EXCEPTIONS

RAISE / THROW ARE UGLY GOTOS

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

let arm_your_bow : weapon = raise (Failure "NoMoreArrow")
let targeted_monster : target = raise (Failure "TooMuchFog")
let hit_monster : weapon -> target -> impacted  =
    fun w t -> Impacted
```

```java
class Weapon {
  Weapon(){
    throw new RuntimeException("NoMoreArrow");
  }
}
class Target {
  Target(){
    throw new RuntimeException("TooMuchFog");
  }
}
class Impacted{}

public class Main {
    Weapon armYouBow = new Weapon();
    Target targetMonster = new Target();
    Impacted hitMonster(Weapon w, Target t) {
        return new Impacted();
    }
}
```

On remarquera le manque d'expressivité de Java

EXCEPTIONS

RAISE / THROW ARE UGLY GOTOS

```java
class Weapon {
  Weapon(){
    throw new RuntimeException("NoMoreArrow");
  }
}
class Target {
  Target(){
    throw new RuntimeException("TooMuchFog");
  }
}
class Impacted{}

public class Main {
    Weapon armYouBow = new Weapon();
    Target targetMonster = new Target();
    Impacted hitMonster(Weapon w, Target t) {
        return new Impacted();
    }
}
```

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

let arm_your_bow : weapon = raise (Failure "NoMoreArrow")
let targeted_monster : target = raise (Failure "TooMuchFog")
let hit_monster : weapon -> target -> impacted  =
    fun w t -> Impacted
```

On remarquera le manque d'expressivité de Java

EXCEPTIONS

RAISE / THROW ARE UGLY GOTOS LOST IN TIME OF ASYNC / FUTURE

```java
class Weapon {
    Weapon() { throw new RuntimeException("NoMoreArrow"); }
}
class Target {
    Target() { throw new RuntimeException("TooMuchFog");}
}
class Impacted {}


public class Main {
    static Weapon armYouBow = new Weapon();
    static Target targetMonster = new Target();

    static Impacted hitMonster(Weapon w, Target t) {
        return new Impacted();
    }
    public static FutureTask<Void> attack = new FutureTask<>(new Runnable() {
        @Override
        public void run() { hitMonster(armYouBow,targetMonster); }
    }, (Void) null);

    public static void main(String[] args)  {
        ExecutorService es = Executors.newSingleThreadExecutor();
        es.execute(attack);
        es.shutdown();
    }
}
```