





```
Optional<Impacted> hitMonsterIf(Optional<Weapon> w, Optional<Target> t) {  
    if (w.isPresent() && t.isPresent()) {  
        return Optional.of(new Impacted());  
    } else {  
        return Optional.empty();  
    }  
}
```

Optional n'a pas évolué en sealed interface/record en Java17

La manipulation de Optional est fastidieuse et demande de la vigilance (pas de pattern matching)

**MODÉLISER UN ÉVÉNEMENT EN VALEUR**

TRAINERS VARNER DRINKER

MAGNÉT-QUALITÄT S-THOMAS & SÖHN

```
Optional<Impacted> hitMonsterFlatMap(Optional<Weapon> w, Optional<Target> t) {  
    return w.flatMap( sw -> t.map(st -> new Impacted()) );  
}
```

**flatMap (aka bind) et map facilitent la manipulation des valeurs optionnelles et**

**garantissent un bon traitement des cas**



# MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

## TRAITER LES VALEURS OPTIONNELLES EN JAVA

```
Optional<Impacted> hitMonsterFlatMap(Optional<Weapon> w, Optional<Target> t) {  
    return w.flatMap( sw -> t.map(st -> new Impacted()));  
}
```

flatMap (aka bind) et map facilitent la manipulation des valeurs optionnelles et garantissent un bon traitement des cas

# MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

## TRAITER LES VALEURS OPTIONNELLES EN TS

```
const hitMonsterIf = (w : Weapon | null, t : Target | null) : Impacted | null =>
  (w && t)
  ? new Impacted()
  : null;
```

`null` est un type littéral qui a une valeur : `null`

Si `strictNullChecks=true` dans `tsconfig.json` alors utiliser `null` est sound

Mais il y a beaucoup de confusion sémantique entre `undefined` (valeur non initialisée) & `null` (absence de valeur)

Les manipulation de type nullable ou leur expressivité est perfectible