Android programozás
Kotlin nyelvi alapok - 1.

Ekler Péter
peter.ekler@aut.bme.hu

Ütemezés

- Programozási alapismeretek
- Környezet bemutatása
- Projekt felépítése, kód szervezés
- Elnevezési konvenciók, szerkesztési konvenciók
- A Kotlin nyelv kialakulása, hasonlóságok és különbségek a Java nyelvhez képest
- A Kotlin nyelv szintaxisa
- Konstansok, változók, elágazások, függvények, stb.
- Vezérlési struktúrák
- Kivételkezelés
- Osztályok, leszármaztatás
- Objektum, enumerációk, konstruktor típusok
- Egyszerű alkalmazások fejlesztése

Ütemezés

- Interfészek, abstract osztály
- Adatstruktúrák, listák kezelése
- Típusok jelentése, típus konverzió
- Nullable típusok
- Szöveges adatok kezelése
- Függvények szerepe és lehetőségei
- Lambdák
- Extension nyelvi elem
- Függvény paraméterek
- Operátorok felüldefiniálása
- Komplex vezérlési struktúrák
- Szálkezelés
- Aszinkron nyelvi elemek
- További példák

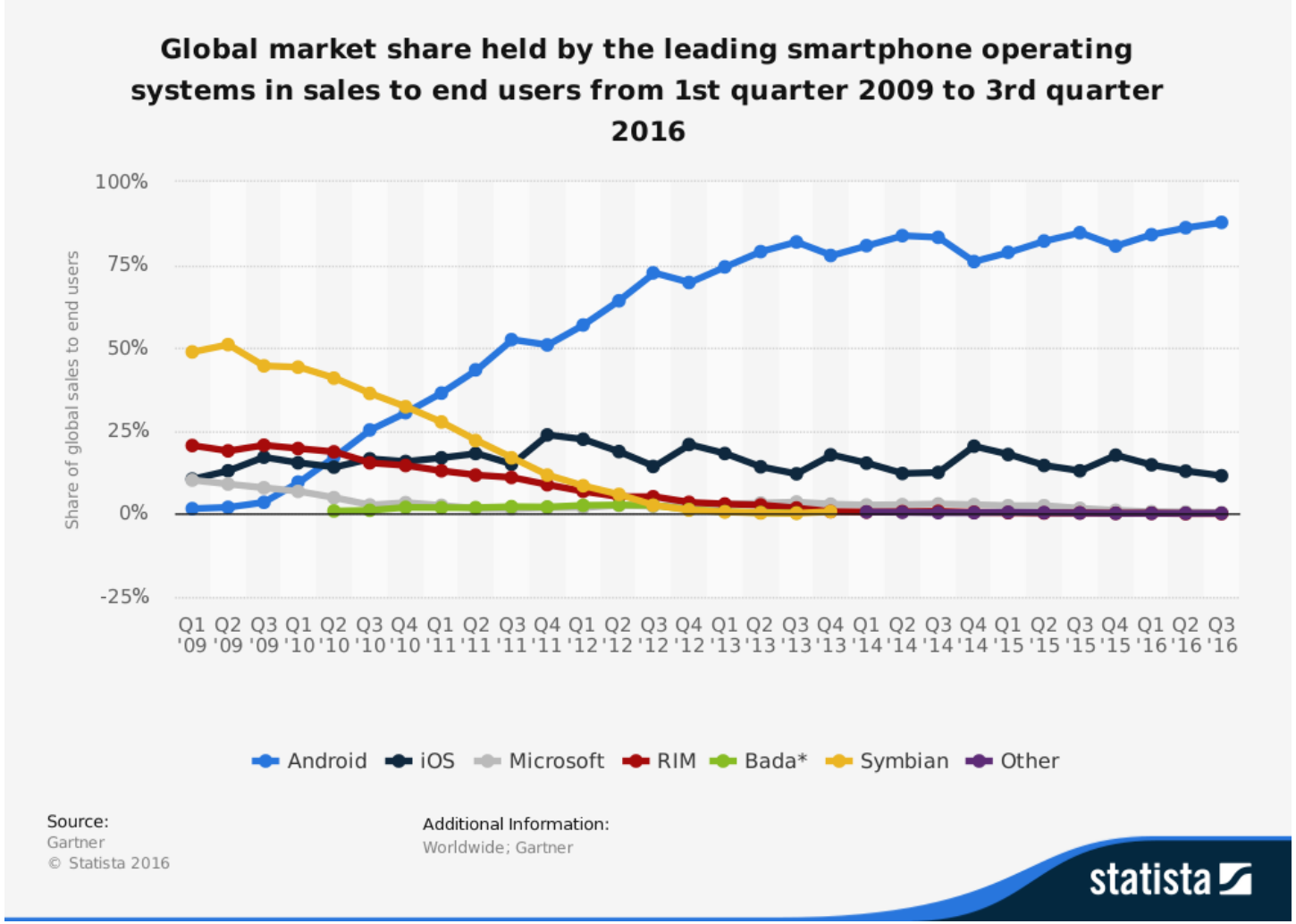
Tartalom

- Fejlesztőkörnyezet bemutatása
- Első projekt létrehozása
- Programozási alapok
- Kotlin nyelvi alapok
- Konstansok, változók
- Tömbök

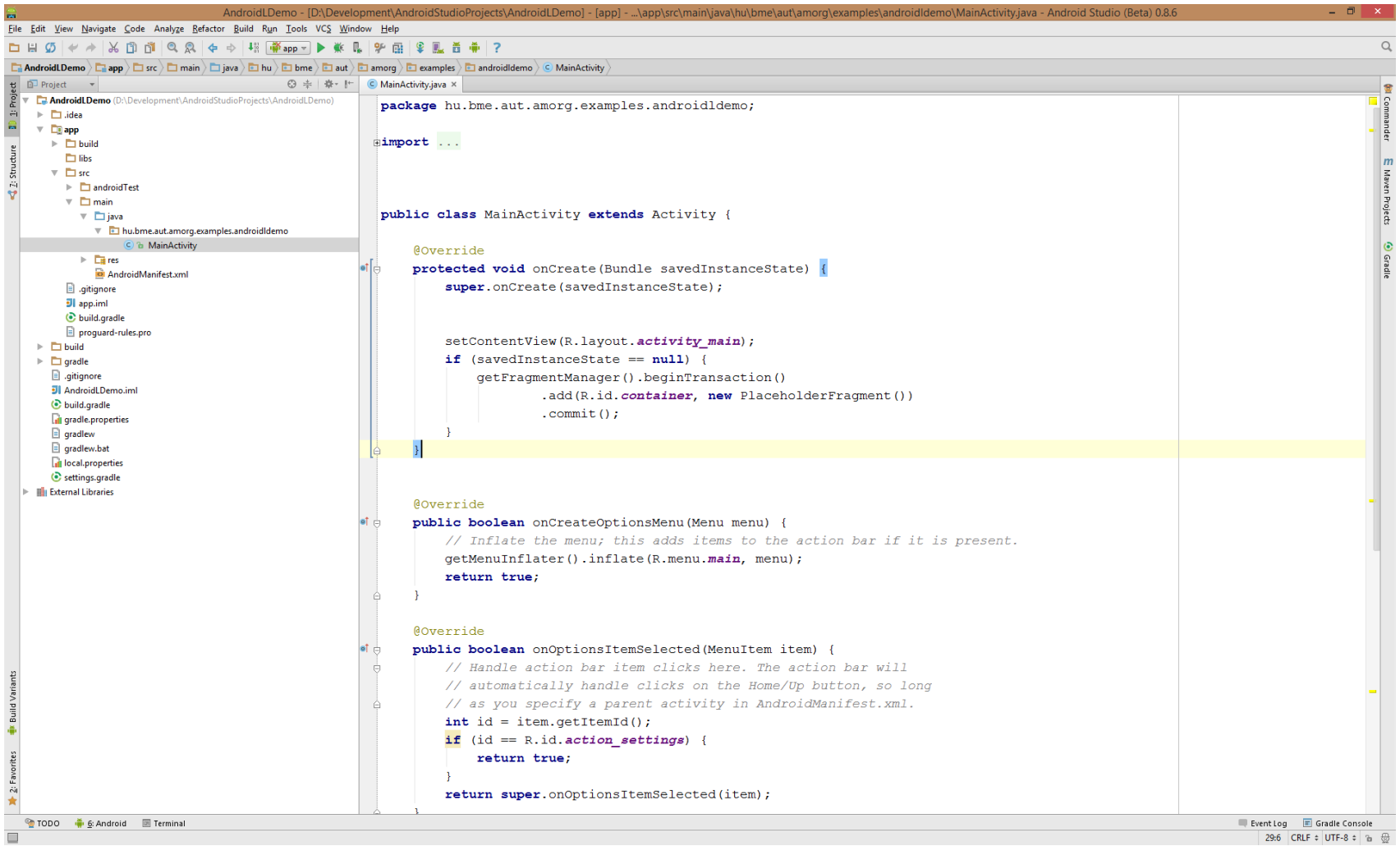
Kurzus példái:
https://github.com/peekler/AndroidProgrammingBasics

