توسعه اندروید
زبان کاتلین

ابوالفضل عبدلی

کاتلین چیست؟

- مختصر، مولتی‌پلتفرم، فان!
- Functional و Object-oriented
- سازنده: JetBrains
- اولین نسخه منتشر شده در July 22, 2011
- آخرین نسخه: 1.8.20 (در زمان نوشتن این جزوه)

- قابلیت سازگاری با جاوا و دیگر زبان ها
- در این جزوه: استفاده از Kotlin for JVM و اجرا روی JVM

Java?

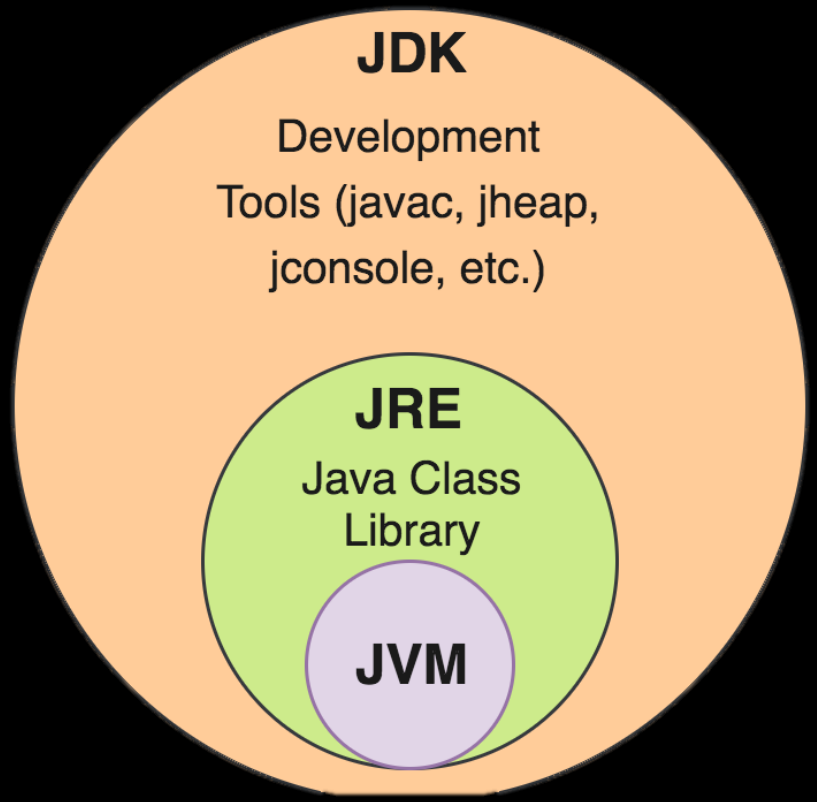
- ایجاد توسط Sun Microsystems در سال 1995
- در حال حاضر تحت مالکیت Oracle
- High-level, Object-oriented

شیوه اجرای کدهای جاوا

Java → Bytecode → JVM

JVM vs JRE vs JDK

JVM: Java Virtual Machine
JRE: Java Runtime Environment
JDK: Java Development Kit



نصب JDK

- دانلود و نصب
  - link1, link2

- افزودن مسیر JDK به PATH سیستم
  - مثال: C:\Program Files\Java\jdk-11.0.1\bin

- تست اجرای موفق در CMD
  - اجرای دستور ‘java –version’

نصب کاتلین (روش 1 – عدم پیشنهاد)

- دانلود کامپایلر از لینک: link
- اضافه کردن kotlinc به PATH سیستم
- ساخت یک فایل با پسوند kt. و جاگذاری کد زیر در آن
  - fun main() { println("Hello, World!") }
- کامپایل کردن فایل با دستور زیر
  - kotlinc {file}.kt -include-runtime -d {file}.jar
- اجرا با استفاده از Java
  - java -jar {file}.jar

نصب کاتلین (روش 2 – پیشنهادی)

- نصب IntelliJ IDEA (که شامل Kotlin نیز میشود)
  - link1, link2

تمام :)

