CURSO DE PROGRAMACION SCALA
Sesión 9

Sergio Couto Catoira

Índice

- Implementación de Either
- Enumerations
- Laziness

Ejercicio

- Implementa las siguientes funciones dentro de Either
  - map
  - flatMap
  - orElse
  - map2

Ejercicio

- Emplea map2 para hacer lift sobre la función calcularCuota

Ejercicio

- Implementa las funciones sequence y traverse, deben devolver el primer error que encuentren si es que hay alguno
  - def sequence[E, A](a: List[Either[E,A]]): Either[E, List[A]]
  - def traverse[E,A,B](a: List[A])(f: A => Either[E,B]): Either[E, List[B]]
- Define sequence en base a traverse

Enumerations

- Objetos que extienden de enumeration
- Implementan clase Value => aporta id por defecto. Puede aportar id propio + String o sólo uno de ellos
- Pueden usarse con pattern matching
- Métodos
  - Values
  - ToString
  - Id
  - withName

Enumerations

- Ejercicio: Define una función que devuelva true si el día recibido es laborable y falso en caso contrario
- Ejercicio: Obtén todos los valores del enumeration Weekday e imprímelos ordenados
- Consigue un map de la forma (index, valor) Pista:Tendrás que usar métodos de la API de List
  - Map(0 -> Lunes, 5 -> Sabado, 1 -> Martes, 6 -> Domingo, 2 -> Miercoles, 3 -> Jueves, 4 -> Viernes)
- En vez de emplear el Value por defecto, podrias escribir una clase que extienda de Val. Hazlo para que cada día tenga un parámetro booleano de si es laborable. Reescribe la función laborable con este nuevo enumerado.

Laziness

- Evita la computación hasta que sea necesaria
- Por defecto, scala emplea evaluación estricta
- Permite evaluación lazy mediante el modificar lazy
  - lazy val x = 5+1
- Lo evalúa la primera vez que se llama. Nunca más
  - Evita errores con variables