Android fejlesztőkörnyezet

Android elterjedtsége



Fejlesztő eszköz: Andorid Studio



Telepítési útmutató:
https://bit.ly/2V4eXEB

Emulátor

- Teljes operációs rendszer emulálása (lassú)
  - Beépített alkalmazások elérhetők
  - Ctrl+F11 (screen orientáció állítás)
- Alternatíva: Genymotion emulátor (https://www.genymotion.com/)

Emulátor elérése konzolról

- Csatlakoztatott emulátorok/eszközök listázása:
  - adb devices
- Shell elérése
  - adb shell
- Csatlakozás telneten keresztül:
  - Indítsunk telnet klienst
  - o localhost 5554
- SMS küldése:
  - sms send <küldő száma> <üzenet>
- Hanghívás
  - gsm call <hívó száma>

Debugolás folyamata

- On-device debug teljes mértékben támogatott
  - Megfelelő USB driver szükséges!
  - Készüléken engedélyezni kell az USB debugolást
- Minden alkalmazás önálló process-ként fut
- Minden ilyen process saját virtuális gépet (VM) futtat
- Minden VM egy egyedi portot nyit meg, melyre a debugger rácsatlakozhat (8600, 8601, stb.)
- Létezik egy úgynevezett „base port” is (8700), mely minden VM portot figyel és erre csatlakozva az összes VM-et debugolhatjuk

Első Android projekt – Felület létrehozása

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:orientation="vertical"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <Button
        android:id="@+id/btnDemo"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Demo"/>

    <TextView
        android:id="@+id/tvHello"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Hello World!" />

</LinearLayout>
```

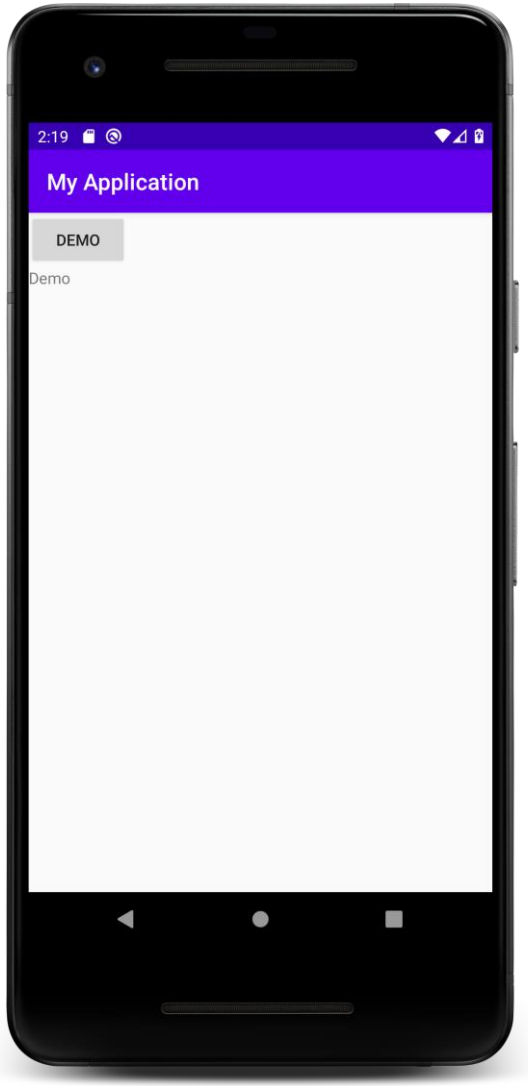
Egyedi ID

Első Android projekt – Kotlin kód

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btnDemo.setOnClickListener {
            tvHello.text = "Demo"
        }
    }
}
```

