Monads (Maybe?)

What is a Monad?

A monad is structure that represents computations defined as sequences of steps. A type with a monad structure defines what it means to chain operations, or nest functions of that type together. This allows the programmer to build pipelines that process data in steps, in which each action is decorated with additional processing rules provided by the monad. *wikipedia*

Kleisli triple

The usual formulation of a monad has 3 parts:

1. a type constructor
2. a unit function
3. a bind function

Kleisli triple (in Scala)

```
trait Monad[M[_]] {
  def unit[A](value: A): M[A]
  def bind[A, B](v: M[A], f: A => M[B]): M[B]
}
```

Petr Pudlak

Practical Application

Maybe monad, Logger monad, I/O monad

Practical Application (Maybe Monad)

Useful for semideterministic programming, that is: where something may be true, or it may not be.

monads in java

Maybe Monad: penicillin for NPE

Typical problem:

```
d:D = e.a.b.c.d
```

In order to get “d” from “e” we need to pull “a” then “b” then “c”. If this was Java this is the place where a future NPE might (will) happen, but Scala is just as bad since we will need to explicitly check when a, b, c, or d are Empty

Maybe Monad: penicillin for NPE

Solution to the previous problem:

```
e match {
  case A(x1) => x1 match {
    case B(x2) => x2 match { .. }
    case Empty => Empty
  case Empty => Empty
}
```

Maybe Monad: penicillin for NPE

```
sealed abstract class Maybe[+A] {
  override def unit[A](v: A): Maybe[A] = Maybe(v)
  override def bind[B](f: A => Maybe[B]): Maybe[B] =
    if (this.isEmpty)
      Empty
    else
      f(this.get)
}
case object Empty extends Maybe[Nothing] { def isEmpty = true }
case object Just[A](x: A) extends Maybe[A] { def isEmpty = false }
```

Maybe Monad: penicillin for NPE

awkward… escape to code

irc.freenode.net
#indyscala

Maybe Monad: penicillin for NPE

```
val d:D = e.a
  .bind(x => x.b)
  .bind(x => x.c)
  .bind(x => x.d)
```