برنامه نویسی دستگاه های سیار (CE364)

جلسه سوم:
آشنایی با زبان کاتلین

سجاد شیرعلی شهرضا

پاییز 1401

دوشنبه، 11 مهر 1401

زبان کاتلین

- در سال 2011 برای اولین بار ارائه شده است
- زبان برنامه نویسی عمومی
- تایپ دهی ایستا (Static-typing)
- طراحی شده جهت تعامل با جاوا
- معمولا بر روی JVM اجرا میشود
- گوگل در سال 2019 اعلام کرد که انتخاب برتر برای برنامه نویسی اندروید است
- قابل تبدیل به جاوا اسکریپت
- کاتلین نام جزیره ای بین روسیه و فنلاند

تاریخچه

- معرفی توسط JetBrains در سال 2011
- عرضه نسخه 1.0 در سال 2016
- پشتیبانی کلاس-اول برای اندروید توسط گوگل در سال 2017
- اعلام به عنوان زبان برگزیده برای اندروید توسط گوگل در سال 2019

گسترش

- چهارمین زبان مورد علاقه در نظرسنجی سال StackOverFlow 2020
  - یازدهمین زبان مورد علاقه و ششمین زمان دارای خواستار در سال 2022
- استفاده توسط بسیاری از شرکت ها
- استفاده برای سرور (backend)
- استفاده برای نرم افزارهای تحت وب (web development)
- استفاده برای برنامه نویسی موبایل (اندروید)

یک برنامه ساده!

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

- خروجی:

Hello, world!

برنامه کمی پیچیده تر!

```
fun main() {
    println("Happy Birthday, Rover!")

    // Let's print a cake!
    println("    ,,,,,    ")
    println("    |||||    ")
    println(" =========")
    println("@@@@@@@@@@")
    println("{~@~@~@~@~}")
    println("@@@@@@@@@@")

    // This prints an empty line.
    println("")

    println("You are already 5!")
    println("5 is the very best age to celebrate!")
}
```

متغیر

- تعریف:

```
val age = 5
```

- استفاده:

```
${variable}
```

- مثال:

```
println("You are already ${age}!")
println("${age} is the very best age to celebrate!")
```

برنامه کمی کاملتر

```
fun main() {

    val age = 5 * 365
    val name = "Rover"

    println("Happy Birthday, ${name}!")

    // Let's print a cake!
    println("   ,,,,,   ")
    println("   |||||   ")
    println(" =========")
    println("@@@@@@@@@@")
    println("{~@~@~@~@~}")
    println("@@@@@@@@@@")

    // This prints an empty line.
    println("")

    println("You are already ${age} days old, ${name}!")
    println("${age} days old is the very best age to celebrate!")
}
```

تعریف و استفاده از تابع

```
fun main() {
    printBorder()
    println("Happy Birthday, Jhansi!")
    printBorder()
}

fun printBorder() {
    repeat(23) {
        print("=")
    }
    println()
}
```

تکرار

```
fun printBorder(border: String) {
    repeat(23) {
        print(border)
    }
    println()
}
```

ارسال متغیر به تابع

```
fun main() {
    val border = "%"
    printBorder(border)
    println("Happy Birthday, Jhansi!")
    printBorder(border)
}

fun printBorder(border: String) {
    repeat(23) {
        print(border)
    }
    println()
}
```

چند متغیر برای تابع

```
fun main() {
    val border = "`-._,-'"
    val timesToRepeat = 4
    printBorder(border, timesToRepeat)
    println("  Happy Birthday, Jhansi!")
    printBorder(border, timesToRepeat)
}

fun printBorder(border: String, timesToRepeat: Int) {
    repeat(timesToRepeat) {
        print(border)
    }
    println()
}
```

مثال بیشتر از تابع

```
fun main() {
    val age = 24
    val layers = 5
    printCakeCandles(age)
    printCakeTop(age)
    printCakeBottom(age, layers)
}

fun printCakeCandles(age: Int) {
    print (" ")
    repeat(age) {
          print(",")
    }
    println() // Print an empty line

    print(" ") // Print the inset of the candles on the cake
    repeat(age) {
        print("|")
    }
    println()
}

fun printCakeTop(age: Int) {
    repeat(age + 2) {
        print("=")
    }
    println()
}
fun printCakeBottom(age: Int, layers: Int) {
    repeat(layers) {
        repeat(age + 2) {
            print("@")
        }
        println()
    }
}
```

سوال؟

ویژگی های زبان کاتلین

تایپ دهی ایستا

- همانند جاوا، فقط مقدار سازگار با نوع متغیر را میتوان به آن نسبت داد

```
var m : Int = 12

m = 10          // ok

m = "twelve" // error!

m = 10.0        // error!
```

خروج از تابع در هرجا

```
fun main() {

    intArrayOf(4, 5, 6).forEach lambda@ {

        if (it == 5) return@lambda

        println(it)

    }

    println()

    loop@ for (i in 0 .. 3) {

        for (j in 0 .. 3) {

            if (i + j == 4) continue@loop

            if (i + j == 5) break@loop

            println(i + j)

        }

    }

}
```