Felület beállítása

Egyedi ID-k használata és eseménykezelő beállítása



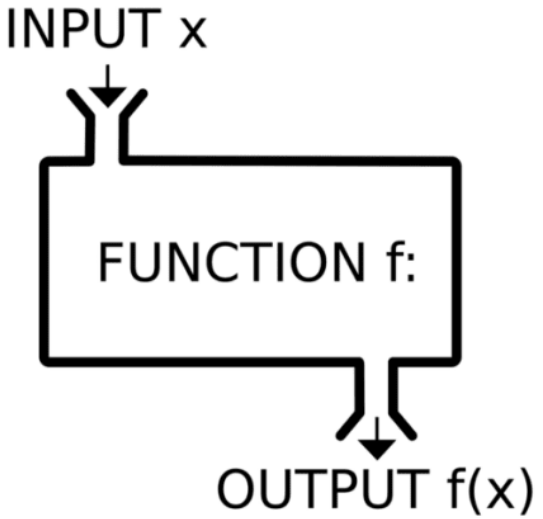
Érdekesség: Függvény mint paraméter

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        btnDemo.setOnClickListener(::demoClick)
    }

    fun demoClick(view: View) {
        tvHello.text = "Demo"
    }
}
```

Programozási alapok

Programozás alap elemei

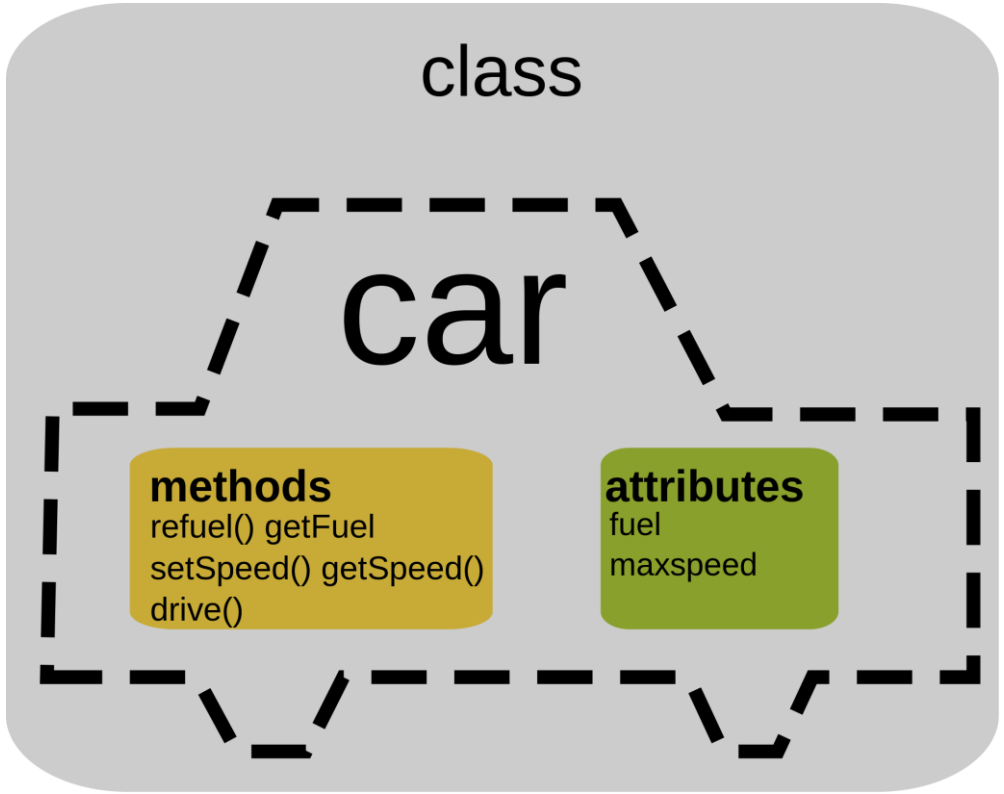
- Konstansok
- Változók
- Típusok
- Speciális értékek: null
- Értékek
- Függvények
- Objektum Orientáltság
  - Osztályok
  - Objektumok
  - Leszármaztatás
  - …



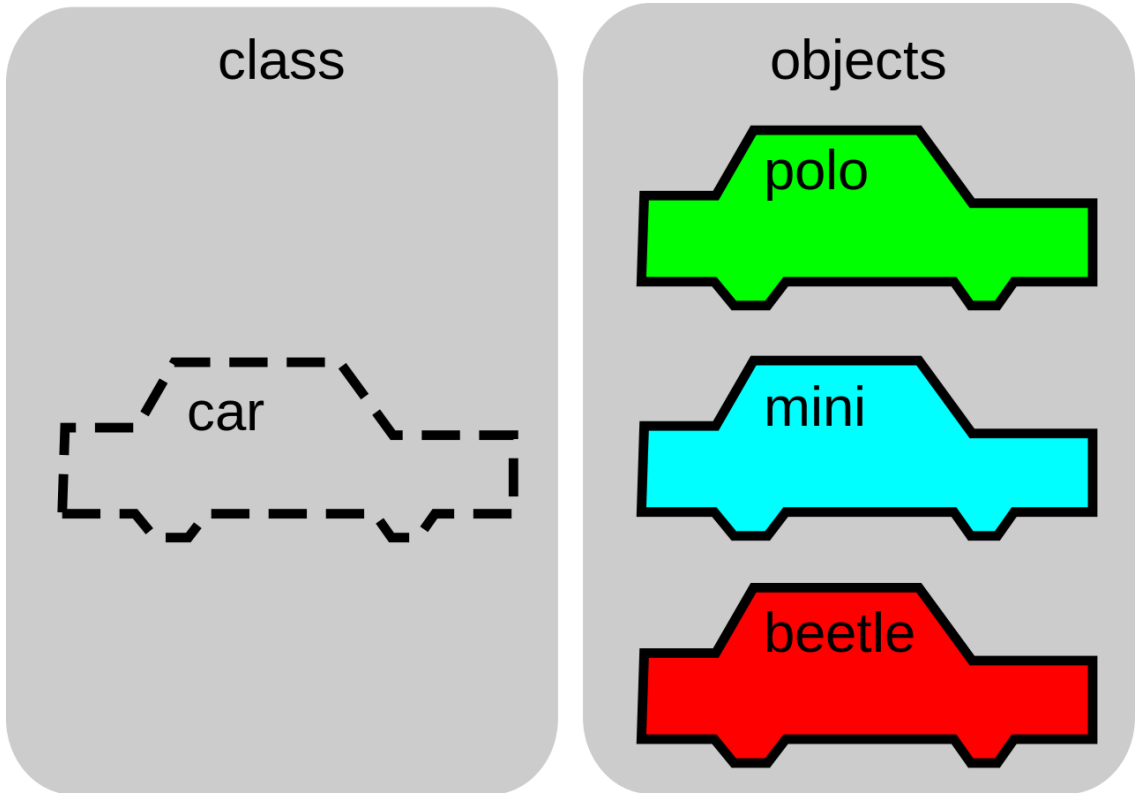
Források:
https://compscihelp.com/java/1-1.php
https://dev.to/navi/why-functional-programming-matters-2o95

Osztályok, objektumok

- Osztály:



- Objektum:



Források:
https://www.miltonmarketing.com/coding/programming-concepts/object-oriented-programming-oop/
https://commons.wikimedia.org/wiki/File:CPT-OOP-objects_and_classes.svg

Kotlin nyelvi alapok

Forrás: https://kotlinlang.org/docs/reference/

Mire utal a Kotlin név?

A. Semmire, csak egy kitalált szó
B. Egy lengyel falu nevére
C. Egy sziget nevére
D. Key Object Tool Language INsight rövidítése

