





ET EN JAVA? JAVA 15 (SEALED), JAVA 14 (RECORD) ET JAVA 17 (EXHAUSTIVE)

**LINK TO THE PATH**

```
sealed interface Direction {  
    record North() implements Direction {}  
    record East() implements Direction {}  
    record South() implements Direction {}  
    record West() implements Direction {}  
}  
  
public static final String label(Direction d) {  
    return switch(d){  
        case Direction.North n -> "north";  
        case Direction.East e -> "east";  
        case Direction.South s -> "south";  
        case Direction.West w -> "west";  
    }  
}
```

Enfin des interfaces scellées + record permettent de décrire correctement un type « OU »

Finalelement JEP 406 <http://openjdk.java.net/jeps/8213076> Java 17 (Septembre 2021) apporte le « switch/case » pour l'exhaustivité d'un pattern matching, ainsi que c'est un nouveau « switch/case » expressif (parce que l'instruction switch java est quand même bien pourri)

Puis JEP 420 Java18 pour une amélioration des pattern de déconstruction au sein d'un switch/case expressif

Modèle similaire à Kotlin avec 10 ans de retard ou Scala avec 15 ans de retard ...

Mettre en prod un projet JAVA < 18 est une faute professionnelle 🤦🏻

MAGNUM2-QUALITÉ D'SI-THC MAS HAE SLÉ & DUE NINBURG

# LINK TO THE PATH

ET EN JAVA? JAVA 15 (SEALED), JAVA 14 (RECORD) ET JAVA 17 (EXHAUSTIVITÉ)

```
sealed interface Direction {  
    record North() implements Direction {}  
    record East() implements Direction {}  
    record South() implements Direction {}  
    record West() implements Direction {}  
}  
public static final String label(Direction d) {  
    return switch(d){  
        case Direction.North n -> "north";  
        case Direction.East e -> "east";  
        case Direction.South s -> "south";  
        case Direction.West w -> "west";  
    }  
}
```

Enfin des interfaces scellées + record permettent de décrire correctement un type « OU »

Finalement JEP 406 <http://openjdk.java.net/jeps/8213076> Java 17 (Septembre 2021) apporte le « switch/case » pour l'exhaustivité d'un pattern matching, ainsi que c'est un nouveau « switch/case » expressif (parce que l'instruction switch java est quand même bien pourri)

Puis JEP 420 Java18 pour une amélioration des pattern de déconstruction au sein d'un switch/case expressif

Modèle similaire à Kotlin avec 10 ans de retard ou Scala avec 15 ans de retard ...

Mettre en prod un projet JAVA < 18 est une faute professionnelle 🤔



# TAKE AWAY

AVEC UN LANGAGE STATIQUEMENT (BIEN) TYPÉ

On peut valider qu'on traite uniquement les valeurs valides

On peut valider qu'on traite toutes les valeurs valides

Dès la compilation

Dans les langages ML : OCaml, Haskell, F#, ...

Mais aussi les langages modernes qui s'en inspirent : TS, Scala, Kotlin, Swift, Rust, C++17, ...

