MODÉLISER UNE ERREUR POTENTIELLE

RESULT

MIAGE M2 - QUALITÉ DU SI - THOMAS HAESSLÉ

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

type ('a, 'b) result =
  | Ok of 'a
  | Error of 'b

let arm_your_bow : (weapon, exa) result  = Error (Failure "not carried")
let targeted_monster : (target, string) result = Error "too far"
```

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

let must_be_carried w = if w = "bow" then Ok w  else Error (Failure "not carried")
let (let*) = Result.bind

let hit_monster_let_star w t =
  let* used =  w in
  let* targeted = t in
  Ok (Impacted targeted)

let foo = hit_monster_let_star (must_be_carried "bow") (Ok "moblin");;
```

Depuis OCaml 4.08, la lib standard dispose de modules Option et Result satisfaisant

S'ils ne sont pas suffisant, regardez les lib Preface, Base ou Bastet

MODÉLISER UNE ERREUR POTENTIELLE

RESULT

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

let must_be_carried w = if w = "bow" then Ok w  else Error (Failure "not carried")
let (let*) = Result.bind

let hit_monster_let_star w t =
  let* used =  w in
  let* targeted = t in
  Ok (Impacted targeted)

let foo = hit_monster_let_star (must_be_carried "bow") (Ok "moblin");;
```

Depuis OCaml 4.08, la lib standard dispose de modules Option et Result satisfaisant

S’ils ne sont pas suffisant, regardez les lib Preface, Base ou Bastet

MODÉLISER UNE ERREUR POTENTIELLE

EN JAVA 17