اجرای اولین پروژه کاتلین در IntelliJ

1. ساخت پروژه جدید در IntelliJ
2. جاگذاری کد زیر

```
fun main() {
    println("Hello, World!")
}
```

3. اجرای کد با دکمه play

آشنایی با محیط IntelliJ

- Panels
  - Project
  - Structure
  - Run
  - Build
  - Terminal
  - Problems
  - Version Control (تحقیق)

Program entry point

```
fun main() {
    println("Hello world!")
}
```

VS

```
fun main(args: Array<String>) {
    println(args.contentToString())
}
```

چاپ در خروجی استاندارد

- print()
  - چاپ در خط جاری
- println()
  - معنی ln: چاپ در خط جاری، و برو به خط بعد

کامنت گذاری

- // This is an single-line (end-of-line) comment

- /*
  This is a block comment
  on multiple lines.
  */

متغیرها

- val a: Int = 1 // تعریف کامل با نوع صریح و مقدار اولیه
- val b = 2 // تشخیص داده میشودIntنوع
- val c: Int // نوع موردنیاز به دلیل نبود مقدار اولیه
  - در این صورت باید پیش از استفاده از متغیر، آن را مقداردهی کنیم:
  - c = 3 // مقداردهی با تاخیر

- نکته: متغیرهای بالا به صورت غیرقابل تغییر (immutable) تعریف میشوند

متغیرِ قابل تغییر! (Mutable)

- تعریف متغیر غیرقابل تغییر با کلمه کلیدی val

- val x = 10
  x = 11 // will cause an error

- تعریف متغیر قابل تغییر با کلمه کلیدی var

- var x = 10
  x = 11 // it's ok

انواع داده ای

- Primitive data type? Non-Primitive data types? (تحقیق)

- در کاتلین، نوع داده ی primitive و non-primitive نداریم!
- همه چیز در کاتلین یک شئ(Object) خواهد بود
  - یعنی هر متغیر از هر نوع داده ای میتواند توابع و اعضا نیز داشته باشد (بر خلاف قانون نوع داده ای primitive)
  - (تحقیق) در نهایت انواع داده ای کاتلین باید روی JVM اجرا شوند و قابل دسترسی باشند، جاوا دارای primitive و non-primitive هستند

نوع داده ای عدد صحیح (Integers)

Type	Size (bits)	Min value	Max value
`Byte`	8	-128	127
`Short`	16	-32768	32767
`Int`	32	-2,147,483,648 (-2^{31})	2,147,483,647 ($2^{31} - 1$)
`Long`	64	-9,223,372,036,854,775,808 (-2^{63})	9,223,372,036,854,775,807 ($2^{63} - 1$)

- val one = 1 // Int
- val threeBillion = 3000000000 // Long
- val oneLong = 1L // Long (with ‘L’ suffix)
- val oneByte: Byte = 1 // Byte

- کامپایلر میتواند نوع داده ای را بر اساس رنج عدد وارد شده تشخیص دهد (در صورتی که نوع به صورت صریح تعریف نشده باشد)
  - نوع پیشفرض عبارات عدد صحیح Int خواهد بود (در صورتی که از محدوده ی Int خارج نباشد)

نوع داده ای عدد اعشاری (Floating-point)

Type	Size (bits)	Significant bits	Exponent bits	Decimal digits
`Float`	32	24	8	6-7
`Double`	64	53	11	15-16

تحقیق : IEEE 754 standard

- عبارات عدد اعشاری به صورت پیشفرض از نوع Double درنظرگرفته میشود

```
val pi = 3.14 // Double
// val one: Double = 1 // Error: type mismatch
val oneDouble = 1.0 // Double
```

- نوع Float به صورت صریح با پسوند F یا f
- در صورتی که مقدار بیش از 6-7 اعشار وجود داشته باشد، گرد خواهد شد

```
val e = 2.7182818284 // Double
val eFloat = 2.7182818284f // Float, actual value is 2.7182817
```

