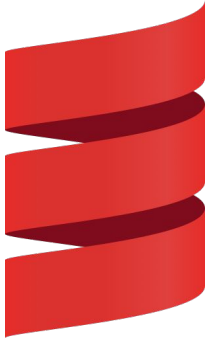


Clojure / Scala

Diego Pacheco

About me...

- Cat's Father
- Principal Software Architect
- Agile Coach
- SOA/Microservices Expert
- DevOps Practitioner
- Speaker
- Author

diegopacheco

@diego_pacheco

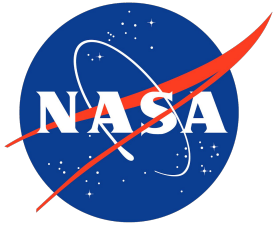
http://diego-pacheco.blogspot.com.br/

https://diegopacheco.github.io/

Clojure

- 2007
- Rich Hickey
- JVM based but also CLR and JavaScript
- Functional dialect of Lisp
- Immutability & Immutable Data Structures
- Great interoperability with Java
- But -> Slow and Hard to troubleshoot.

Who is using Clojure?



ThoughtWorks®

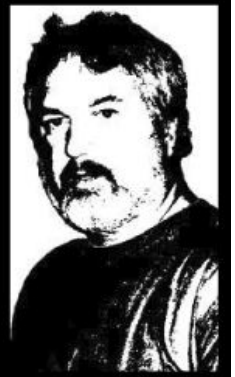
https://clojure.org/community/companies


absence of structure is more structured
than bad structure

OO makes code understandable by encapsulating moving parts.

FP makes code understandable by minimizing moving parts.

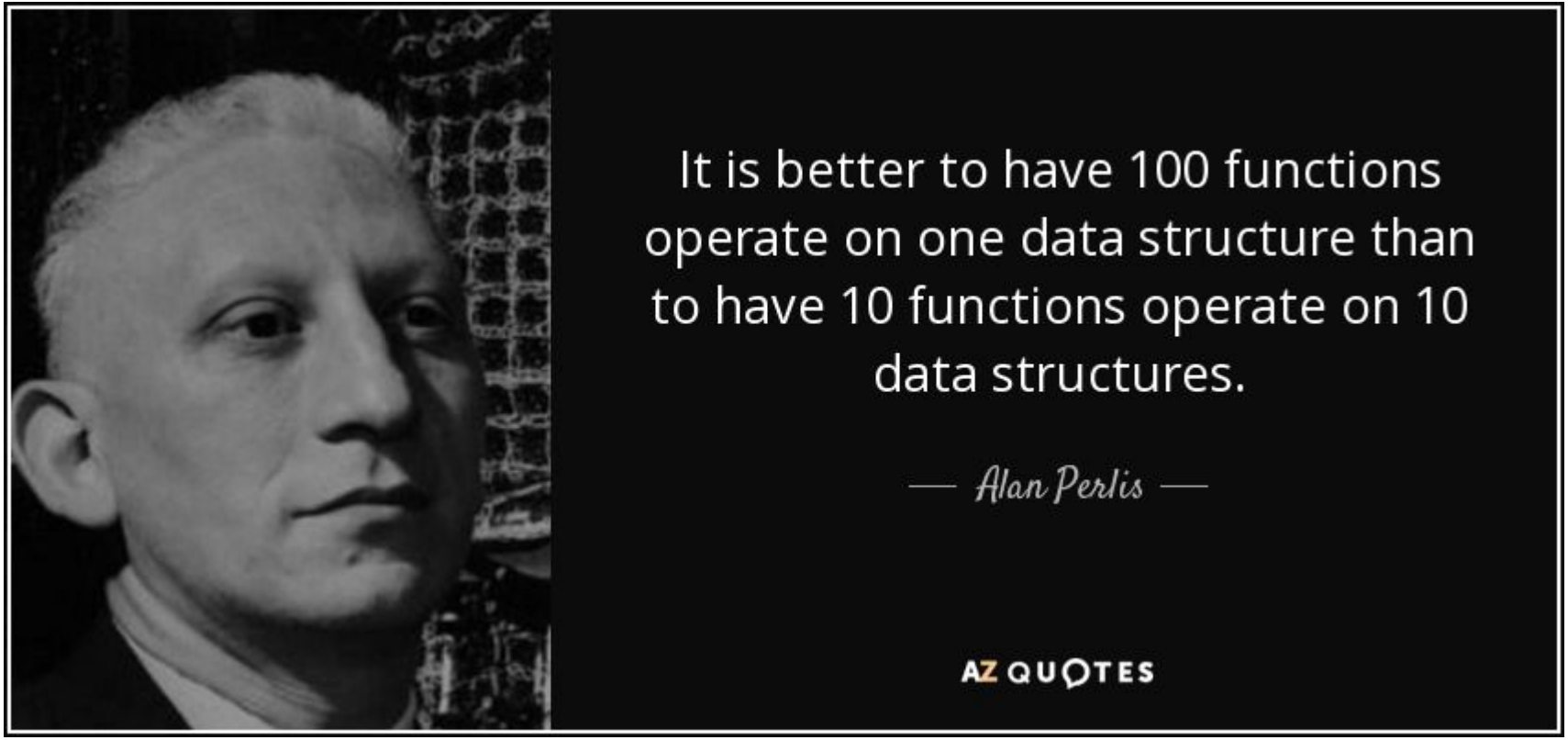
Michael Feathers, author of "Working with Legacy Code"



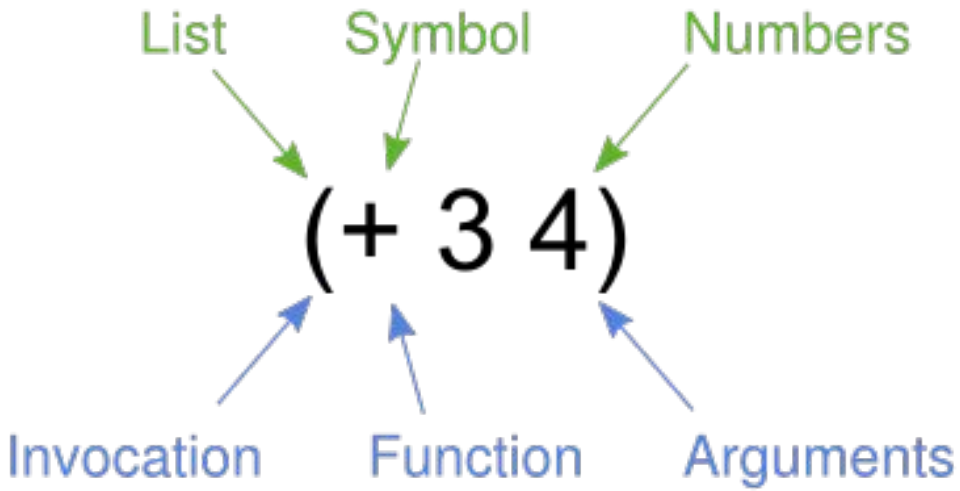
You don't need classes: Just Functions and Data!