Mire utal a Kotlin név?

A. Semmire, csak egy kitalált szó
B. Egy lengyel falu nevére
C. Egy sziget nevére
D. Key Object Tool Language INsight rövidítése

Kotlin története

- 2011-ben jelent meg először
- JetBrains gondozásában
- Nyílt forráskódú: 2012
- v1.0 verzió: 2016
- 2017-es Google I/O: hivatalos támogatás Androidra
- Statikusan típusos
- Objektum orientáltság mellett a funkcionális programozást is támogatja

A Kotlin főbb jellemzői

- JVM (Java Virtual Machine) byte kódra (vagy akár JavaScriptre is) fordul
- Meglévő Java API-k, keretrendszerek és könyvtárak használhatók
- Automatikus konverzió Java-ról Kotlinra
- Null-safety
  - Vége a NullPointerException korszaknak
- Kód review továbbra is egyszerű
  - A nyelv alapos ismerete nélkül is olvasható a kód

Miért érdemes Kotlinba programozni?

40%-al kevesebb kód mint Java-ban

Olvasható kód

Többféle lehetőség: Támogatja az objektum orientált és a funkcionális programozást egyaránt

Kevesebb futás idejű NullPointerException

Nincs boilerplate kód

Írhatunk Kotlin kódot meglévő Java projektekbe is

Kedvenc Java osztálykönyvtár ak továbbra is használhatók

Kód hosszúság Java7 vs Java8 vs Kotlin - *CaesarCipherDecoder*

