```
@Sun
try {    null, 🪁?, 🐟 as 🐠, (a, b) = 👜
                                } catch(e: Exception)
```

Kotlin

Macha DA COSTA

Kotlin Avancé

- Gestion de variable nulle, *Null safety*
- Conversion de type
- Valeur multiple ~~*Tuple*~~
- Déconstruire les valeurs
- Gestion des exceptions
- Déclaration de constante
- Annotation

Kotlin Avancé

Gestion de variable nulle, *Null safety*

📎 Rien n'est *null*

❌
```kotlin
var s1:String = null // Ne compile pas !
```

📎 C'est quand même possible

✅
```kotlin
var s2:String? = null // «?» signifie peut être nul
```

📎 Vérification avant compilation

❌
```kotlin
s2.length
```

📎 Il est peut être *null*

✅
```kotlin
s2?.length // Ça revient à if(s2 != null) {s2.length}
```

📎 Il est déjà initialisé

✅
```kotlin
s2!!.length // Si c'est pas le cas -> Exception NullPointerException
```


Référence [19] : https://antonioleiva.com/nullity-kotlin/

Kotlin Avancé

Initialisation avec classe déléguée

```
val mStringArray: Array<String> by lazy
{ resources.getStringArray(R.array.motivation_quote) }
```

ℹ Initialisation quand la variable est prête, c’est une constante !

 Référence [15] : https://antonioleiva.com/property-delegat/

Kotlin Avancé

Initialisation tardive de variable

```
lateinit var ballArray: Array<MagicCircle>
```

5 Référence [15] : https://antonioleiva.com/property-delegat/

Kotlin Avancé

Exemple d'initialisation tardive

```
var s: String? = null
lateinit var s1 :String

override fun onCreate(savedInstanceState: Bundle?) {
  super.onCreate(savedInstanceState)

  s = getString(R.string.app_name)
✖ s?.length

  s1 = getString(R.string.app_name)
✔ s1.length
}
```

ℹ Erreur `Uninitialized Property Access Exception`

 Référence [15] : https://antonioleiva.com/property-delegat/

Kotlin Avancé

Conversion de type

📎 *casting cast*

ℹ️ déclaration type optionnelle

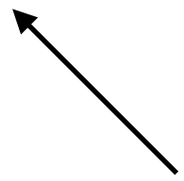
```kotlin
val mX = 20 // Conversion implicite en Int
val mZ = 10F // Conversion implicite en Float
val mY: Float = mX.toFloat() // Conversion explicite en Float
```

</> Conversion de nombres

Kotlin Avancé

Conversion de type

```kotlin
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_main)
    var tv = findViewById(R.id.mainTextView) as TextView
}
```



ℹ conversion en `TextView`

</> Conversion de type

Kotlin Avancé

~~*Tuple*~~

- Principe : retourner plusieurs valeurs

deprecated

Référence [] : kotlinlang.org: tuples-deprecated

Kotlin Avancé

Valeur multiple

- Principe : retourner plusieurs valeurs

ℹ *multiple return value*

Référence [] : jetbrains.com: traverse-a-map

Kotlin Avancé

Déclaration multiple

```
val (a, b) = returnTwoThings()
println(a)
println(b)
```

Kotlin Avancé

Déconstruire les valeurs

Deconstructing values

- Principe : obtenir une valeur particulière d'une fonction retournant usuellement une classe

ℹ ~ multiple valeur

Référence [] : kotlinlang.org: destructuring-declarations

Kotlin Avancé

Déconstruire les valeurs

```
class Country(val name: String, val capital: String) {
    operator fun component1(): String = name
    operator fun component2(): String = capital
}
```

ℹ data class : `component1()` `component2()`

Référence [] : medium.com/deconstructing-values-in-kotlin

Kotlin Avancé

Gestion des *exceptions*

Principe : Toute exception a

- un message
- une pile de traçage
- une possible cause

ℹ ~ multiple valeur

Référence [] : kotlinlang.org: exceptions.html

Kotlin Avancé

Gestion des *exceptions*

```
throw Exception("Hi There!")
```

```
try {
    // some code
} catch (e: SomeException) {
    // handler
} finally {
    // optional finally block
}
```

- lancer une exception

- attraper une exception

Référence [] : medium.com/deconstructing-values-in-kotlin

Kotlin Avancé

Déclaration de constante

```
companion object {
    const val DELTA = 8
    val delta = getDelta()
}
```