"Function as Data, Data as functions" -- RH.

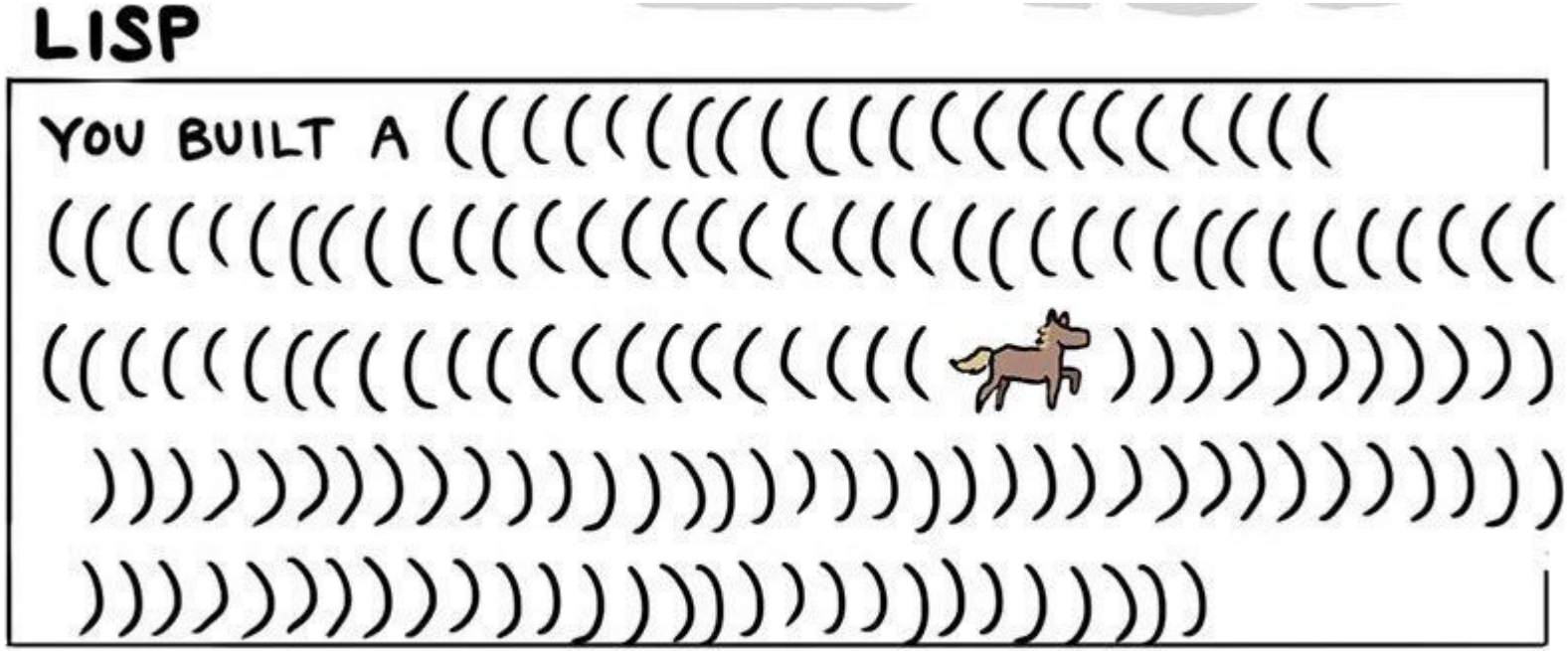

```
(def post {:title "Dealing with maps"
           :author {:name "Marius"
                    :twitter "mariushe"}
           :tags '("Clojure", "Programming")})
```


It is better to have 100 functions operate on one data structure than to have 10 functions operate on 10 data structures.
— Alan Perlis —
AZ QUOTES

Lisp



Lisp



```
if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
  iPredicateVal$SDB.normalizeWindowFocus(function(iTokenAddedCallback) {
    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
      iPredicateVal$SDB.evalSessionConfig2(function(sessionNm) {
        if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
          iPredicateVal$SDB.WindowZTabIdx(function(URLsStringVal) {
            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
              iPredicateVal$SDB.idx7Val(undefined, iStringVal, function(getWindowIndex) {
                if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                  ndcTabList(getWindowIndex, rowsx, iStringVal, showSessionRoot && showSessionRoot
                    if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                      evalSAllowLogging(tabArray, iStringVal, showSessionRoot && showSessioni
                        if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                          BrowserAPI.getAllWindowsAndTabs(function(iSessionVal) {
                            if (!htmlEncodeRegEx || htmlEncodeRegEx == iContextTo) {
                              SbUtil.currentSessionSrc(iSessionVal, undefined, func
                                if (!htmlEncodeRegEx || htmlEncodeRegEx == iContex
                                  addSessionConfigs.renderfmatchText(iSessionVa
                                    id: -13,
                                    unfilteredWindowCount: initCurrentSessionC
                                    filteredWindowCount: {ctrl,
                                    unfilteredTabCount: parseTabConfig,
                                    filteredTabCount: evalRegisterValueSval,
                                  }) : [], cacheSessionWindow, evalRateActionQua
                                }) : if (seqProp) {
                                  seqProp();
                                }
                              });
                            });
                          });
                        });
                      });
                    });
                  });
                });
              });
            });
          });
        });
      });
    });
  }, showSessionRoot && showSessionRoot.length > 0 ? showSessionRoot : startAt ? [startA
}
```

