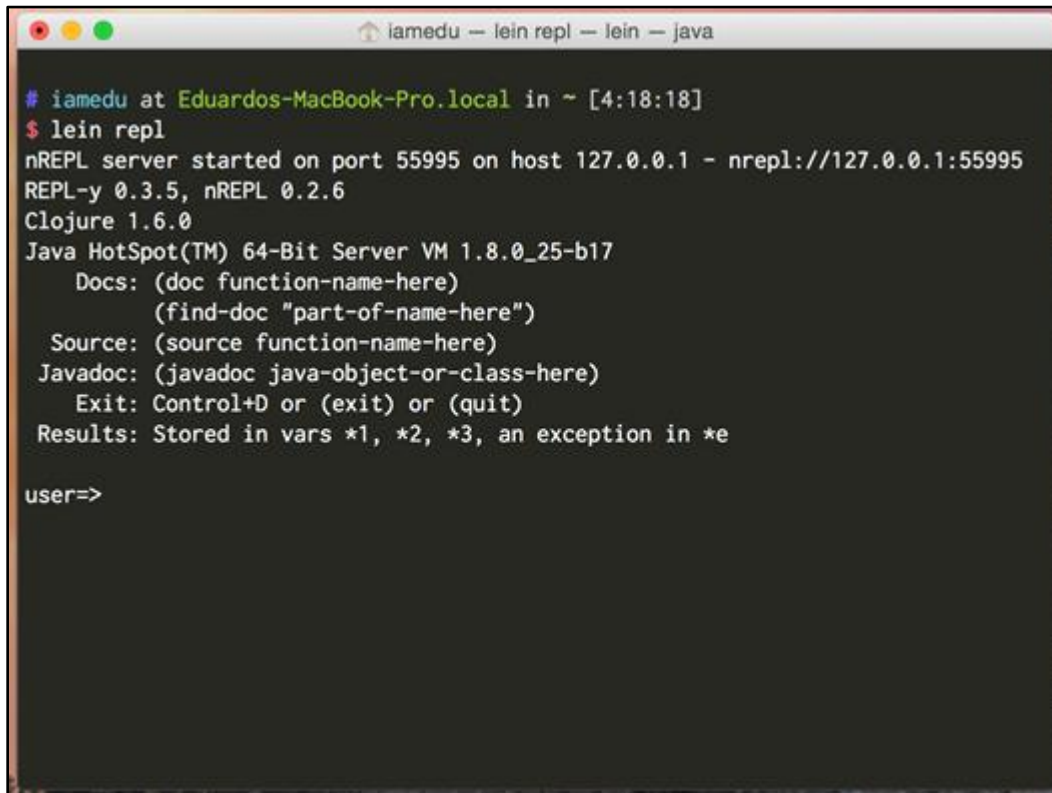


Chapter 1: Getting Started with Clojure



```
iamedu -- lein repl -- lein -- java
# iamedu at Eduardos-MacBook-Pro.local in ~ [4:18:18]
$ lein repl
nREPL server started on port 55995 on host 127.0.0.1 - nrepl://127.0.0.1:55995
REPL-y 0.3.5, nREPL 0.2.6
Clojure 1.6.0
Java HotSpot(TM) 64-Bit Server VM 1.8.0_25-b17
  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=>
```

```
iamedu — lein repl :connect localhost:55995 — lein — java
user=> "Hello world"
"Hello world"
user=> (println "Hello world")
Hello world
nil
user=>
```

```
getting-started — iamedu@Eduardos-MacBook-Pro: — ..tting-started — zsh
# iamedu at Eduardos-MacBook-Pro.local in ~/Development/getting-started [20:47:23]
$ tree .
.
├── LICENSE
├── README.md
├── doc
