EXCEPTIONS

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

```
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();
      }
}
```

EXCEPTIONS

TRY / CATCH SONT LES RACINES DU MAL

```
class Weapon {
  Weapon(){
    throw new RuntimeException("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")
```