be null
    x?.foo()  // OK, safe call

    x?.foo() ?: doIfNull() // OK: safe call + Elvis

    if (x != null) {
        x.foo() // OK, smart cast
    }
}
```

Защита от NullPointerException

```
fun caller(x: Type? /* ? means "can be null" */) {
    x.foo()   // Error: x can be null
    x?.foo()  // OK, safe call

    x?.foo() ?: doIfNull() // OK: safe call + Elvis

    if (x != null) {
        x.foo() // OK, smart cast
    }
    x!!.foo() // OK, but KNPE is possible
}
```

Kotlin: характеристики + цели

- Более **безопасный** язык, чем Java
  - Защита от `NullPointerException`
  - Защита от `ClassCastException`

Защита от ClassCastException

```
interface Base
class Derived : Base {
    fun fromDerived() {}
}
fun caller1(x: Base) {
    (x as  Derived).fromDerived()  // OK, CCE possible
}
```

Защита от ClassCastException

```
interface Base
class Derived : Base {
    fun fromDerived() {}
}
fun caller2(x: Base) {
    (x as? Derived).fromDerived()  // Error
}
```

Защита от ClassCastException

```
interface Base
class Derived : Base {
    fun fromDerived() {}
}
fun caller2(x: Base) {
    (x as? Derived).fromDerived()  // Error
    (x as? Derived)?.fromDerived() // OK, safe call
}
```

Защита от ClassCastException

```
interface Base
class Derived : Base {
    fun fromDerived() {}
}
fun caller2(x: Base) {
    (x as? Derived).fromDerived()  // Error
    (x as? Derived)?.fromDerived() // OK, safe call
    if (x is Derived) {
        x.fromDerived()            // OK, smart cast
    }
}
```

Kotlin: характеристики + цели

- Более **безопасный** язык, чем Java
  - Защита от `NullPointerException`
  - Защита от `ClassCastException`
  - Защита от `ArrayStoreException`

Защита от ArrayStoreException

```
fun foo(arr: Array<String>) {
    val anyArr: Array<Any> = arr // ???
}
```

Защита от ArrayStoreException

```
fun foo(arr: Array<String>) {
    // Error! Arrays are invariant in Kotlin
    val anyArr: Array<Any> = arr
}
```

Защита от ArrayStoreException

```
fun foo(arr: Array<String>) {
    // Error! Arrays are invariant in Kotlin
    val anyArr: Array<Any> = arr
    anyArr[0] = 42 // ASE in Java
}
```

Array -> List

```
fun foo(list: List<String>) {
    // OK! Lists are covariant in Kotlin
    val anyList: List<Any> = list
    anyList[0] = 42 // Error! List is read-only
}
```

Variance

```
fun foo(list: List<String>) {
    // OK! Lists are covariant in Kotlin
    val anyList: List<Any> = list
    anyList[0] = 42 // Error! List is read-only
}

interface List<out T> : ... { ... }
```

Variance

```
fun foo(list: List<String>) {
    // OK! Lists are covariant in Kotlin
    val anyList: List<Any> = list
    anyList[0] = 42 // Error! List is read-only
}

interface List<out T> : ... { ... }

// Contravariant
interface Comparable<in T> : ... { ... }
```

Интероперабельность: класс Line

- Класс Line, использующий класс Point
  - `class Point(val x: Double, val y: Double)`
- Builder для создания линии по двум точкам

Класс Line на Java

```
class Point(val x: Double, val y: Double)
// Java: ax + by = 1
public class Line {
    private final double a;
    private final double b;
    public Line(double a, double b) {
        this.a = a; this.b = b;
    }
    static public Line create(Point p1, Point p2) {
        double d  = p1.getX() * p2.getY() -
                    p1.getY() * p2.getX();
        double da = p2.getY() - p1.getY();
        double db = p1.getX() - p2.getX();
        return new Line(da / d, db / d);
    }
}
```

Класс Line на Kotlin

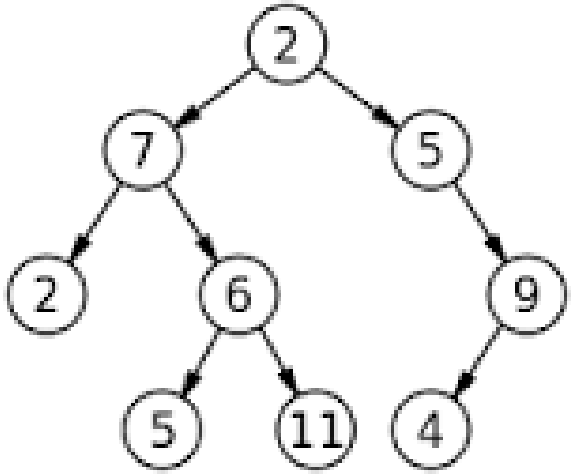
```
class Point(val x: Double, val y: Double)
// Kotlin: ax + by = 1
class Line (val a: Double, val b: Double)

fun createLine(p1: Point, p2: Point): Line
{
    val d  = p1.x * p2.y - p1.y * p2.x
    val da = p2.y - p1.y
    val db = p1.x - p2.x
    return Line(da / d, db / d)
}
```

Алгебраические классы

- Пример: бинарное дерево

- Узел дерева хранит ключ / значение +
  - или лист
  - или поддерево



Бинарное дерево

```
sealed class Element<Key, out Value>(
    val key: Key, val value: Value) {
}
```

Бинарное дерево