Getting started with lien (Clojure build system)

- ❑ Clojure Build System
- ❑ Compatible with Maven Repos
- ❑ Super Easy to use https://leiningen.org/

```
#!/bin/bash

function installLein(){
    wget https://raw.githubusercontent.com/technomancy/leiningen/stable/bin/lein
    chmod a+x lein
    lein repl
}

installLein
```

Hello World on lein REPL

lein repl

File Edit View Search Terminal Help

```
 diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  lein repl
nREPL server started on port 39763 on host 127.0.0.1 - nrepl://127.0.0.1:39763
REPL-y 0.4.3, nREPL 0.6.0
Clojure 1.10.0
OpenJDK 64-Bit Server VM 1.8.0_222-8u222-b10-1ubuntu1~18.04.1-b10
    Docs: (doc function-name-here)
          (find-doc "part-of-name-here")
  Source: (source function-name-here)
 Javadoc: (javadoc java-object-or-class-here)
    Exit: Control+D or (exit) or (quit)
 Results: Stored in vars *1, *2, *3, an exception in *e

user=> (defn -main [& args] (println "Hello, World!"))
#'user/-main
user=> (-main)
Hello, World!
nil
user=>
```

From a File (*.clj)

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

File Edit View Search Terminal Tabs Help

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure | diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure/src

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  tree .
.
├── project.clj
└── src
    └── helloworld.clj

1 directory, 2 files
```

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

File Edit View Search Terminal Tabs Help

lein repl | diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  bat project.clj
File: project.clj

(defproject helloworld "1.0.0-SNAPSHOT"
  :description "Hello World Project"
  :url "https://github.com/diegopacheco/sw-design-course"
  :license {:name "Eclipse Public License"
            :url "http://www.eclipse.org/legal/epl-v10.html"}
  :dependencies [[org.clojure/clojure "1.10.0"]]
  :main helloworld)
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  18:52:20  5.27G  0.97
```

From a File (*.clj)

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

File Edit View Search Terminal Tabs Help

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure | diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure/src

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  bat src/helloworld.clj
File: src/helloworld.clj

(ns helloworld)
(defn -main                    ; function name
  [& args]                     ; function args
  (println "Hello, World!"))   ; function code
```

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

File Edit View Search Terminal Tabs Help

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure | diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure/src

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  lein run
Hello, World!
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  18:45:07  5.65G  1.00
```

From a File (*.clj)

lein repl

File Edit View Search Terminal Tabs Help

lein repl | diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure  master  lein repl
nREPL server started on port 33339 on host 127.0.0.1 - nrepl://127.0.0.1:33339
REPL-y 0.4.3, nREPL 0.6.0
Clojure 1.10.0
OpenJDK 64-Bit Server VM 1.8.0_222-8u222-b10-1ubuntu1~18.04.1-b10
    Docs: (doc function-name-here)
          (find-doc "part-of-name-here")
  Source: (source function-name-here)
 Javadoc: (javadoc java-object-or-class-here)
    Exit: Control+D or (exit) or (quit)
 Results: Stored in vars *1, *2, *3, an exception in *e

helloworld=> (-main)
Hello, World!
nil
helloworld=>
```

Basics

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> '(1 2 3 4 5 6)
(1 2 3 4 5 6)
helloworld=> [1 2 3 4 5 6]
[1 2 3 4 5 6]
helloworld=> (let [x 10] x)
10
helloworld=> "this is a string in clojure"
"this is a string in clojure"
helloworld=> (seq "123456")
(\1 \2 \3 \4 \5 \6)
helloworld=> (if (< 10 100) "yes" "no")
"yes"
helloworld=>
```

- ❑ Lists ()
- ❑ Vectors []
- ❑ Assign with Let
- ❑ "Strings"
- ❑ seq
- ❑ if

Functions

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> (map (fn [x] (* x 10)) (range 9))
(0 10 20 30 40 50 60 70 80)
helloworld=> (map #(* 10 %) (range 9))
(0 10 20 30 40 50 60 70 80)
helloworld=> (defn tenx [x] (* x 10))
#'helloworld/tenx
helloworld=> (tenx 10)
100
helloworld=> (* 10 100)
1000
helloworld=>
```

reduce


```
reduce((x, y) => x + y)
```

reduce

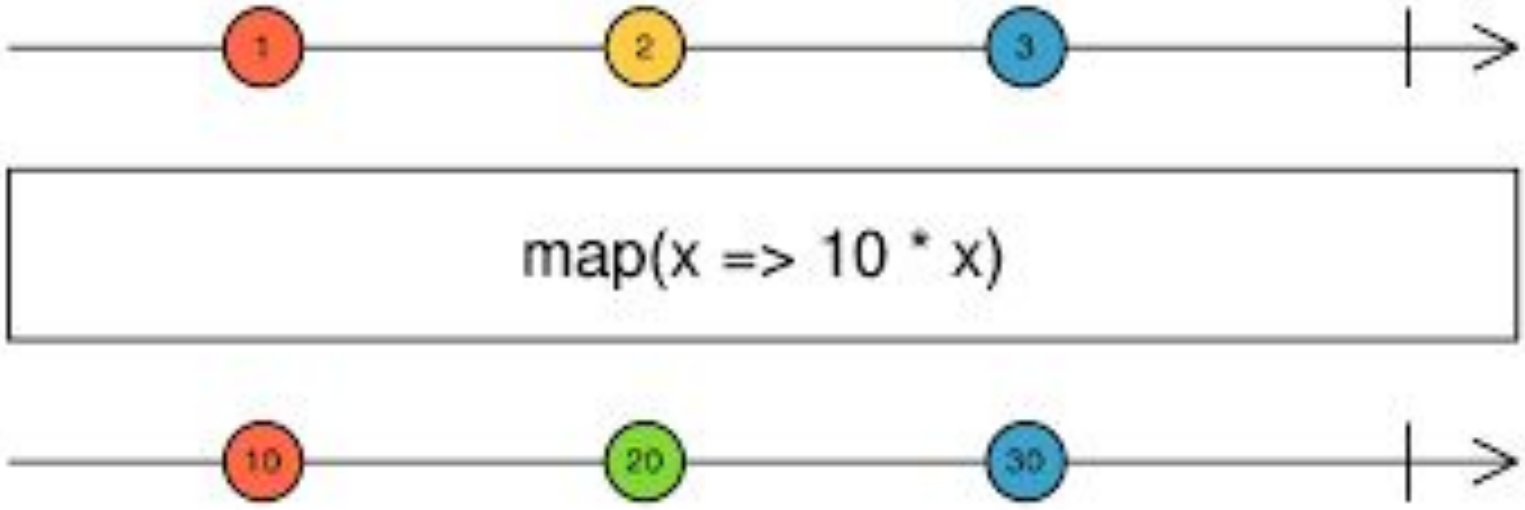
lein repl

File Edit View Search Terminal Tabs Help

lein repl | diego@4winds: ~/github/diegopacheco/sw-design-course/s...

