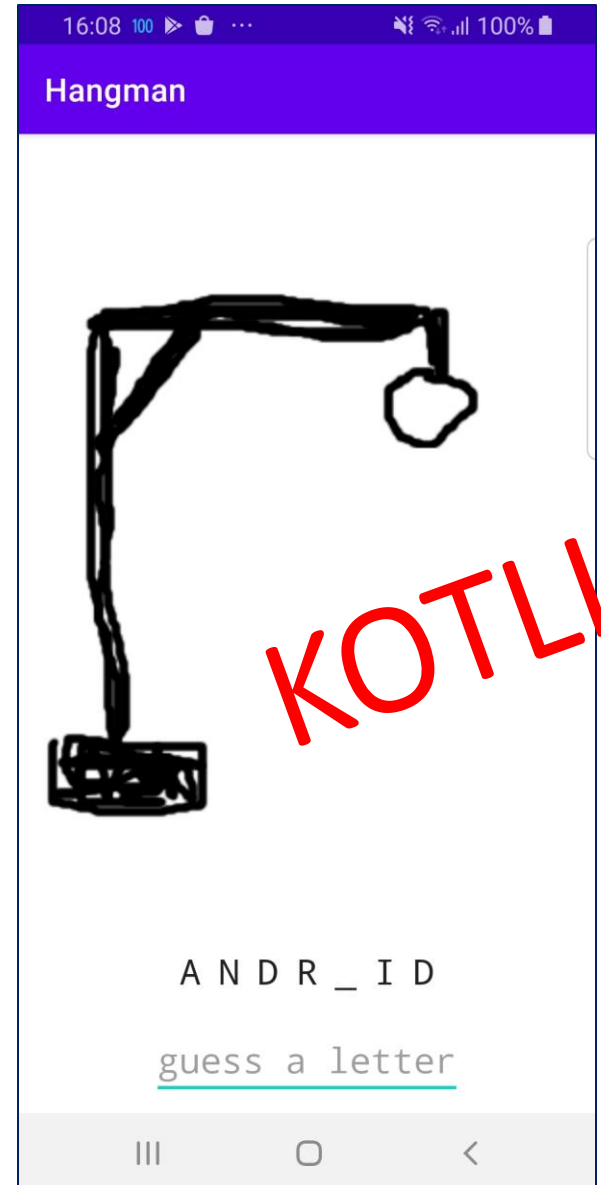


Hangman

Aplikácia, ktorú sme už raz robili.
Ale tentokrát v Kotlin.



Kotlin

- **Android je Kotlin-first** od Google I/O 2019
- Kotlin od 2011, Java od 1995
- Funguje spolu s Javou
- Menej kódu, lepšia čitateľnosť, bezpečnejší kód
- Android, iOS backend, web apps



Kotlin na prvý pohľad

- Menej ukecané (POJO, Data Class)
- Syntax - žiadna bodkočiarka
- Typy premenných – nie vždy explicitne povedané
- == vs === (v Java equals a ==)
- String literály (a mnoho iného neskôr)
- Nový objekt bez new

```
public class Student {  
    private int rollNo;  
    private String firstName;  
    private String lastName;  
  
    public int getRollNo() {  
        return rollNo;  
    }  
  
    public void setRollNo(int rollNo) {  
        this.rollNo = rollNo;  
    }  
  
    public String getFirstName() {  
        return firstName;  
    }  
  
    public void setFirstName(String firstName) {  
        this.firstName = firstName;  
    }  
  
    public String getLastName() {  
        return lastName;  
    }  
  
    public void setLastName(String lastName) {  
        this.lastName = lastName;  
    }  
}
```

```
data class Student(val rollNumber: Int, var firstName: String, val lastName: String)
```

Rozširovanie tried

- Aj v Jave vznikali triedy **iba** rozširovaním
- Java: Object
- **Kotlin: Any** (equals, hashCode, toString)
- Rovnako sa implementuje aj interface (viac rozhraní oddelíme čiarkou)
- Súvisiace veci na neskôr: dedičnosť a keyword open, deklarovanie konštruktora

Definovanie metód

- Funkcia **fun** keyword
- **Override** už nie iba ako anotácia
- Definovanie premenných
 - Názov: Typ
 - Default arguments (**a: Int = 0**)
 - Named arguments - pri zavolaní funkcie
- Návratový typ
 - Píše sa na konci
 - Java – void, Kotlin – Unit (**nemusí sa písať**)
 - Unit – typ s jedinou hodnotou Unit

Null safety ?

- Menšia početnosť **NullPointerException** a viac pod kontrolou
- premenná **surprise** - nemôže byť null
(kompilačná chyba ak by sme sa o to pokúsili)
- premenná **surprise?** - môže byť null
 - volanie `surprise.length` nepovolené
- **Safe call - ?.**
surprise?.length vráti hodnotu alebo null
- Budeme sa tomu venovať podrobnejšie neskôr ...

premenné

- **var** - mutable
- **val** – read-only
- Typy premenných
 - Kotlin nepoužíva primitívne typy (iba interne po kompilácii)
- Int, Long, Double, Boolean, Char, String ...
- Nie vždy je potrebné písať typ premennej

Companion object

- Kotlin **class** - pre triedy, ktoré môžu mať viac inštancií
- Kotlin **object** – pre singletony
- **companion object**
 - pridružený k nejakej triede
 - Funkcie a premenné - ako static v Java
- **Const** – pre hodnoty známe v čase kompilácie, ktoré sa nebudú meniť
 - Len pre Stringy a *primitívne typy*

Konštruktory

- **1 primárny** v hlavičke triedy
- keyword **constructor** len pri zmene viditeľnosti
`class Person(firstName: String)`
- Primárny konštruktor nemá kód - **init** blok (aj na viacerých miestach - vykonáva sa postupne)
- Ľubovoľne veľa **sekundárnych** (+ keyword)

Kolekcie

- List – iba immutable operácie
- MutableList a Array poskytujú mutable operácie
- **Array** (zodpovedá poľu v java)
 - arrayOfNulls()
 - arrayOf()
 - IntArray, intArrayOf()
- Veľa užitočných funkcií v porovnaní s Javou
- Lambda a funkcionálny prístup

cykly

- Keyword **in** (ako for-each)

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

```
for (i in 1..3)
```

```
for (i in array.indices)
```

```
for (i in 6 downTo 0 step 2)
```

Single expression funkcie

- Ak funkcia **vráti iba výraz**, môže byť vrátený priamo cez = namiesto bloku kódu

```
fun double(x: Int): Int = x * 2
```

- Návratový typ je **voliteľný** - ak to vie kompilátor doplniť
- Funkcie (s blokom kódu) musia definovať návratový typ, inak sa doplní Unit

Stringy a porovnavanie

== VS ===

- **Structural** equality
== porovnáva objekty (ako equals v java)
- **Referential** equality
=== porovnáva referencie (ako == v java)