```kotlin
class CaesarCipherKotlinDecoder(val firstLetterInAbc: Char, val lastLetterInAbc: Char) : CeaserDecoder {
    val alphabet: CharArray = CharArray(lastLetterInAbc - firstLetterInAbc + 1) { it -> it.plus(firstLetterInAbc.toInt()).toChar() }
    override fun decode(text: String, n: Int): String {
        return String(text.toCharArray().map { alphabet[calculateDecodedPosition(it, n)] }.toCharArray())
    }
    override fun calculateDecodedPosition(character: Char, n: Int): Int {
        val characterIndex = alphabet.indexOf(character);
        var decodedIndex = characterIndex - n
        if (decodedIndex < 0) decodedIndex = alphabet.size + decodedIndex
        return decodedIndex;
    }
}
```

Kotlin: 12 lines

```java
public class CaesarCipherJavaDecoder implements CeaserDecoder {
    private char firstLetterInAbc;
    private char lastLetterInAbc;
    private List<Character> alphabet = new ArrayList<Character>();
    public CaesarCipherJavaDecoder(char firstLetterInAbc, char lastLetterInAbc) {
        this.firstLetterInAbc = firstLetterInAbc;
        this.lastLetterInAbc = lastLetterInAbc;
        generateAlphabet(firstLetterInAbc, lastLetterInAbc);
    }
    private void generateAlphabet(char firstLetterInAbc, char lastLetterInAbc) {
        int size = lastLetterInAbc - firstLetterInAbc + 1;
        for (Integer i = 0; i < size; i++) {
            char newLetter = (char) (i + (int) firstLetterInAbc);
            alphabet.add(newLetter);
        }
    }
    @Override
    public String decode(String s, Integer n) {
        char[] chars = s.toCharArray();
        char[] decodedChars = new char[chars.length];
        for (int i = 0; i < decodedChars.length; i++) {
            int decodedPosition = calculateDecodedPosition(chars[i], n);
            decodedChars[i] = alphabet.get(decodedPosition);
        }
        return String.valueOf(decodedChars);
    }
    @Override
    public int calculateDecodedPosition(char character, int n) {
        int characterIndexInAbc = character - firstLetterInAbc;
        int decodedIndex = characterIndexInAbc - n;
        if (decodedIndex < 0) decodedIndex = alphabet.size() + decodedIndex;
        return decodedIndex;
    }
}
```

Java7: 33 lines