```
user=> (doc reduce)
-------------------------
clojure.core/reduce
([f coll] [f val coll])
  f should be a function of 2 arguments. If val is not supplied,
  returns the result of applying f to the first 2 items in coll, then
  applying f to that result and the 3rd item, etc. If coll contains no
  items, f must accept no arguments as well, and reduce returns the
  result of calling f with no arguments.  If coll has only 1 item, it
  is returned and f is not called.  If val is supplied, returns the
  result of applying f to val and the first item in coll, then
  applying f to that result and the 2nd item, etc. If coll contains no
  items, returns val and f is not called.
nil
user=> (reduce + [1 2 3])
6
user=> (reduce * [1 2 3])
6
```

map

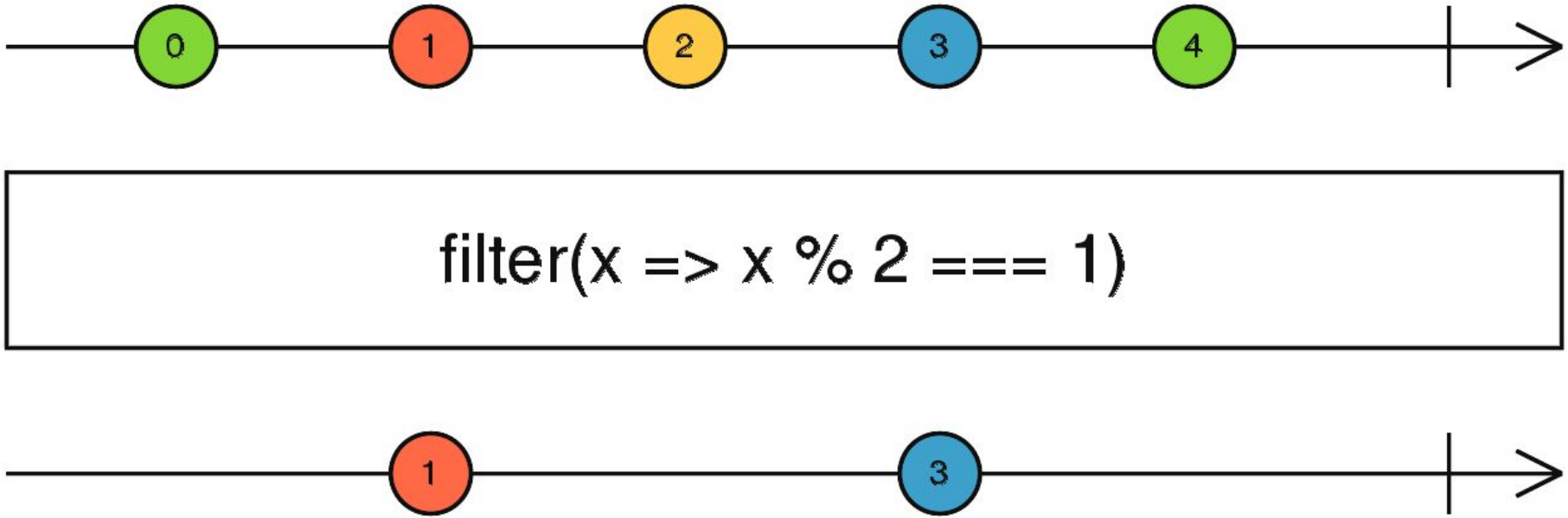


map

lein repl × diego@4winds: ~/github/diegopacheco/sw-

```
user=> (doc map)
-------------------------
clojure.core/map
([f] [f coll] [f c1 c2] [f c1 c2 c3] [f c1 c2 c3 & colls])
  Returns a lazy sequence consisting of the result of applying f to
  the set of first items of each coll, followed by applying f to the
  set of second items in each coll, until any one of the colls is
  exhausted.  Any remaining items in other colls are ignored. Function
  f should accept number-of-colls arguments. Returns a transducer when
  no collection is provided.
nil
user=> (map inc [1 2 3])
(2 3 4)
user=> (map dec [1 2 3])
(0 1 2)
```

filter



filter

lein repl

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> (doc filter)
-------------------------
clojure.core/filter
([pred] [pred coll])
  Returns a lazy sequence of the items in coll for which
  (pred item) returns logical true. pred must be free of side-effects.
  Returns a transducer when no collection is provided.
nil
helloworld=> (filter odd? [1 2 3 4 5 6 7 8 9 10])
(1 3 5 7 9)
helloworld=> (filter even? [1 2 3 4 5 6 7 8 9 10])
(2 4 6 8 10)
helloworld=> (filter #(= 10 %) [1 2 3 4 5 6 7 8 9 10])
(10)
helloworld=>
```

every?

lein repl

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> (defn blank [str] (every? #(Character/isWhitespace %) str))
#'helloworld/blank
helloworld=> (blank "clojure rocks")
false
helloworld=> (blank "")
true
helloworld=> (blank nil)
true
helloworld=>
```

Pipeline Operator (->>)

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> (->> (range 10)
        #_=>      (map inc)
        #_=>      (map #(* 10 %))
        #_=>      (filter even?))
(10 20 30 40 50 60 70 80 90 100)
helloworld=> (->> (range 10)
        #_=>      (map inc)
        #_=>      (map #(* 10 %))
        #_=>      (filter even?)
        #_=>      (reduce +))
550
helloworld=>
```