عبارات عددی (Number literals)

- Normal number: 123, 111, 1000
  - Inferred as Int by default
  - Long literal with ‘L’ suffix: 123L, 111L, 1000L
- Hexadecimals: 0x7B, 0x6F, 0x3E8
  - means 123, 111, 1000
- Binaries: 0b01111011, 0b01101111, 0b0000001111101000
  - means 123, 111, 1000
- Floating-point: 123.0, 10.5, 16.75
  - Infered as Double by default
  - Float with ‘f’ or ‘F’ suffix: 123.0f, 10.5F, 16.75f

- استفاده از زیرخط (_) برای خوانایی بیشتر عبارات عددی

```
val oneMillion = 1_000_000
val creditCardNumber = 1234_5678_9012_3456L
val socialSecurityNumber = 999_99_9999L
val hexBytes = 0xFF_EC_DE_5E
val bytes = 0b11010010_01101001_10010100_10010010
```

تبدیل اعداد

- برخلاف دیگر زبان ها (مانند جاوا) در کاتلین تبدیل عدد به صورت ضمنی وجود ندارد
  - (Research: link)

- تبدیل اعداد به صورت صریح:
  - استفاده از متدهای ‘.toX()’
  - .toDouble(), .toInt(), .toFloat(), etc.

```
val b: Byte = 1 // OK, literals are checked statically
// val i: Int = b // ERROR
val i1: Int = b.toInt()
```

WTF?

- کد زیر با موفقیت اجرا خواهد شد. چرا؟؟ (Research: link)

```
val l = 1L + 3 // Long + Int => Long
```

WTF is stand for “why the face”, (meaning.. why do you look like that)...

تبدیل رشته متنی به عدد

- استفاده از متد های ()toX. روی شئ رشته متنی

```
val intString = "1234"
val doubleString = "1234.74"
val int = intString.toInt()
val double = doubleString.toDouble()
```

تبدیل هر چیز به رشته متنی

- استفاده از متد toString(). روی هر شئ از هر نوع داده ای

```
val int = 1234
val double = 1234.75

val str1 = int.toString()
val str2 = double.toString()
```

عملیات های عددی

- عملیات های اصلی:

```
println(1 + 2)
println(2_500_000_000L - 1L)
println(3.14 * 2.71)
println(10.0 / 3)
println(10 % 3)
```

- تقسیم دو عدد صحیح
  - بخش اعشار حذف میشود

```
println(5 / 2) // prints 2
```

- برای نمایش اعشار، یکی از طرفین باید به نوع اعشاری تبدیل شوند

```
println(5 / 2.0) // prints 2.5
println(5 / 2.0f) // prints 2.5
println(5 / 2.toDouble()) // prints 2.5
```

- عملگر یکی (Unary)

- x++
- x--

  vs
- ++x
- --x

اعداد صحیح بدون علامت

Type	Size (bits)	Min value	Max value
UByte	8	0	255
UShort	16	0	65535
UInt	32	0	2^32 - 1
ULong	64	0	2^64 - 1

```
val b: UByte = 1u  // UByte, expected type provided
val s: UShort = 1u // UShort, expected type provided
val l: ULong = 1u  // ULong, expected type provided

val a1 = 42u // UInt: no type provided, constant fits in UInt
val a2 = 0xFFFF_FFFF_FFFFu // ULong: no type provided, constant doesn't fit in UInt

val b1 = 1uL // ULong, with 'uL' suffix & no type provided
val b2 = 1UL // ULong, with 'UL' suffix & no type provided
```

کاربرد عدد صحیح بدون علامت

- کاربرد اصلی اعداد بدون علامت، استفاده از محدوده ی کامل اشغال شده در حافظه برای ذخیره اعداد مثبت با بازه ی وسیع تر است
  - مثال: نمایش اعداد مربوط به رنگ ها (32-bit AARRGGBB format)

```
data class Color(val representation: UInt)

val yellow = Color( representation: 0xFFCC00CCu)
```