ℹ constantes de classe

Kotlin Avancé

Annotation

Principe : c’est un moyen d'attacher des métadonnées au code

```
annotation
```


Référence [] : kotlinlang.org: annotations.html

Kotlin Avancé

Annotation

```
@Target(AnnotationTarget.CLASS, AnnotationTarget.FUNCTION,
        AnnotationTarget.TYPE_PARAMETER, AnnotationTarget.VALUE_PARAMETER,
        AnnotationTarget.EXPRESSION)
@Retention(AnnotationRetention.SOURCE)
@MustBeDocumented
annotation class Fancy
```

ℹ type d'élément pouvant être annoté

ℹ fait partie de l'API publique, inclue dans la signature pour la documentation

ℹ stocké dans la classe compilé visible à l'exécution

Kotlin Avancé, Bibliothèque Standard

- Bibliothèque Standard *Kotlin*
- Collection *Kotlin*
- *Filtering*, *Mapping* et *Flatmapping*
- Évaluation *lazy*

Kotlin Avancé

Bibliothèque Standard Kotlin

Kotlin Standard Library

Principe : fournit les éléments essentiels pour le travail quotidien avec Kotlin

- Fonction Higher-order (`let`, `apply`, …)
- Fonction d'extension sur les collections
- Outils pour `String` et `Char`
- Extension pour les classes du JDK

Référence [] : kotlinlang.org/api/latest/jvm/stdlib/

Kotlin Avancé

Collection *Kotlin*

Principe : Outils pour la gestion de collection -groupe d'un nombre variable d'éléments (éventuellement zéro)

- *List :* collection ordonnée avec indice pour accéder à l'élt.
- *Set* : collection d'élt. unique
- *Map* : ensemble de paire clé-valeur

- Mécanisme : KSL offre classe interface fonction générale pour créer, peupler, gérer collection de n'importe quel type

Référence [] : kotlinlang.org: collections-overview.html

Kotlin Avancé

Collection *Kotlin*

```
val beaches = listOf(“sunglasses”, “towel”, “umbrella”)

beaches.forEach { beach - > println(beach) }

val party = beaches.map { it.reversed() }
```


Référence [] : kotlinlang.org: collections-overview.html

Kotlin Avancé

Filtering, *Mapping* et *Flatmapping*

```
beaches.mapNotNull { … }

beaches.mapIndexed { idx, m -> … }

beaches.filter { predicate }

beaches.filterNot { predicate }

beaches.take(2)

beaches.drop(100)
```


Référence [] : kotlinlang.org: collections-overview.html

Kotlin Avancé

Filtering, *Mapping* et *Flatmapping*

```
pontoon.average()

pontoon.sum()

beaches.sumOf { it.length }

pontoon.maxOrNull()

pontoon.minOrNull()

pontoon.count()

pontoon.count { it > 5 }
```


Référence [] : kotlinlang.org: collections-overview.html

Kotlin Avancé

Filtering, *Mapping* et *Flatmapping*

```
val ls2 = listOf("Monday", "Tuesday", "Wednesday",
“Thrusday","Friday","Saturday","Sunday")

println(ls2.flatMap { it.toList() })
```


Référence [] : kotlinlang.org: collections-overview.html

Kotlin Avancé

Évaluation *lazy*

```
val mStringArray: Array<String> by lazy
{ resources.getStringArray(R.array.motivation_quote) }
```

ℹ Initialisation quand c'est prêt :
LazyThreadSafetyMode.SYNCHRONIZED

Référence [] : kotlinlang.org: stdlib/kotlin/lazy.html

Conclusion

</> *Null safety*

📎 `toDouble()` / `as`

📎 Valeur multiple ~~*Tuple*~~

📎 val (a, b) = aAndb()

📎 `try {}catch(e:Exception) {}`

📎 `Companion object { const val }`

📎 @Annotation

📎 Bibliothèque Standard *Kotlin,* Collection *Kotlin*

📎 *Filtering*, *Mapping* et *Flatmapping*

📎 Évaluation *lazy*

THE EXPERT
AT ANYTHING
WAS ONCE
A BEGINNER.

_ Helen Hayes

Références

Kotlin for Android

- TRY Kotlin
- Kotlin Workshop on Github: Slides and Questions
- https://antonioleiva.com/free-kotlin-android-course/
- ChillCoding.com : Introduction à Kotlin
- ChillCoding.com : Configurer Kotlin dans un projet Android Studio

Library

- ChillCoding.com : Utiliser des bibliothèques graphiques Kotlin dans un projet Android

Fonction d’extension

- Odelia Technologies : Les fonctions d'extension de Kotlin

Kotlin in videos

- Jake Wharton and Kotlin (DEC 2015)
- Tue Dao & Christina Lee on The Road to Kotlintown (KotlinConf 2017)
- Introduction to Kotlin Google I/O '17