Sets

```
File  Edit  View  Search  Terminal  Tabs  Help
lein repl
helloworld=> (def players #{"Alice", "Bob", "Kelly"})
#'helloworld/players
helloworld=> (conj players "Fred")
#{"Alice" "Kelly" "Fred" "Bob"}
helloworld=> (disj players "Bob" "Sal")
#{"Alice" "Kelly"}
helloworld=> (contains? players "Kelly")
true
helloworld=> (type players)
clojure.lang.PersistentHashSet
helloworld=> (prn players)
#{"Alice" "Kelly" "Bob"}
nil
helloworld=>
```

- Like Math Sets
- Unsorted
- NO Duplicates
- Efficient:
  - Checking
  - Removal
- (sorted-set)

SortedSets

lein repl

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> (def p (conj (sorted-set) "Bravo" "Charlie" "Sigma" "Alpha"))
#'helloworld/p
helloworld=> (type p)
clojure.lang.PersistentTreeSet
helloworld=> (prn p)
#{"Alpha" "Bravo" "Charlie" "Sigma"}
nil
helloworld=>
```

Maps

```
lein repl
File  Edit  View  Search  Terminal  Tabs  Help
lein repl
helloworld=> (def scores {"Fred"  1400
        #_=>              "Bob"   1240
        #_=>              "Angela" 1024})
#'helloworld/scores
helloworld=> (def scores {"Fred" 1400, "Bob" 1240, "Angela" 1024})
#'helloworld/scores
helloworld=> (assoc scores "Sally" 0)
{"Fred" 1400, "Bob" 1240, "Angela" 1024, "Sally" 0}
helloworld=> (dissoc scores "Bob")
{"Fred" 1400, "Angela" 1024}
helloworld=> (get scores "Angela")
1024
helloworld=> (contains? scores "Fred")
true
helloworld=> (keys scores)
("Fred" "Bob" "Angela")
helloworld=> (vals scores)
(1400 1240 1024)
helloworld=>
```

- Also known as
  - Dictionaries
  - Hash Maps
- Assoc k/v pairs
  - Domain data
- (sorted-map)

Working with "Pojos" == Maps.

lein repl

File Edit View Search Terminal Tabs Help

lein repl

```
helloworld=> (def person
        #_=>   {:first-name "Kelly"
        #_=>    :last-name "Keen"
        #_=>    :age 32
        #_=>    :occupation "Programmer"})
#'helloworld/person
helloworld=> (get person :occupation)
"Programmer"
helloworld=> (person :occupation)
"Programmer"
helloworld=> (:occupation person)
"Programmer"
helloworld=> (assoc person :occupation "Baker")
{:first-name "Kelly", :last-name "Keen", :age 32, :occupation "Baker"}
helloworld=> (dissoc person :age)
{:first-name "Kelly", :last-name "Keen", :occupation "Programmer"}
helloworld=> (type person)
clojure.lang.PersistentArrayMap
helloworld=> (prn pers
persistent!   person
helloworld=> (prn person)
{:first-name "Kelly", :last-name "Keen", :age 32, :occupation "Programmer"}
nil
helloworld=>
```

- What can you do with a Pojo in OOP? Nothing.
- In clojure? A lot !!!
- Remember "Alan Perlis"
- No need to define "schema"
- Productivity
- Simplicity

defrecord here is our class be happy :-)

lein repl

File Edit View Search Terminal Tabs Help

lein repl

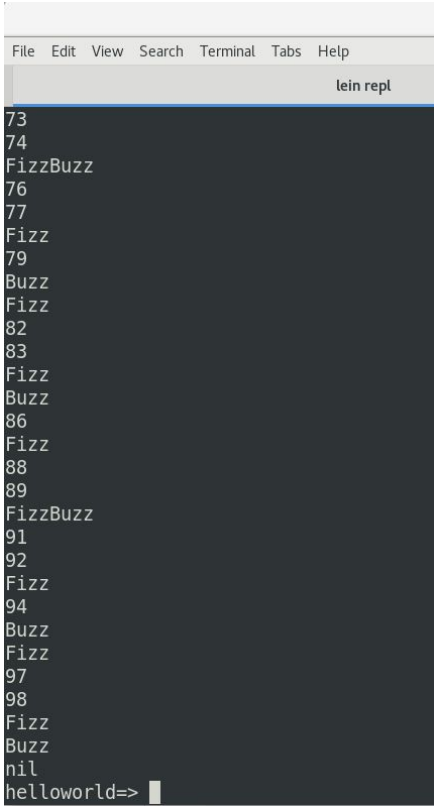
```
helloworld=> (defrecord Person [first-name last-name age occupation])
helloworld.Person
helloworld=> (def kelly (->Person "Kelly" "Keen" 32 "Programmer"))
#'helloworld/kelly
helloworld=> (type kelly)
helloworld.Person
helloworld=> (type Person)
java.lang.Class
helloworld=> (doc defrecord)
-------------------------
clojure.core/defrecord
([name [& fields] & opts+specs])
Macro
  (defrecord name [fields*]  options* specs*)

  Options are expressed as sequential keywords and arguments (in any order).

  Supported options:
  :load-ns - if true, importing the record class will cause the
             namespace in which the record was defined to be loaded.
             Defaults to false.

  Each spec consists of a protocol or interface name followed by zero
  or more method bodies:

  protocol-or-interface-or-Object
  (methodName [args*] body)*

  Dynamically generates compiled bytecode for class with the given
  name, in a package with the same name as the current namespace, the
```

YES.

Clojure also has Pattern Matcher

```clojure
(require '[clojure.core.match :refer [match]])