│   └── intro.md
├── project.clj
├── resources
├── src
│   ├── getting_started
│   └── core.clj
└── test
    ├── getting_started
    └── core_test.clj

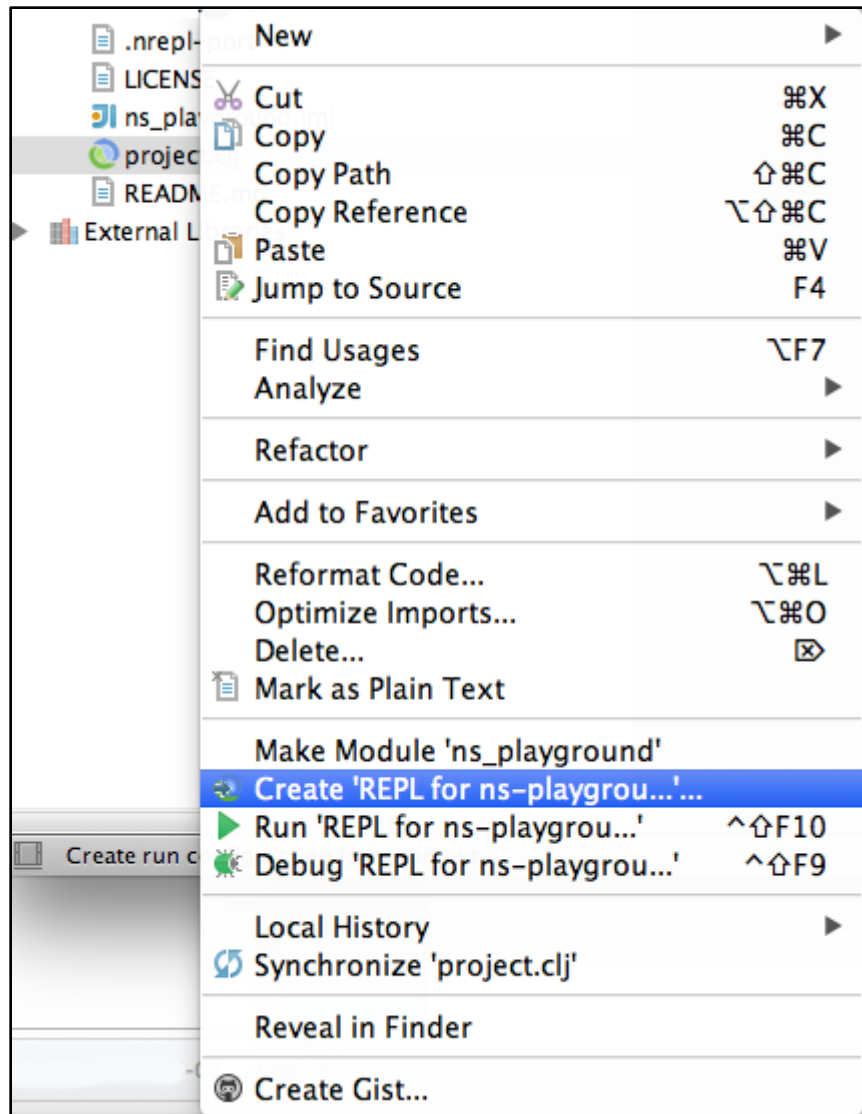
6 directories, 6 files

# iamedu at Eduardos-MacBook-Pro.local in ~/Development/getting-started [20:47:23]
$ |
```

Chapter 2: Namespaces, Packages, and Tests

```
.
├── CHANGELOG.md
├── LICENSE
├── README.md
├── doc
│   └── intro.md
├── project.clj
├── resources
├── src
│   └── ns_playground
│       └── core.clj
└── test
    └── ns_playground
        └── core_test.clj

6 directories, 7 files
```



```
(ns ns-playground.core-test
  (:require [clojure.test :refer :all]
            [ns-playground.core :refer :all]))

(deftest a-test
  (testing "FIXME, I fail."
    (is (= 0 1))))
```