```
sealed class Element<Key, out Value>(
    val key: Key, val value: Value) {

    class Leaf<Key, out Value>(
        key: Key, value: Value) : Element<Key, Value>(key, value)

}
```

Бинарное дерево

```
sealed class Element<Key, out Value>(
    val key: Key, val value: Value) {

    class Leaf<Key, out Value>(
        key: Key, value: Value) : Element<Key, Value>(key, value)

    class Tree<Key, out Value>(
            key: Key, value: Value,
            val left: Element<Key, Value>,
            val right: Element<Key, Value>? = null
    ) : Element<Key, Value>(key, value)
}
```

Поиск в дереве

```
fun <K, V>                          Element<K, V>.search(
    key: K): V? =
        if (this.key == key) value
        // Exhaustive when
        else when (this) {
            is Element.Leaf -> null
            is Element.Tree -> left.search(key) ?:
                               right?.search(key)
        }
```

Перегрузка операций

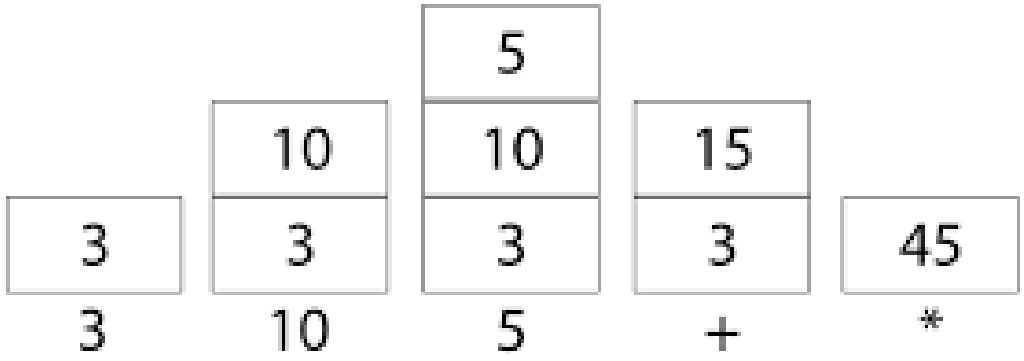
```
fun <K : Comparable<K>, V> Element<K, V>.search(
    key: K): V? =
        if (this.key == key) value
        // Exhaustive when
        else when (this) {
            is Element.Leaf -> null
            is Element.Tree -> if (key < this.key)
                                   left.search(key)
                               else right?.search(key)
        }
```

Функции высшего порядка

Вычислить выражение
в обратной польской записи

```
    "3 10 5 + *"                          3 * (10 + 5)
// обратная польская запись          инфиксная запись
```

Equation: 3 10 5 + *



Калькулятор

```
private val operationMap =
        mapOf<String, (Double, Double) -> Double>(
            "+" to { x, y -> x + y },  "-" to { x, y -> y - x },
            "*" to { x, y -> x * y },  "/" to { x, y -> y / x })
```

Калькулятор

```
private val operationMap =
        mapOf<String, (Double, Double) -> Double>(
            "+" to { x, y -> x + y },  "-" to { x, y -> y - x },
            "*" to { x, y -> x * y },  "/" to { x, y -> y / x })


fun polishCalculate(expr: String): Double {
    val stack = Stack<Double>()
    for (arg in expr.split(" ")) {
        val op = operationMap[arg]
        if (op != null) stack.execute(op)
        else stack.push(arg.toDouble())
    }
    return stack.pop()
}
```

Калькулятор

```
private val operationMap =
        mapOf<String, (Double, Double) -> Double>(
            "+" to { x, y -> x + y },  "-" to { x, y -> y - x },
            "*" to { x, y -> x * y },  "/" to { x, y -> y / x })
fun Stack<Double>.execute(op: (Double, Double) -> Double) =
        push( op(pop(), pop()) )
fun polishCalculate(expr: String): Double {
    val stack = Stack<Double>()
    for (arg in expr.split(" ")) {
        val op = operationMap[arg]
        if (op != null) stack.execute(op)
        else stack.push(arg.toDouble())
    }
    return stack.pop()
}
```

Демонстрационный проект

- https://github.com/mglukhikh/JavaToKotlin
- Содержит рассмотренные выше примеры:
  - _1_hello: Hello, $Language!
  - _2_exceptions: Исключения
  - _3_geometry: Точка + Линия
  - _4_tree: Бинарное дерево
  - _5_polish: Калькулятор обратной польской записи
  - ...

Kotlin сейчас

- 1.0: 15 февраля 2016
  - Обратная совместимость
- Open-Source: http://github.com/JetBrains/Kotlin
- Компиляция в JVM / Android / JS
- Плагины для Intellij IDEA (встроенный) и для Eclipse

Kotlin сейчас

- Около 25 разработчиков в проекте Kotlin
- Около 50 компаний, более десяти тысяч активных программистов на Kotlin
- Около 3 миллионов строк кода в открытых проектах на GitHub
  - Из них >80% в сторонних проектах
- Около 1500 участников в Kotlin Slack
- 2 книги
  - Antonio Leiva. **Kotlin for Android Developers**
  - Dmitry Jemerov, Svetlana Isakova. **Kotlin in Action**

Полезные ссылки

- http://kotlinlang.org – сайт языка
- https://kotlinlang.org/docs/reference/ – документация
- http://github.com/JetBrains/Kotlin – репозиторий проекта на GitHub
- http://blog.jetbrains.com/kotlin/ – блог языка
- http://kotlin.link/ – Kotlin-ресурсы
- https://kotlinlang.slack.com/ – Kotlin Slack

- Вопросы?