# EXCEPTIONS

# TRY / CATCH SONT LES RACINES DU MAL

# MIAGE M2 - QUALITÉ DU SI - THOMAS HAESSLÉ

```
let arm_your_bow : weapon = try raise (Failure "NoMoreArrow") with
    Failure s -> let _ = print_endline s in raise (Failure « unarmed")
```

```
class Weapon {
  Weapon(){
    throw new Exception("NoMoreArrow");
  }
}
public class Main {
 try {
    doSometingThatMayThrow();
    Weapon armYouBow = new Weapon();
 }catch(Exception e){

 }
}
```

## Dans une expression on peut passer la main ou traiter définitivement

Dans une instruction on peut s’arrêter là

On casse aussi la hiérarchie d’héritage en OOP

# EXCEPTIONS

## TRY / CATCH SONT LES RACINES DU MAL

```
class Weapon {
  Weapon(){
    throw new Exception("NoMoreArrow");
  }
}
public class Main {
 try {
    doSometingThatMayThrow();
    Weapon armYouBow = new Weapon();
 }catch(Exception e){

 }
}
```

```
let arm_your_bow : weapon = try raise (Failure "NoMoreArrow") with
    Failure s -> let _ = print_endline s in raise (Failure « unarmed")
```

Dans une instruction on peut s'arrêter là

On casse aussi la hiérarchie d'héritage en OOP

Dans une expression on peut passer la main ou traiter définitivement

# TAKE AWAY

MAUVAIS CHOIX POUR …

Modéliser l’absence de valeur

Modéliser une erreur fonctionnelle

Modéliser des erreurs asynchrones

OK SI …

Vous n’espérez pas que quelqu’un les « catch »

Vous voulez semer le chaos sur Hyrule

Vous savez ce que vous faites (contributeur VM)