Chapter 3: Interacting with Java

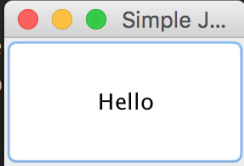
```
# iamedu at Eduardos-MacBook-Pro.local in /Library/Java/JavaVirtualMachines/jdk1.8.0_66.jdk/Contents/Home [5:53:37]
$ jar -tf ./jre/lib/rt.jar | grep AbstractMap
java/beans/MetaData$java_util_AbstractMap_PersistenceDelegate.class
java/util/AbstractMap$1$1.class
java/util/AbstractMap$1.class
java/util/AbstractMap$2$1.class
java/util/AbstractMap$2.class
java/util/AbstractMap$SimpleEntry.class
java/util/AbstractMap$SimpleImmutableEntry.class
java/util/AbstractMap.class
```

```
user=> (def image-stream (io/input-stream "http://imgs.xkcd.com/comics/angular_momentum.jpg"))
#'user/image-stream
user=> (def image (ImageIO/read image-stream))
#'user/image
user=> image
#object[java.awt.image.BufferedImage 0x69a568f0 "BufferedImage@69a568f0: type = 10 ColorModel: #pixelBits = 8 numComponents = 1 color space
= java.awt.color.ICC_ColorSpace@6e4f49da transparency = 1 has alpha = false isAlphaPre = false ByteInterleavedRaster: width = 600 height =
386 #numDataElements 1 dataOff[0] = 0"]
user=> (.getHeight image)
386
user=> |
```

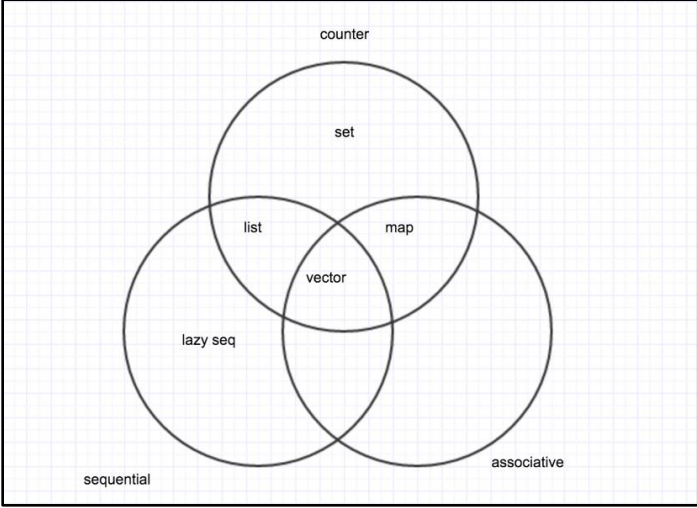
```
1 (defproject thumbnails "0.1.0-SNAPSHOT"
2   :description "FIXME: write description"
3   :url "http://example.com/FIXME"
4   :license {:name "Eclipse Public License"
5             :url "http://www.eclipse.org/legal/epl-v10.html"})
6   :dependencies [[org.clojure/clojure "1.6.0"]
7                 [org.imgscalr/imgscalr-lib "4.2"]]
8   :aot [thumbnails.image-java]
9   :repositories [["jcenter" "http://jcenter.bintray.com/"]])
~
~
~
~
```

```
# iamedu at Eduardos-MacBook-Pro.local in ~/Development/clj/source/chapter03/initial/thumbnails [6:25:18]
$ lein install
Created /Users/iamedu/Development/clj/source/chapter03/initial/thumbnails/target/thumbnails-0.1.0-SNAPSHOT.jar
Wrote /Users/iamedu/Development/clj/source/chapter03/initial/thumbnails/pom.xml
Installed jar and pom into local repo.
```

```
user=> (sample)
#<JFrame javax.swing.JFrame[frame=java.awt.BorderLayout,titl
e=Simple Java Integration,resizablity=java.awt.BorderLayout,titl
ootPane=javax.swing.JRootPane[,0,0],rootPaneCheckingEnabled=true]
ut,alignmentX=0.0,alignmentY=0.0,preferredSize=[0,0],rootPaneCheckingEnabled=true]
=,preferredSize=],rootPaneCheckingEnabled=true]>
user=> Hello world
Hello world
Hello world
Hello world
█
```



