Android 2

Kotlin

Sommaire

- Bases
- Variables
- Fonctions
- Classes
- Contrôles
- Aller plus loin
- Tips

Bases

Bases

- Langage proche du Swift d’iOS
- Pas de point-virgule en fin d’expression
- Inter-opérable avec Java
  - On peut avoir des classes en Kotlin et en Java dans le même projet.
- Pas de types primitif
  - int, double, bool... => Int, Double, Boolean...
- Instanciation simple
  - Java : Pokemon pikachu = new Pokemon()
  - Kotlin: val pikachu = Pokemon()
- Pleins de raccourcis à apprivoiser.

Variables

Variables - Déclaration

Java

- private Pokemon pikachu;

Kotlin

- private var pikachu: Pokemon
  - Variable modifiable
  - pikachu = Pokemon() => OK
- private val pikachu: Pokemon
  - Variable non modifiable
  - pikachu = Pokemon() => KO

Variables - Null safety

- Une variable **ne peut pas** être nulle
  - Initialisée à la déclaration
  - var pikachu: Pokemon = Pokemon()
- Une variable **ne peut pas être** nulle mais est **initialisée plus tard**
  - lateinit var pikachu: Pokemon
  - Non initialisée à la déclaration
  - Ne fonctionne pas avec val
- Une variable **peut être nulle**
  - Pas forcément initialisée à la déclaration
  - var pikachu: Pokemon?
- Une variable **peut être nulle** mais on sait qu'elle ne l'est pas à un **moment précis**
  - pikachu!.attacks

Fonctions

Fonctions - Déclaration

Java

- Pokemon getPokemon(int id){...}

Kotlin

- fun getPokemon(id: Int): Pokemon{...}

Fonctions - Particularités

- Pas de type de retour
  - fun updateName(): Unit {...}
  - fun updateName() {...}
- Fonction simple
  - fun heal(life: Int, heal: Int): Int = life + heal
  - fun heal(life: Int, heal: Int) = life + heal

Classes

Classes - Déclaration

Java

```
public class Pokemon{

    Pokemon(int level, Type type){
        this.level = level;
        this.type = type;
    }

}
```

Kotlin

```
class Pokemon(var level: Int, var type:
Type)

class Pokemon(var level: Int, var type:
Type){
    init{
        ...
    }
}
```

Pas de getter/setter en Kotlin de base

Classes - Héritage

- Non instanciable et héritable
  - abstract class Pokemon(level: Int, type: Type)
- Instanciable et héritable
  - open class Pokemon(level: Int, type: Type)
- Surcharge
  - Les fonctions et variables surchargeables doivent être déclarée open/abstract
  - override
- Pas de static !

```
companion object{
        var ....
        fun ...
}
```

Classes - Hériter et Implémenter

Java

- abstract class Base{...}
- interface OnClic{...}

class Derived extends Base implements Onclic

Kotlin

- open class Base{...}
- interface OnClic{...}

class Derived: Base, Onclic

Contrôles

Boucles

Java

```
for( i  ; i < 10 ;  i++){
        val pokemon = pokemons[i]
        pokemon.evolve()
}
```

Kotlin

```
for(i in 0..10){
        val pokemon = pokemons[i]
        pokemon.evolve
}

for(pokemon in pokemons){
        pokemon.evolve
}
```

Les boucles while et do..while fonctionnent de la même manière en Java et Kotlin

When

Java

```
switch(index){
        case 1:
                println(“first pokemon is” +
pokemon.get(1));
                break;
        case 2:
                ...
                break;
}
```

Kotlin

```
when(index){
        1 -> println(“first pokemon is
${pokemon[1]}”)
        2 -> ...
        else-> ...
}
```

En kotlin, on peut faire des when sur tout et n’importe quoi (ou presque)

Aller plus loin

Extensions

- Ajoute une fonction ou une variable sur une classe que vous ne voulez pas hériter

```
fun String.removeSpaces(){
    this.replace(“ “, “”)
}
```

```
var someString = “I don’t want any spaces here.”
someString.removeSpaces() // contient maintenant “Idon’twantanyspaceshere.”
```

Tips

Tips

- **Android Studio (AS) à toujours (ou presque) raison**
- AS convertit les fichiers Java en Kotlin
  - Attention ! La conversion n’est pas toujours parfaite
- Copier du code Java dans une classe Kotlin AS le convertit (presque) automatiquement
- https://kotlinlang.org/docs/reference/