Kotlin

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Razvoj mobilnih aplikacija i servisa
Računarstvo i informatika

2024.

Kotlin

- Osnovni jezik za Android programiranje
- Razvijen od strane JetBrains-a
- Uključen u Android Studio
- Za razliku od Jave, predstavlja mešavinu objektne i funkcionalne paradigme
- Lambda izrazi

Deklaracija promenljivih

- Promenljive se u Kotlin-u definišu na dva načina:
  - **var** – promenljiva koja može menjati vrednost
  - **val** – promenljiva koja ostaje konstantna nakon prve dodele vrednosti

```
val a: Int = 1  // immediate assignment
val b = 2   // `Int` type is inferred
val c: Int  // Type required when no initializer is provided
c = 3       // deferred assignment
```

Operatori

- Matematički (+, -, *, /, %)
- Inkrementiranje (++), dekrementiranje (--)
- Komparatori (<, <=, >, >=)
- Operator dodele (=)
- Poređenja (==, !=, ===)

Tipovi podataka

- Numerički tipovi
- Logički
- Karakteri
- Stringovi
- Nizovi

Integer tipovi

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 ($2^{63} - 1$)

FP tipovi

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

Konverzija među numeričkim tipovima

- Manji tipovi nisu podvrste većih!
- Potrebna je eksplicitna konverzija, osim u slučaju primene aritmetičkih operatora

```
val b: Byte = 1 // OK, literals are checked statically
val i: Int = b // ERROR
val i1: Int = b.toInt()
val l = 1L + 3 // Long + Int => Long
```

Boolean

- Boolean tip – *objekti* koji mogu biti ***true*** ili ***false***
- Operatori: &&, || i !

Character

- Tip Char
- Pišu se u jednostrukim navodima (’a’)
- Escaped karakteri:
  - \t – tab
  - \b – backspace
  - \n – novi red (LF)
  - \r – carriage return (CR)
  - \’ – jednostruki navod
  - \“ – navodnici
  - \\ – backslash
  - \$ – dolar znak

Stringovi

- Tip String, navodi se kao „neki string“
- Elementima stringa se može pristupiti indeksiranjem: s[i]
- Stringovi su *immutable* – nemoguće je promeniti im vrednosti, samo dodeliti novu
  - Drugim rečima, ne može s[i]=’a’
  - Može s=„Sava“

Stringovi

- Za konkatenaciju se koristi operator +
  - `val s = "abc" + 1 + "def" // abc1def`
  - Nije preporučljivo koristiti konkatenaciju, bolji način je sa string templates
- Multiline string – string koji ne može da sadrži escaped karaktere, ali može da ima novi red u sebi:

```
val text = """
    for (c in "foo")
        print(c)
"""
```

String Templates

- Koristi se za umetanje vrednosti u string

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

- Ukoliko se umeće neki izraz, moraju se iskoristiti *{* i *}*

```
val s = "abc"
println("$s.length is ${s.length}")
```

Nizovi

- Array – niz elemenata istog tipa
- Broj elemenata je fiksni, jedini način da se niz proširi jeste kreiranje novog niza

```
var riversArray = arrayOf("Nile", "Amazon", "Yangtze")
// Using the += assignment operation creates a new riversArray,
// copies over the original elements and adds "Mississippi"
riversArray += "Mississippi"
println(riversArray.joinToString())
// Nile, Amazon, Yangtze, Mississippi
```

Kreiranje nizova

- Funkcijama: arrayOf(), arrayOfNulls(), emptyArray():

```
val arr = arrayOf(1, 2, 3)
val nullArray: Array<Int?> = arrayOfNulls(3)
var exampleArray = emptyArray<String>()
```

- Array konstruktor

```
// Creates an Array<Int> that initializes with zeros [0, 0, 0]
val initArray = Array<Int>(3) { 0 }
// Creates an Array<String> with values ["0", "1", "4", "9", "16"]
val asc = Array(5) { i -> (i * i).toString() }
```

Naredbe za kontrolu toka

- if/else, when, for, while
- Range – opseg (inkluzivni):
  - 1..100 – opseg od 1 do 100, uključujući 1 i 100

```
val numberOfStudents = 50
if (numberOfStudents in 1..100) {
    println(numberOfStudents)
}
```

Naredba when

- Slično switch/case naredbi

```
enum class Color {
    RED, GREEN, BLUE
}

when (getColor()) {
    Color.RED -> println("red")
    Color.GREEN -> println("green")
    Color.BLUE -> println("blue")
    // 'else' is not required
because all cases are covered
}
```

```
when (getColor()) {
    Color.RED -> println("red") // no
branches for GREEN and BLUE
    else -> println("not red") // 'else'
is required
}
```

For naredba

- For služi za iteraciju kroz bilo šta što ima iterator

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

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

for (i in 6 downTo 0 step 2) {
    println(i)
}
```

For naredba – iteriranje kroz listu pomoću indeksa

