

MODÉLISER UN ÉVÉNEMENT EN VALEUR

UNE VALEUR PROUR SE MÉR LA DÉSO LATION: NUL

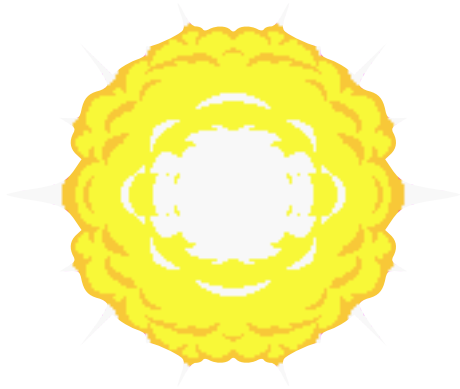
MAGNET 2-DUALS - THOUGHTS

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

public class Main {
    static Weapon armYouBow = null;
    static Target targetMonster = null;
    static Impacted hitMonster(Weapon w, Target t) {
        return null;
    }
    public static void main(String[] args) {
        hitMonster(armYouBow, targetMonster).toString();
    }
}
```

OCaml, Rust, Haskell : null n'existe pas

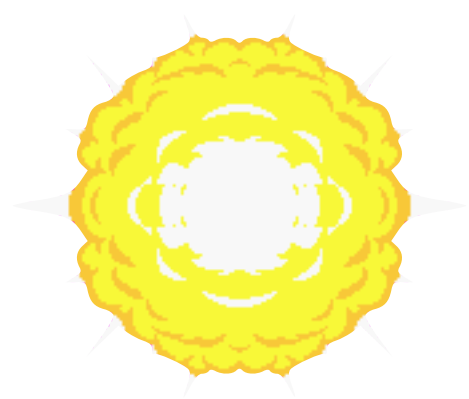
Kotlin, Swift, F# : n'autorisent null que si explicitement autorisé à la déclaration



```
Exception in thread "main" java.lang.NullPointerException
```


MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

UNE VALEUR POUR SEMER LA DÉSOLATION : NULL



Exception in thread "main" java.lang.NullPointerException

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

public class Main {
    static Weapon armYouBow = null;
    static Target targetMonster = null;
    static Impacted hitMonster(Weapon w, Target t) {
        return null;
    }
    public static void main(String[] args) {
        hitMonster(armYouBow, targetMonster).toString();
    }
}
```

OCaml, Rust, Haskell : null n'existe pas

Kotlin, Swift, F# : n'autorisent null que si explicitement autorisé à la déclaration

MODÉLISER UNE ABSENCE POTENTIELLE DE VALEUR

OPTION

```
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();
}
```

```
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
```