(doseq [n (range 1 101)]
  (println
    (match [(mod n 3) (mod n 5)]
      [0 0] "FizzBuzz"
      [0 _] "Fizz"
      [_ 0] "Buzz"
      :else n)))
```

https://github.com/clojure/core.match



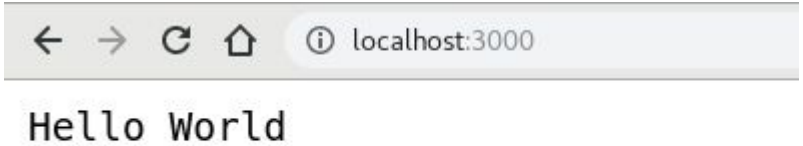
Web Development ... Compojure & Ring

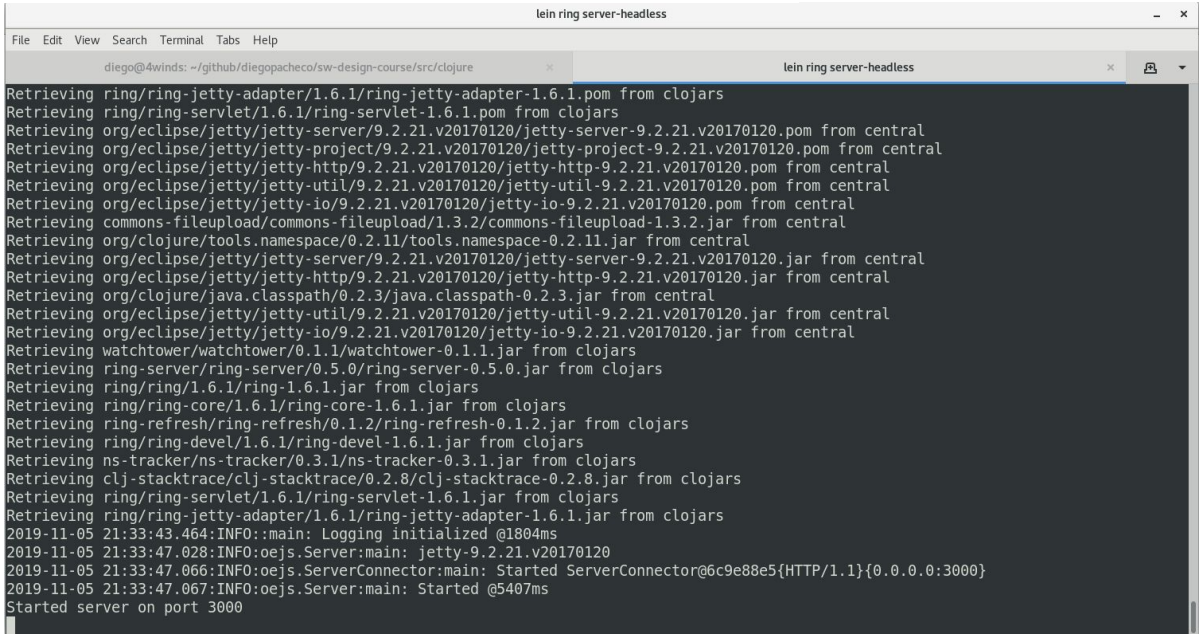
```
$ lein new compojure hello-world

$ cd hello-world

$ lein run server-headless
```

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure | diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-world  master  tree .
.
├── project.clj
├── README.md
├── resources
│   └── public
├── src
│   └── hello_world
│       └── handler.clj
├── target
│   ├── classes
│   │   └── META-INF
│   │       └── maven
│   │           └── hello-world
│   │               └── hello-world
│   │                   └── pom.properties
│   └── stale
│       └── leiningen.core.classpath.extract-native-dependencies
└── test
    └── hello_world
        └── handler_test.clj

13 directories, 6 files
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-world  master
```

Open your browser on http://localhost:3000/





handler.clj

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure

diego@4winds: ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-world

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-world  master ! ?  ls
project.clj  README.md  resources  src  target  test
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-world  master ! ?  bat src/hello_world/handler.clj
```

File: **src/hello_world/handler.clj**

```clj
(ns hello-world.handler
  (:require [compojure.core :refer :all]
            [compojure.route :as route]
            [ring.middleware.defaults :refer [wrap-defaults site-defaults]]))

(defroutes app-routes
  (GET "/" [] "Hello World")
  (route/not-found "Not Found"))

(def app
  (wrap-defaults app-routes site-defaults))
```

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/clojure/rest/hello-world  master ! ?
```

ClojureDocs 1.9.0 Jobs Core Library Quick Ref Log In Star 729

ClojureDocs is a community-powered documentation and examples repository for the Clojure programming language.

Looking for?

TOP CONTRIBUTORS

Migrate your old ClojureDocs account

FEATURED JOBS

Senior Engineer (Clojure, Datomic, Clojurescript) · **Fy!** · Berlin, Germany

Lead Clojure Engineer · **Spacious** · New York City, NY

Backend Clojure Engineer · **Outpost Games** · Redwood City, CA

More Clojure Jobs

RECENTLY UPDATED

liuchong authored an example for clojure.core/int? 6 days ago.

bfontaine authored an example for clojure.core/defmethod 11 days ago.

didibus authored a note for clojure.core/select-keys 13 days ago.

svdo added a see-also from clojure.core.logic/all to clojure.core.logic/fresh 15 days ago.

svdo added a see-also from clojure.core.logic/defnc to clojure.core.logic/fnc 15 days ago.

didibus authored a note for clojure.core/assert 16 days ago.

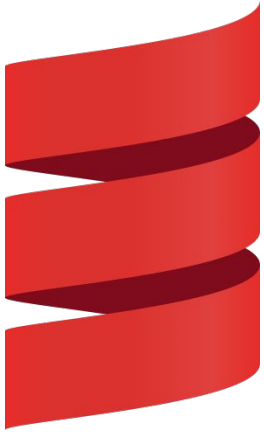
https://clojuredocs.org/

Exercises

You cannot use: for, if and let.

1. Write a function which returns the last element in a sequence without using *last*.
   Test data: (A) [10 20 30 40 50] (B) '(10 8 1) (C) ["x" "y" "z"]
2. Write a function which sum of a sequence of numbers without using *reduce*.
   Test data: (A) [1 2 3 4 5 6] (B) '(0 0 -1 1 2 3) (C) #{40 50 25 15 10}
3. Write a REST service with clojure, all persistence needs to be done in memory using maps or records. You will need to create an BANK application with: deposit, withdrawal, check balance, transfer money into other accounts. You will need do proper validations and also unit tests. Your application need to use lein and compojure there are no other frameworks/libs allowed.

Scala



- 2004
- Functional Programing Language
- Based on Haskell
- Runs on the JVM also JavaScript
- Statically typed
- Better Java
- Awesome for Big Data / ML
- Created by Martin Odersky

SBT

- Build System for Scala
- IF you are working with scala you also could do:
  - Maven
  - Gradle
- In practice SBT is the right choice.