```java
public class CaesarCipherJava8Decoder implements CeaserDecoder {
    private char firstLetterInAbc;
    private char lastLetterInAbc;
    private List<Character> alphabet;
    public CaesarCipherJava8Decoder(char firstLetterInAbc, char lastLetterInAbc) {
        this.firstLetterInAbc = firstLetterInAbc;
        this.lastLetterInAbc = lastLetterInAbc;
        generateAlphabet(firstLetterInAbc, lastLetterInAbc);
    }
    private void generateAlphabet(char firstLetterInAbc, char lastLetterInAbc) {
        Stream<Character> alphabetStream = IntStream.range(0, lastLetterInAbc - firstLetterInAbc + 1).mapToObj(i -> (char) (i + (int)
firstLetterInAbc));
        alphabet = alphabetStream.collect(Collectors.toList());
    }
@Override
    public String decode(String text, Integer n) {
        Stream<Character> s = text.codePoints().mapToObj( c -> alphabet.get(calculateDecodedPosition((char)c, n)));
        return s.map(String::valueOf).collect(Collectors.joining());
    }
    @Override
    public int calculateDecodedPosition(char character, int n) {
        int characterIndexInAbc = character - firstLetterInAbc;
        int decodedIndex = characterIndexInAbc - n;
        if (decodedIndex < 0) decodedIndex = alphabet.size() + decodedIndex;
        return decodedIndex;
    }
}
```

Java8: 24 lines

Kotlin kód szerkesztő

- Teljes Kotlin támogatás: AndroidStudio, IntelliJ és IntelliJ Community Edition
- REPL (Read Eval Print Loop)
  - Kotlin fordító/futtató (interpreter)
  - Megnyitás pl. Android Studio-ból: Tools -> Kotlin -> Kotlin REPL
  - Futtatás: CTRL + Enter
- Java->Kotlin automatikus átalakítás

Változók és konstansok

- Változtatható érték, deklaráció kulcsszava: **`var`**

```
var a: Int = 10
a = 20
```

Nincs szükség ‚;'-re a sor végén

```
var a: Int = 10; a = 10
```

‚;' használható sorok belüli elválasztásra

- Nem változtatható érték, deklaráció kulcsszava: **`val`**

```
val a: Int = 10
a = 20
```

Fordítási hiba: *„Val cannot be reassigned"*

Változók – Nem kötelező a típus megadása

```
var a = 10.1
```

A típust a fordító a kontextus alapján automatikusan kitalálja

Típusok

- Minden valójában objektum -> hívhatók függvények és lekérhetők tulajdonságok a változókon

```
1.toLong()
```

- Némelyik típus (számok, karakterek, boolean) futási időbe primitív típusként vannak reprezentálva meg, de osztályokként látszanak

Szám típusok

Típus	Bit méret
Double	64
Float	32
Long	64
Int	32
Short	16
Byte	8

A karakterek nem tartoznak a szám típusok közé (Java-val ellentétben)

Szám kifejezések

Típus	Kifejezés
Byte, Short, Int, Long	3, 1_200
Long	3L, 1_200L
Hexadecimal (Lehet :Byte, Short, Int, Long)	0xAF, 0XAF, 0xFF_EC_DE_5E
Binary (Lehet: Byte, Short, Int, Long)	0b0101, 0B0101, 0b01_01
Double	1.23, 1.23e10, 1.2_66
Float	1.2F, 1.2f, 1.2_66f

Kasztolás (típus váltás)