```
for (i in array.indices) {
    println(array[i])
}
for ((index, value) in array.withIndex()) {
    println("the element at $index is $value")
}
```

Null safety

- Promenljive ne mogu biti null osim ako se ne naznače kao nullable tip (npr. Int?, Array?)
- Testiranje da li je promenljiva null: operator **?**
- Bacanje null pointer exception: operator **!!**

Null safety

```
if (numberOfBooks != null) {
    numberOfBooks = numberOfBooks.dec()
}
```

- U Kotlin-u:

```
numberOfBooks = numberOfBooks?.dec()
```

Klase i objekti

prof. dr Bratislav Predić
dipl. inž. Nevena Tufegdžić

Kotlin
Razvoj mobilnih aplikacija i servisa

Klase u Kotlin-u

```
class House {
    val color: String = "white"
    val numberOfWindows: Int = 2
    val isForSale: Boolean = false

    fun updateColor(newColor: String){...}
    ...
}
```

```
val myHouse = House()
```

Konstruktori

- Primarni konstruktor se definiše u hederu klase (pri definiciji)
- Konstruktori (kao i sve funkcije) mogu imati default vrednosti parametara

```
class Box(val length: Int, val width:Int = 20, val height:Int = 40)
val box1 = Box(100, 20, 40)
val box2 = Box(length = 100)
val box3 = Box(length = 100, width = 20, height = 40)
```

Primeri

```
class A
```

```
val aa = A()
```

```
class B(x: Int)
```

```
val bb = B(12)
println(bb.x)
  // compiler error unresolved reference
```

```
class C(val y: Int)
```

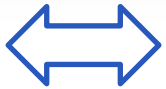
```
val cc = C(42)
println(cc.y)
  // 42
```

Primarni konstruktor

- Može se definisati u hederu klase zajedno sa init blokom
- U init bloku piše se kod za inicijalizaciju, koji se obavezno izvršava pri instanciranju objekta date klase
- Moguće je imati više init blokova, ali svi oni ulaze u telo primarnog konstruktora

Primarni konstruktor

```
class Circle(i: Int) {
    init {
        ...
    }
}
```



```
class Circle {
    constructor(i: Int) {
        ...
    }
}
```

Sekundarni konstruktori

- Klasa može imati više sekundarnih konstruktora
- Svi sekundarni konstruktori definišu se pomoću ključne reči constructor
- Svi sekundarni konstruktori moraju da pozivaju primarni konstruktor
- Sekundarni konstruktor ne mora da ima telo

Sekundarni konstruktori

```
class Circle(val radius: Double) {
    constructor(name: String) : this(1.0)
    constructor(diameter: Int) : this(diameter / 2.0) {
        println("in diameter constructor")
    }
    init {
        println("Area: ${Math.PI * radius * radius}")
    }
}
val c = Circle(3)
```

Class Properties

- Deklarišu se kao var ili val
- Pristup preko dot notacije (obj.property)
- Mogu imati custom get i set metode
- Ukoliko se get i set ne predefinišu, podrazumevaju se default getter i setter

Class Properties

```
var fullName:String = ""
    get() = "$firstName $lastName"
    set(value) {
        val components = value.split(" ")
        firstName = components[0]
        lastName = components[1]
        field = value
    }
person.fullName = "Jane Smith"
```

Nasleđivanje

- Podržano je samo jednostruko nasleđivanje klasa
- Nasleđivanje je javno – klasa nasleđuje sve metode i atribute nadklase, uključujući i one koje je nadklasa nasledila od svoje nadklase
- Private metode i atributi se ne nasleđuju, ali protected se nasleđuju
- Klase su default-no final – ukoliko klasa treba da bude nasleđena, koristiti ključnu reč *open* ispred definicije klase

Override-ovanje članova klase

- Ukoliko je nekog člana klase moguće predefinisati, ispred definicije člana mora da stoji open
- Pri predefinisanju člana klase, koristi se ključna reč override
- Ako je nešto u klasi A open, u nasleđenoj klasi B override, u klasi C koja nasleđuje klasu B moguće ga je predefinisati

Interfejsi

- Klasa može implementirati proizvoljan broj interfejsa
- Interfejsi mogu da nasleđuju druge interfejse
- Sadrži deklaracije metoda i imena atributa
- Primer:

```
interface NameOfInterface { interfaceBody }
```

Apstraktne klase

- Ključna reč abstract
- Nemoguće instancirati, moguće naslediti
- Za razliku od interfejsa, mogu da čuvaju *state*
- Svaki property ili metoda koje su obeležene ključnom rečju abstract moraju biti predefinisani
- Mogu imati neapstraktne properties i metode

Data klase

- Klasa koja je specijalizovana za skladištenje podataka
- Označava se ključnom rečju data
- Automatski generiše gettere i settere za svaki svoj property
- Automatski generiše toString(), equals(), hashCode(), copy()

```
data class <NameOfClass>( parameterList )
```

Literatura

- https://kotlinlang.org
- https://developer.android.com/kotlin/first
- https://www.w3schools.com/KOTLIN/index.php

Hvala na pažnji!