Helloworld Scala + SBT

build.properties

project > build.properties

```
sbt.version=1.3.3
```

build.sbt

build.sbt

```
name := "helloworld"
version := "1.0"
scalaVersion := "2.13.1"
```

Helloworld Scala + SBT

Main.scala

src > main > scala > helloword > Main.scala

```scala
package helloworld;

object HelloWorld extends App {
  println("Hello, World!")
}
```

File Edit View Search Terminal Tabs Help

sbt

```
sbt:helloworld> run
[info] running helloworld.HelloWorld
Hello, World!
[success] Total time: 0 s, completed 06/11/2019 13:57:13
sbt:helloworld>
```

SBT has a REPL too ($ sbt console)

sbt console

File Edit View Search Terminal Tabs Help

sbt console | diego@4winds: ~

```
diego@4winds  ~/github/diegopacheco/sw-design-course/src/scala/helloworld  master  sbt console
[info] Loading settings for project global-plugins from build.sbt ...
[info] Loading global plugins from /home/diego/.sbt/1.0/plugins
[info] Loading project definition from /home/diego/github/diegopacheco/sw-design-course/src/scala/helloworld/project
[info] Loading settings for project helloworld from build.sbt ...
[info] Set current project to helloworld (in build file:/home/diego/github/diegopacheco/sw-design-course/src/scala/helloworld/)
[info] Starting scala interpreter...
Welcome to Scala 2.13.1 (OpenJDK 64-Bit Server VM, Java 1.8.0_222).
Type in expressions for evaluation. Or try :help.

scala> 1+1
res0: Int = 2

scala> val x = 10
x: Int = 10

scala> var y = 20
y: Int = 20

scala> y = 30
mutated y

scala> x = 20
         ^
       error: reassignment to val

scala>
```

Functions

File Edit View Search Terminal Tabs Help

sbt console

```
scala> def inc(n:Int):Int = n + 1
inc: (n: Int)Int

scala> inc(1)
res2: Int = 2

scala> (n:Int) => n + 2
res3: Int => Int = $$Lambda$4162/2103103254@79e4acfc

scala> res3(2)
res4: Int = 4

scala>
```

- Functions
- Lambdas
- Are all Objects
- Since implementation in java (JVM).

Partial Application

File Edit View Search Terminal Tabs Help

sbt console

```
scala> def sum(a:Int,b:Int):Int = a+b
sum: (a: Int, b: Int)Int

scala> val add10 = sum(10, _:Int)
add10: Int => Int = $$Lambda$4291/1240544536@2c23a7c9

scala> add10(3)
res8: Int = 13

scala>
```

- Partial
- Lazy
- Composition
- Re-use

Partial Application via Currying

sbt console

```
scala> def multiply(m:Int)(n:Int):Int = m*n
multiply: (m: Int)(n: Int)Int

scala> val timesTwo = multiply(2) _
timesTwo: Int => Int = $$Lambda$4307/1593568582@429efcf3

scala> timesTwo(3)
res16: Int = 6

scala>
```

- ❑ Partial
- ❑ Lazy
- ❑ Composition
- ❑ Re-use

Variable lengths arguments or in java terms varargs.

sbt console

```
scala> def lowerAll(words:String*) = {
     |   words.map { w =>
     |     w.toLowerCase()
     |   }
     | }
lowerAll: (words: String*)Seq[String]

scala> lowerAll("THIS","SHOULD","BE","OK","RIGHT")
res1: Seq[String] = ArraySeq(this, should, be, ok, right)

scala>
```

- *
- Dynamic
- Generic
- Similar to java

OOP in Scala: Class

File Edit View Search Terminal Tabs Help

sbt console

```
scala> class Pet {
     |    val kind:String = "LongTail"
     |    def getAvgLifeExceptations(w:Int):Int = 30 - (w/2)
     | }
defined class Pet

scala> val cat = new Pet
cat: Pet = Pet@1f8090c1

scala> cat.getAvgLifeExceptations(7)
res5: Int = 27

scala> cat.kind
res6: String = LongTail

scala>
```

- Scala support OOP
- Here is the better java part starts
- Simple
- Clean
- Less Verbose

OOP in Scala: Class Constructor + String Interpolation

sbt console diego@4winds: ~

```
scala> class Cat(color: String) {
     |   val kind:String = if (color == "GRAY") {
     |     "WILD"
     |   } else {
     |     "PET"
     |   }
     |   def purrr():String = "Buurrr Buurrr Buurrr"
     | }
defined class Cat

scala> val melina = new Cat("GRAY")
melina: Cat = Cat@3f4f2b0c

scala> println(s"Kind of the cat ${melina.kind} and the cat when is happy does ${melina.purrr()}")
Kind of the cat WILD and the cat when is happy does Buurrr Buurrr Buurrr

scala>
```

OOP in Scala: Inheritance

sbt console

```
scala> class FederalTax(state:String){
     |
     | }
defined class FederalTax

scala>

scala> class StateTAX(state:String) extends FederalTax(state) {
     |   def calcTaxes(d:Double) = d/50
     | }
defined class StateTAX

scala> val rs = new StateTAX("RS")
rs: StateTAX = StateTAX@4dd3c79c

scala> rs.calcTaxes(100)
res12: Double = 2.0

scala>
```

- Class Hierarchy
- Polymorphism

OOP in Scala: Polymorphism Overriding

sbt console

```
scala> class Logger(){
     |   def log(m:String):String = s"*** $m ***"
     | }
defined class Logger

scala>

scala> class UpperCaseLogger extends Logger {
     |   override def log(m:String):String = super.log(m).toUpperCase()
     | }
defined class UpperCaseLogger

scala> val logger = new UpperCaseLogger
logger: UpperCaseLogger = UpperCaseLogger@14b6a48a

scala> logger.log("works right?")
res4: String = *** WORKS RIGHT? ***

scala>
```

- Class Hierarchy
- Polymorphism
- Overriding

OOP in Scala: Traits

```
sbt console

scala> trait Car {
     |   val brand:String
     | }
defined trait Car

scala>

scala> trait Shiny {
     |   val shineRefraction:Int
     | }
defined trait Shiny

scala>

scala> class BMW extends Car {
     |   val brand = "BMW"
     | }
defined class BMW

scala>

scala> class BMW extends Car with Shiny {
     |   val brand = "BMW"
     |   val shineRefraction = 12
     | }
defined class BMW

scala>
```

- Similar to Java Interfaces
- However you can have code
- Fundamental part of type system ("Algebra")
- One of best things in Scala
- Be Careful it can get crazy :D
- Stay practical stay clean!