جستجو در مجموعه

```
val languageArray = arrayOf("Serbian", "Swahili", "Japanese", "German", "Spanish")

val selectedLang = languageArray

.filter { name -> name.startsWith("s", ignoreCase = true) }

//or: .filter { it.startsWith("s", ignoreCase = true) }

.sortedBy { name -> name.length }

.first()
```

توسعه اشیاء

```
class Person {

  var firstName = ...

  var lastName = ...

}

// Maybe the author of the Pers
on class forgot to include this

fun Person.setName(firstName: S
tring, lastName: String) {

  this.firstName = firstName

  this.lastName = lastName

}
```

- امکان اضافه کردن تابع به اشیاء پس از تعریف آنها
- امکان توسعه اشیاء به سادگی
- خواناتر شدن برنامه

برخی تفاوت های کاتلین و جاوا

کاتلین	جاوا
• امکان تغییر و ادغام کد در اندروید، JVM، ...	• عدم امکان تغییر و ادغام برای محیط های دیگر
• طراحی شده برای برنامه نویسی شی گرا و تابعی	• زبان شی گرا
• طراحی شده برای مختصر بودن	• طراحی شده برای دقیق بودن
• تبدیل نوع هوشمند و ضمنی	• قوانین نوع سختگیرانه
• عدم امکان استفاده از null	• امکان استفاده از null
• پشتیبانی از ویژگی های مدرن همانند کوروتین	• ساختارهای پیچیده جهت فراهم کردن ویژگی های مدرن

تفاوت متغیر متغیر و ثابت!

```
var <variableName> = <value> //mutable

val <variableName> = <value> //read-only

val <variableName>: <dataType> = <value> // explicit type casting
```

مقداردهی اولیه متغییر

```
val number = 17

println("number = $number")

number = 42  // Not allowed, th
rows an exception
```

```
var number = 17

println("number = $number")

number = 42  // var can be reas
signed

println("number = $number")
```

داده عدد صحیح

```
val byte: Byte = 127

val short: Short = 32767

val int: Int = 2147483647

val long: Long = 9223372036854775807
```

داده عدد اعشاری

```
val float: Float = 3.4028235e38f

val double: Double = 1.79769313486231570e308
```

- اعداد اعشاری به طور پیش فرض، Double هستند

داده متنی

```
val character: Char = '#'

val text: String = "Learning about Kotlin's data types"
```

- امکان تعریف رشته چند خطی با """text"""

داده بولین

```
val yes: Boolean = true

val no: Boolean = false
```

استنتاج نوع متغير

```
val string = "Educative"

val int = 27

val long = 42L

val double = 2.71828

val float = 1.23f

val bool = true
```

ساختار کلی شرطی

```
<conditional> (<desiredCondition>) <code>

<conditional> (<desiredCondition>) { <codeBlock>

}
```

ساختار if

```
var max = a

if (a < b) max = b

// With else

var max: Int

if (a > b) {

    max = a

} else {

    max = b

}

// As expression

val max = if (a > b) a else b
```

ساختار when

```
when (x) {
    1 -> print("x == 1") //branch 1
    2 -> print("x == 2") //branch 2
    else -> { // Note the block
        print("x is neither 1 nor 2")
    }
}
```

- همانند ساختار switch

حلقه for

```
for (item in collection) print(item)
```

```
for (i in 1..3) {
    println(i)
}
```

حلقه while و do-while

```
while (x > 0) {

    x--

}

do {

    val y = retrieveData()

} while (y != null) // y is visible here!
```

لیست

- شروع از صفر
- پویا و قابل تغییر

```
val numbers = listOf("one", "two", "three", "four")

println("Number of elements: ${numbers.size}")

println("Third element: ${numbers.get(2)}")

println("Fourth element: ${numbers[3]}")

println("Index of element \"two\" ${numbers.indexOf("two")}")
```

مجموعه

```
val numbers = setOf(1, 2, 3, 4)

println("Number of elements: ${numbers.size}")

if (numbers.contains(1)) println("1 is in the set")

val numbersBackwards = setOf(4, 3, 2, 1)

println("The sets are equal: ${numbers == numbersBackwards}")
```

آرایه

- مشابه آرایه جاوا
  - طول ثابت
  - مقادیر قابل تغییر

```
val priorities = arrayOf("HIGH", "MEDIUM", "LOW")

println(priorities[1])

priorities[1] = "NORMAL"

println(priorities[1])

println(priorities.size)
```

تعریف تابع

- امکان تعریف خارج از کلاس
- امکان ارسال و دریافت تابع به عنوان پارامتر تابع

```
fun <variableName>(<inputName>: <inputType>): <returnType>

fun fibonacci(index: Int): Long
```

توابع توسعه یافته (Extended Function)

```
fun MutableList < Int > .swap(index1: Int, index2: Int) {

val tmp = this[index1]

this[index1] = this[index2]

this[index2] = tmp

}
```

سوال؟