



KOTLIN PAR L'EXEMPLE



Alexandre Delattre

Expert Mobile & Backend

 @alexandre_del31

 alexandre@monkeypatch.io



Igor Laborie

Expert Java & Web

 @ilaborie

 igor@monkeypatch.io





Roadmap

#1

13h20

13h30

15h00

Présentation,
Installation

Exercice

...



Sinon utilisez le wifi

WI-FDE-INVITE /

7CndsqaiFDEUM.

- Installation de l'IDE

 [Android Studio 3.0](#),  [Download IntelliJ IDEA](#)

- Configuration éventuelle du plugin Kotlin

Tools | Kotlin | Configure Kotlin Plugin
Updates

Version basique:

1. `git clone http://github.com/MonkeyPatchIo/KotlinByExample-Lite`

Pour ceux qui ont: du réseau et qui veulent faire l'exercice serveur ou android ou web :

1. `git clone http://github.com/MonkeyPatchIo/KotlinByExample`
2. `git checkout {branch}` avec la branche dans `exo-mobile`,
`exo-server`, `exo-web`
3. `./gradlew clean assemble test` (les tests doivent être en erreurs)

WATER POURING PROBLEM

Tonneau magique

#6



@SunnyTech_MTP @alexandre_del31 @ilaborie #Kotlin #JAVA





8 / 8



0 / 6



1 / 4

Remplir

#8

Fill



1 / 4



4 / 4

Pour

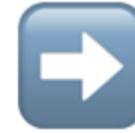


6 / 8

into



2 / 6



2 / 8



6 / 6

Empty



3 / 4



0 / 4

Démo

#11

EXERCICES

Basique

Implémenter le solveur
sans libs

Serveur avec SpringBoot
2

Implémentation du solveur
côté serveur



Définition et affichage de la
solution sous Android

Navigateur avec
KotlinJS

Définition et affichage de la
solution dans un navigateur

POUR DÉMARRER

```
fun main(args: Array<String>) {  
    println("Hello Sunny-Tech !")  
}
```



Utilisez Alt + Shift + (Cmd | Ctrl) + K pour convertir une classe Java en Kotlin

Ou copiez du code Java dans un fichier Kotlin

```
data class Glass(val capacity: Int,  
                val current: Int = 0) {  
  
    init {  
        require(capacity > 0) {  
            "Capacity: $capacity should be > 0"  
        }  
        require(current in 0..capacity) {  
            "Current: $current should be into [0, $capacity]"  
        }  
    }  
}  
  
typealias State = List<Glass>
```



En écrivant du Kotlin vous aurez plein de fun !

Le typealias nécessite Kotlin 1.1.



```
sealed class Move

data class Empty(val index: Int) : Move()

data class Fill(val index: Int) : Move()

data class Pour(val from: Int, val to: Int) : Move() {
    init {
        require(from != to)
    }
}
```



Avec les `sealed` et les `data class` on peut faire des *Abstract Data Class*
Le `sealed` nécessite Kotlin 1.1.

A vos