```
val b: Byte = 1
val c: Int = b
```

Fordítási hiba: *Type mismatch: inferred type is Byte but Int was expected*

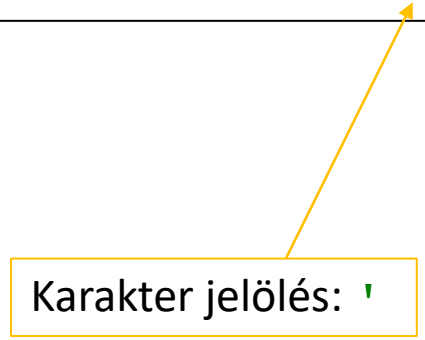
- Implicit kasztolás nem engedélyezett (mivel tipikus hibaforrás szokott lenni)
  - Például: integer változóhoz nem lehet byte értéket hozzárendelni

- Use explicit casting:

```
val c: Int = b.toInt()
```

Karakterek

```
var someCharacter: Char = 'a'
```



Boolean típus

```
var someBoolean: Boolean = true
var otherBoolean = false
```


```
someBoolean && otherBoolean || someBoolean && !otherBoolean
```

Tömbök

```
public class Array<T> {
    /**
     * Creates a new array with the specified [size], where each element is calculated by
calling the specified
     * [init] function. The [init] function returns an array element given its index.
     */
    public inline constructor(size: Int, init: (Int) -> T)
…}
```

- Típusos tömbök

Tömb létrehozás (utility function)

Tömb létrehozás az Array osztály konstruktorával

```
var myIntArray = intArrayOf(1, 2, 3)
var myIntArray2 = Array<Int>(3, {it * 1})

println(Arrays.toString(myIntArray))
println(Arrays.toString(myIntArray2))
println(myIntArray[2])
[1, 2, 3]
[0, 1, 2]
3
```

it: az aktuális elem indexe

Hozzáférés az elemekhez a [] operátorral lehetséges

Tömbök megjelenítése a Java `Arrays.toString` függvénnyel

- Vegyes típusokat tartalmazó tömb

```
var mixedArray = arrayOf(1, "something")
println(Arrays.toString(mixedArray))
[1, something]
```

String (szöveg) típus

A szövegek a `String` típussal (osztály) kezelhetők

```
var someString: String = String(charArrayOf('m', 'y', ' ', 's', 't', 'r', 'i', 'n', 'g'))
```

Minden String valójában karakterek tömbje

```
var someOtherString = "my string"
```

String készítés egyszerűen a `"..."`-el lehetséges

String minták (template)

- A String template-k $ jellel kezdődnek
- Kirétékelésre kerülnek és az értékek egy nagy stringé konkatenálódnak
- A templatek tartalmazhatnak:

Egyszerű változókat

```
val nrOfProgrammers = 10
print("There are $nrOfProgrammers in the room")

There are 10 in the room
```

Komplex kifejezéseket

```
val nrOfProgrammers = 10
val nrOfManagers = 12
print("There are ${nrOfProgrammers + nrOfManagers} man in the room")

There are 22 man in the room
```

String Literal Ttípusok

Escapelt stringek

- Tartalmazhatnak escape karaktereket: \t, \b, \n, \r, \', \", \\ és \$
- Elválasztó: idézőjel (")
- Például:

```
var escapedString = "Hello!\nThis is me!"
```

- Tartalmazhat template-ket

Nyers stringek

- Tartalmazhat újsor és más nem-escapelt karaktereket
- Elválasztó: (""")
- Pl:

```
var rawString = """Hello!
    |This is me!
""".trimMargin()
```

- A *trimMargin* használható a kezdő üres spacek és | jel előtti spacek eltávolítására, valamintaz első és az utolsó üres sor törlésére

```
print(rawString)
Hello!
This is me!
```

- Tartalmazhat templateket

== operátor

```
var s1 = "hello"
var s2 = "hello"
println(s1 == s2) //eredmény: true
```

A “==” operátor használható bármilyen objektum összehasnlítására. Valójában egy .equals() függvényt fog hívni ha a == mindkét oldalán egy nem null értékű objektum áll.

Gyakoroljunk – tömbök, listák

- Rendezés
- Véletlen sorrend
- Műveletek tömbökkel
- Keresés
- Egyéb műveletek

Köszönöm a figyelmet!

Ekler Péter
peter.ekler@aut.bme.hu