- چرا نباید همه جا از اعداد بدون علامت استفاده کرد؟
  - (Research: link, link2)

Booleans

- مقادیر منطقی: true, false

- عملگر های منطقی

- || disjunction (OR)
- && conjunction (AND)
- ! negation (NOT)

Characters

- نوع ‘Char’
- داخل single quotes مثل ‘A’, ‘b’, ‘5’

- دریافت مقدار عددی کاراکترِ عدد با متد digitToInt()

Escape characters

- \t tab
- \b backspace
- \n new line (LF)
- \r carriage return (CR)
- \’ single quotation mark
- \" double quotation mark
- \\ backslash
- \$ dollar sign
- \u prefix for Unicode character codes

Characters Example

```
val aChar: Char = 'a'
println(aChar) // prints a

print('\n') // prints newline

println('\u0041') // prints A

val digitChar = '5'
val integerValueOfChar: Int = digitChar.digitToInt()
println(integerValueOfChar) // prints 5
```

رشته های متنی String

- نوع داده ای String
- یک مقدار رشته متنی، دنباله ای از کاراکترهاست
- داخل double quotes مثل “123”, “hello”, “abcd”
- دسترسی به کاراکترهای رشته با اندیس
  - str[0], str[10]

- رشته های متنی غیرقابل تغییر هستند، هر تغییر روی رشته های متنی، یک رشته متنی جدید تولید میکند

```
val str = "abcd"
println(str.uppercase()) // Create and print a new String object, prints "ABCD"
println(str) // The original string remains the same, prints "abcd"
```

الحاق رشته های متنی (Concatenate)

- عملگر +
  - در صورتی که اولین عملوند String باشد، بقیه عملوند ها به String تبدیل میشوند

```
val s1 = "Hello" + " " + "World"
println(s1) // prints "Hello World"

val s2 = "abc" + 1
println(s2 + "def") // prints "abc1def"
```

- در اکثر موارد توصیه میشود به جای string concat از raw string ها و string template ها استفاده شود (در ادامه مورد بررسی قرار میگیرد)

عبارات رشته متنی (String literals)

- رشته های متنی escape شده، شامل کاراکترهای escape

```
val s = "Hello\nworld!"
```

- Raw strings
  - با استفاده ” ” ” triple quote “
  - میتواند شامل کاراکتر خط جدید، و تمامی کاراکترهای دیگر باشد
  - نمیتواند شامل escaped characters باشد

Raw strings

```
val text = """
    Tell me and I forget.
    Teach me and I remember.
    Involve me and I learn.
    (Benjamin Franklin)
"""
println(text) // prints the text with all leading whitespaces
```

Raw strings

- حذف whitespaces های اضافه
  - استفاده از متد trimIndent(). تشخیص خودکار indent
  - استفاده از یک کاراکتر برای مارجین مشخص و متد trimMargin().
    - کاراکتر مارجین پیشفرض ' | ' pipe است
    - trimMargin(">")

```
val text = """
    |Tell me and I forget.
    |Teach me and I remember.
    |Involve me and I learn.
    |(Benjamin Franklin)
""".trimMargin()
println(text) // prints the text without leading
              // whitespaces and margin (the pipe '|' char)
```

String templates

- Template expression هر درون یک رشته متنی با dollar sign ‘$’ شروع میشود
- در ادامه یک نام متغیر یا یک عبارت (درون { }) قرار میگیرد

```
val i = 10
println("i = $i") // Prints "i = 10"


val s = "abc"
println("$s.length is ${s.length}") // Prints "abc.length is 3"


println("${1 + 2 * 3}") // Prints "7"
```

Escape characters در Raw String

- استفاده از string template ها در raw string ها:
  - جهت درج escape character ها (که به صورت عادی قابل درج نیستند)
  - جهت درج علامت $ (escape کردن علامت $)

```
val price = """
    price is:${"\t\t"}${'$'}_10
""".trimIndent()
print(price) // prints "price is:		$_10"
```