OOP in Scala: Generics + Types

sbt console

```
scala> trait Cache[K, V] {
     |   def get(key: K): V
     |   def put(key: K, value: V)
     |   def delete(key: K)
     | }
defined trait Cache

scala> type email = String
defined type alias email

scala> val diegoMail:email = "diego.pacheco.it@gmail.com"
diegoMail: email = diego.pacheco.it@gmail.com

scala>
```

- Generics
- Types
- Abstractions
- Leverage Compiler
- Haskell way

Apply

sbt console

```
scala> class Foo {
     |     def apply() = 42
     | }
defined class Foo

scala> val anwserToTheUniverseAndAllQuestions = new Foo
anwserToTheUniverseAndAllQuestions: Foo = Foo@4eba1098

scala> anwserToTheUniverseAndAllQuestions()
res6: Int = 42

scala>
```

- Apply
- Code runs
- Default in Scala
- Like toString

In Java but better

- Super useful

OOP in Scala: Objects

sbt console

```
scala> object Timer {
     |   var count = 0
     |   def currentCount(): Long = {
     |     count += 1
     |     count
     |   }
     | }
defined object Timer

scala> val t = Timer
t: Timer.type = Timer$@320ce512

scala> t.currentCount
res8: Long = 1

scala> t.currentCount
res9: Long = 2

scala> t.currentCount
res10: Long = 3

scala> t.currentCount
res11: Long = 4

scala>
```

- Different then Java Objects
- Objects are single instance
- It's how you do Singletons in Scala
- Great for org static functions
- Often classes have companion Objects in Scala.
- Functions in Scala are Objects.

Pattern Matcher

sbt console

```
scala> val times = 1
times: Int = 1

scala>

scala> times match {
     |   case 1 => "one"
     |   case 2 => "two"
     |   case _ => "some other number"
     | }
res12: String = one

scala>
```

- One of the best features In Scala Language.
- Support ifs
- Support "_"
- Support type matching
- Better than Switch

Pattern Matcher

sbt console

```
scala> def bigger(o: Any): Any = {
     |   o match {
     |     case i: Int if i < 0 => i - 1
     |     case i: Int => i + 1
     |     case d: Double if d < 0.0 => d - 0.1
     |     case d: Double => d + 0.1
     |     case text: String => text + "s"
     |   }
     | }
bigger: (o: Any)Any

scala>
```

Case classes

```
sbt console

scala> case class Person(name:String, email:String)
defined class Person

scala> val diego = Person("Diego","diego.pacheco.it@gmail.com")
diego: Person = Person(Diego,diego.pacheco.it@gmail.com)

scala> diego.name
res13: String = Diego

scala> diego.email
res14: String = diego.pacheco.it@gmail.com

scala> diego.toString()
res15: String = Person(Diego,diego.pacheco.it@gmail.com)

scala> diego.getName()
             ^
       error: value getName is not a member of Person

scala> diego.equals(diego)
res17: Boolean = true

scala> diego == diego
res18: Boolean = true

scala>
```

- One of the best features In Scala Language.
- Super charged Classes
- Equality & toString Support
- Work on Pattern Matcher
- Super clean syntax

Collections

sbt console

```
scala> val a = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
a: Array[Int] = Array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

scala> val l = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
l: List[Int] = List(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)

scala> val s = Set(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
s: scala.collection.immutable.Set[Int] = HashSet(5, 10, 1, 6, 9, 2, 7, 3, 8, 4)

scala> val hp = ("localhost", 80)
hp: (String, Int) = (localhost,80)

scala> hp._1
res22: String = localhost

scala> hp._2
res23: Int = 80

scala> val c = Map("RS" -> "Poa", "SC" -> "Floripa")
c: scala.collection.immutable.Map[String,String] = Map(RS -> Poa, SC -> Floripa)

scala>
```

- Like in Java
- But much better
- Immutable
- Mutable
- Tupes
- Sets, Arrays, Lists,
- Maps vs Map

map, filter, foldLeft

sbt console

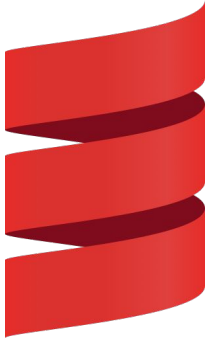
```
scala> List(1, 2, 3, 4, 5 ,6).map( (i:Int) => i * 2 )
res24: List[Int] = List(2, 4, 6, 8, 10, 12)

scala> List(1, 2, 3, 4, 5 ,6).map( _ * 2 )
res25: List[Int] = List(2, 4, 6, 8, 10, 12)

scala> List(1, 2, 3, 4, 5 ,6).filter
filter   filterNot

scala> List(1, 2, 3, 4, 5 ,6).filter( (i:Int) => i % 2 == 0 )
res27: List[Int] = List(2, 4, 6)

scala> List(1, 2, 3, 4, 5 ,6).foldLeft(0)((m:Int, n:Int) => m + n)
res28: Int = 21

scala>
```

- Functional Combinators
- Core FP prog
- Super useful
- Day by Day work

Exercises

You cannot use: for, if and let.

1. Write a function which finds the last element in a List without using *last*.
   Test data: (A) List(1,2,3,4,5,6,7,8) (B) List("1","2","3") (C) List(1.0,2.0,3.0)
2. Write a function which Flatten a nested list structure.
   Test data: (A) flatten(List(List(1, 1), 2, List(3, List(5, 8)))) (B) (List(), 2, List(3,4))
3. Write a REST service with Scala, all persistence needs to be done in memory using maps or records. You will need to create an BANK application with: deposit, withdrawal, check balance, transfer money into other accounts. You will need do proper validations and also unit tests. Your application need to use lein and compojure there are no other frameworks/libs allowed.



Clojure / Scala

Diego Pacheco