Chapter 4: Collections and Functional Programming



Chapter 5: Multimethods and Protocols

```
package shapes;

import java.util.List;
import java.util.Arrays;

public class Main {

    public static double totalArea(List<Shape> shapes) {

        double result = 0;
        for(Shape s : shapes) {
            result += s.getArea();
        }

        return result;
    }

    public static void main(String args[]) {
        System.out.println(totalArea(Arrays.asList(new Circle(5), new Square(3), new Circle(4.5))));
    }
}
```

```
user=> (defmulti walk :type)
#'user/walk
user=>

user=> (defmethod walk ::animal [_] "Just walk")
#object[clojure.lang.MultiFn 0x58c715bf "clojure.lang.MultiFn@58c715bf"]
user=> (defmethod walk ::primate [_] "Primate walk")
#object[clojure.lang.MultiFn 0x58c715bf "clojure.lang.MultiFn@58c715bf"]
user=> (walk {:type ::animal})
"Just walk"
user=>

user=> (walk {:type ::primate})
"Primate walk"
user=>
```



```
user=> (defn avg [& coll]
  #_=> (/ (apply + coll) (count coll)))
#'user/avg
user=> (defn get-race [& ages]
  #_=> (if (> (apply avg ages) 120)
    #_=> :timelord
    #_=> :human))
#'user/get-race
user=> (defmulti travel get-race)
#'user/travel
user=>

user=> (defmethod travel :timelord [& ages]
  #_=> (str (count ages) " timelords travelling by tardis"))
#object[clojure.lang.MultiFn 0x6cc1d65c "clojure.lang.MultiFn@6cc1d65c"]
user=> (defmethod travel :human [& ages]
  #_=> (str (count ages) " humans travelling by car"))
#object[clojure.lang.MultiFn 0x6cc1d65c "clojure.lang.MultiFn@6cc1d65c"]
user=> (travel 2000 1000 100 200)
"4 timelords travelling by tardis"
user=> (travel 80 20 100 40)
"4 humans travelling by car"
user=> |
```

Chapter 6: Concurrency

```
# iamedu at Eduardos-MacBook-Pro.local in ~ [5:23:35]
$ lein new clojure-concurrency
Generating a project called clojure-concurrency based on the 'default' template.
The default template is intended for library projects, not applications.
To see other templates (app, plugin, etc), try `lein help new`.
```

```
-----
clojure-concurrency.core=> (require 'clojure-concurrency.core :reload-all)
WARNING: await already refers to: #'clojure.core/await in namespace: clojure-concurrency.core, being replaced by: #'co.paralleluniverse.pulsar.core/await
WARNING: promise already refers to: #'clojure.core/promise in namespace: clojure-concurrency.core, being replaced by: #'co.paralleluniverse.pulsar.core/promise
WARNING: test already refers to: #'clojure.core/test in namespace: clojure-concurrency.core, being replaced by: #'clojure-concurrency.core/test
nil
clojure-concurrency.core=> (in-ns 'clojure-concurrency.core)
#object[clojure.lang.Namespace 0x736fc8a0 "clojure-concurrency.core"]
clojure-concurrency.core=> (start-thread #(println "Hello threaded world"))
Hello threaded world
nil
```

```
clojure-concurrency.core=> (in-ns 'clojure-concurrency.core)
#object[clojure.lang.Namespace 0x736fc8a0 "clojure-concurrency.core"]
clojure-concurrency.core=> (def p (promise))
#'clojure-concurrency.core/p
clojure-concurrency.core=> (start-thread
  #_=> #(do
    #_=> (deref p)
    #_=> (println "Hello world")))
nil
```

```
-----
clojure-concurrency.core=> (deliver p 5)
Hello world
#object[co.paralleluniverse.pulsar.core$promise$reify__2871 0x6459f6f {:status :ready, :val 5}]
```

```
clojure-concurrency.core=> (def p (promise))
#'clojure-concurrency.core/p
clojure-concurrency.core=> (deliver p 5)
#object[co.paralleluniverse.pulsar.core$promise$reify__2871 0x68ead4cd {:status :ready, :val 5}]
clojure-concurrency.core=> @p
5
clojure-concurrency.core=> (println "Hello world")
Hello world
nil
```

```
# iamedu at Eduardos-MacBook-Pro.local in ~/Development/clj/images/chapter06 [5:40:28]
$ java -XX:+PrintFlagsFinal -version | grep ThreadStackSize

    intx CompilerThreadStackSize           = 0                {pd product}
    intx ThreadStackSize                   = 1024             {pd product}
    intx VMThreadStackSize                  = 1024             {pd product}
java version "1.8.0_66"
Java(TM) SE Runtime Environment (build 1.8.0_66-b17)
Java HotSpot(TM) 64-Bit Server VM (build 25.66-b17, mixed mode)
```

```
clojure-concurrency.core=> (future (dosync (test)))
Transaction started
0 20000
#object[clojure.core$future_call$reify__6736 0x490bc5f4 {:status :pending, :val nil}]
clojure-concurrency.core=> (future (dosync (Thread/sleep 4000) (ref-set account 5)))
#object[clojure.core$future_call$reify__6736 0x2d31b139 {:status :pending, :val nil}]
clojure-concurrency.core=> 1 20000
2 20000
Transaction started
0 5
1 5
2 5
3 5
4 5
```

