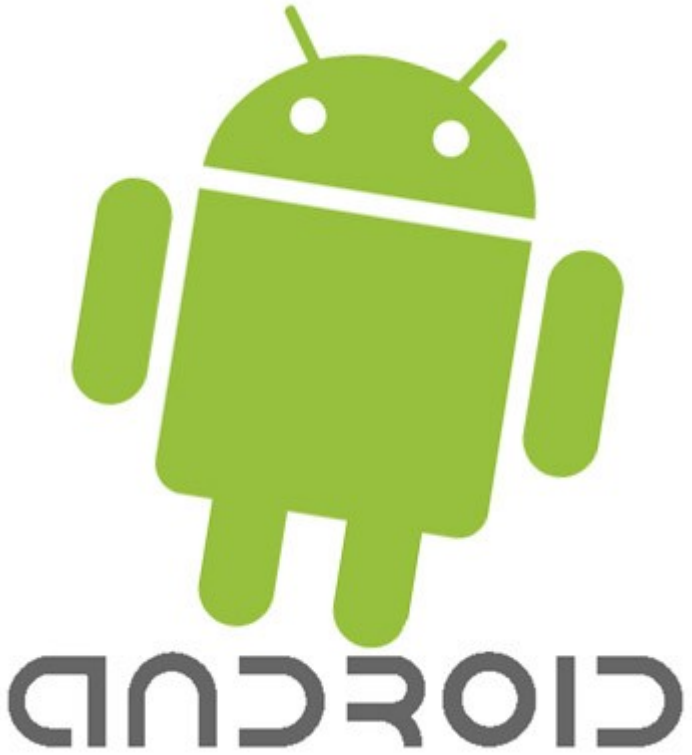

Aggiornamenti Google I/O 2019

Kotlin e Android

Kotlin & Android

- Due anni fa, Google ha introdotto **Kotlin** come linguaggio sperimentale per Android
- Tre giorni fa, Google ha dichiarato che **Kotlin** è adesso il **linguaggio di riferimento** per lo sviluppo su Android
  - Java e C/C++ ancora supportati con parità di feature
    - Ma Kotlin è quello “figo”, e sarà il più testato
    - Il 50% degli sviluppatori Android già usa Kotlin come linguaggio principale
  - Motivazioni multiple:
    - Linguaggio più moderno e più comodo rispetto a Java
    - Diminuire ulteriormente la dipendenza di Android da tecnologie proprietarie o comunque non controllate da Google (e ridurre il rischio-cause)
      - Come per Dalvik / ART al posto della JVM

Kotlin: sintassi base

- Kotlin è **quasi** una diversa sintassi per Java
  - Notazione dei tipi “stile Pascal”
  - int send(String act) { … } → **fun** send(act: String): Int { … }
  - Tipi nullabili: String? può essere null, String no, String non si sa
    - if (p!=null) p.m(x) → p?.m(x)
    - r=(p!=null?p.f:dflt) → p?.f ?: dflt
    - x=a?.b?.c?.d
  - Distingue le variabili “vere” dalle costanti letterali
    - **var** x:Int – x è una variabile
    - **val** x:Int – x può essere assegnata solo 1 volta
  - Stringhe template:
    - “x vale $x” oppure “il conto fa ${f(x)+3}”

Kotlin: sintassi base

- Kotlin è **quasi** una diversa sintassi per Java
  - Assai migliore **inferenza di tipi**
    - Riduce la necessità di cast, migliora i controlli del compilatore
  - **Data class** per oggetti semplici
    - data class Punti(val giocatore:String, var punti:Int)
    - Crea in automatico costruttori, getter, setter, toString(), ecc.
  - Singleton nel linguaggio
    - object me { val nome="Vincenzo" }

Kotlin: lambda

- Il supporto alle lambda-expressions semplifica la scrittura di listener, runnable, ecc.
  - x -> x*2, oppure x,y -> { f(x); g(x,y); }
  - val sum = { x: Int, y: Int -> x + y }
- Un blocco di codice fra graffe è implicitamente una funzione con un solo parametro *it*
  - list.map { 2*it } // si possono omettere le ()
- Grazie alla type inference, spesso i tipi possono essere omessi

Kotlin: classi sealed e extension

- Le classi in Kotlin sono per default **sealed**
- Le classi destinate a essere sottoclassate devono essere dichiarate **open**
  - open class Padre …
  - class Figlio : Padre …
- Idem per l'override di metodi (e di proprietà)
  - open fun m() {} // nel Padre
  - override fun m() {} // nel Figlio
- Rationale: evitare l'overriding accidentale

Kotlin: classi sealed e extension

- È possibile **aggiungere metodi e proprietà** a classi già esistenti (extension)
  - fun Padre.n(…) {…}
  - In tutto lo scope di visibilità di una simile dichiarazione, gli oggetti di classe Padre avranno un metodo in più (n) rispetto a quelli definiti dalla dichiarazione della classe

Kotlin: delegation

- Un pattern tipico alternativo all'ereditarietà è la **delega**
  - **public class A {…}**
    public class B {
    A a = new A();
    public void m1(args) { a.m1(args); }
    public int m2(args) { a.m2(args); }
    …
    }
  - **class B(a: A): A by a**

Kotlin: coroutines

- Le coroutines sono un modo alternativo (rispetto ai thread) per implementare la concorrenza
- Si definiscono funzioni o lambda con la keyword **suspend** che possono interrompere l'esecuzione e passare il controllo allo scheduler delle coroutine
- Il qualificatore **async** identifica un blocco di codice che può essere eseguito in maniera asincrona
- Meccanismo simile alle async/await in Javascript

Kotlin: Androidismi

- Il compilatore Kotlin in Android Studio ha un po' di plug-in pensati per facilitare la vita al programmatore Android
- Esempi
  - Estrazione degli ID delle View in un layout (*synthetic properties)*
    - String te = ((TextEdit)findViewById(R.id.textedit)).getText();
    - import kotlinx.android.synthetic.main.<*layout*>.*
      var te=textedit.getText();
  - Trasformazione sintattica di codice (o intere classi) da Java a Kotlin
    - … che a volte funziona anche!

Kotlin: Andoidismi

- Es: accesso a SQLite via annotazioni

```
@Table(name="users", database = AppDatabase::class)
class User: BaseModel() {

    @PrimaryKey(autoincrement = true)
    @Column(name = "id")
    var id: Long = 0

    @Column
    var name: String? = null
}
```

O anche:

```
@Table(database = KotlinDatabase::class)
data class User(@PrimaryKey var id: Long = 0, @Column var name: String? = null)
```

Kotlin: Java interop

- Il codice Kotlin è compilato sulla JVM
- Può interagire con classi e librerie Java in maniera trasparente
  - Chiamare Java da Kotlin
  - Chiamare Kotlin da Java
  - Accedere ai campi di oggetti in entrambe le direzioni

Sommario

- In definitiva, Kotlin:
  - È un linguaggio moderno
  - Orientato alla produttività
  - Focalizzato sulla prevenzione degli errori
  - Con customizzazioni per Android

Vale la pena dargli un'occhiata!

Kotlin
https://kotlinlang.org