MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR


OPTION

MIAGE M2 - QUALITÉ DU SI - THOMAS HAESSLÉ

```
type weapon
type target
type impacted = | Impacted

type 'a option =
  | Some of 'a
  | None

let arm_your_bow : weapon option  = None
let targeted_monster : target option = None
let hit_monster : weapon -> target -> impacted option  =
  fun w t -> None
```

```
class Weapon {}
class Target {}
class Impacted{}

interface Option<A>{}
class None<A> implements Option<A>{}
class Some<A> implements Option<A>{
  protected A value;
}

Option<Weapon> armYouBow = new None();
Option<Target> targetMonster = new None();
Option<Impacted> hitMonster(Weapon w, Target t) {
        return new None();
}
```

```
import java.util.Optional;
class Weapon {}
class Target {}
class Impacted{}

Optional<Weapon> armYouBow = Optional.empty();
Optional<Target> targetMonster = Optional.empty();
Optional<Impacted> hitMonster(Optional<Weapon> w,
Optional<Target> t) {
        return Optional.empty();
}
```

```
type weapon
type target
type impacted = | Impacted

let arm_your_bow : weapon option  = None
let targeted_monster : target option = None
let hit_monster : weapon -> target -> impacted option  =
  fun w t -> None
```

MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

OPTION

```java
import java.util.Optional;
class Weapon {}
class Target {}
class Impacted{}

Optional<Weapon> armYouBow = Optional.empty();
Optional<Target> targetMonster = Optional.empty();
Optional<Impacted> hitMonster(Optional<Weapon> w,
Optional<Target> t) {
        return Optional.empty();
}
```

```fsharp
type weapon
type target
type impacted = | Impacted

let arm_your_bow : weapon option  = None
let targeted_monster : target option = None
let hit_monster : weapon -> target -> impacted option  =
  fun w t -> None
```

MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

TRAITER LES VALEURS OPTIONNELLES

```
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

```
let hit_monster_pattern_match w t  =
  match w with
  | None -> None
  | Some w -> match t with
    | None -> None
    | Some t -> Some Impacted
```

Les patterns matching exhaustifs sont faciles à lire…
mais vite fastidieux à écrire