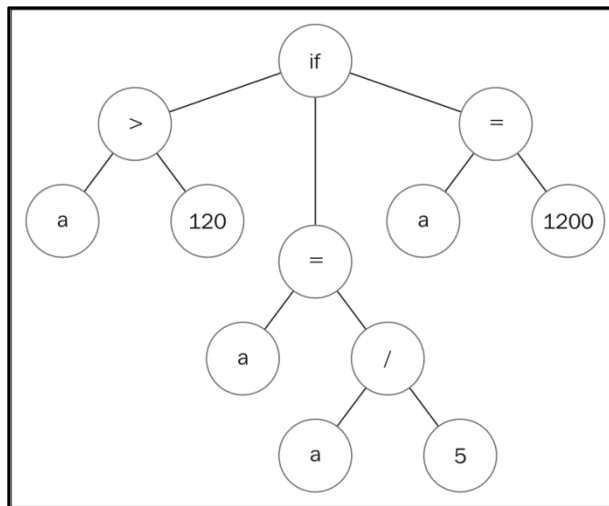
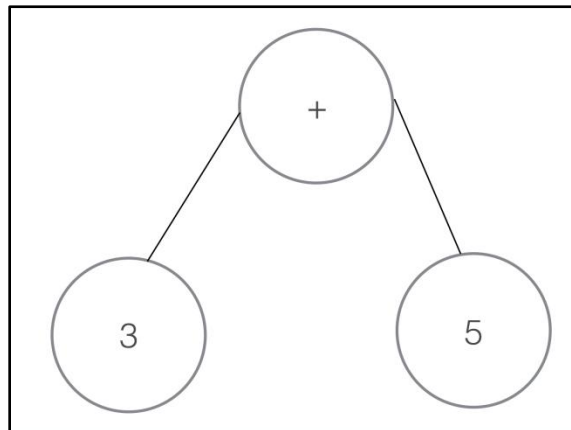
```
clojure-concurrency.core=> (future (move account-a account-b 50))
#object[clojure.core$future_call$reify__6736 0x17f9691c {:status :pending, :val nil}]

Transaction started
clojure-concurrency.core=> @started
true
clojure-concurrency.core=> (dosync (ref-set account-a 20))
20
clojure-concurrency.core=>

clojure-concurrency.core=>
Transaction started

Transaction finished
```

Chapter 7: Macros in Clojure



```
user=> (defmacro my-if [test positive negative]
  #_=> `(if ~test ~positive ~negative))
#'user/my-if
user=> (macroexpand-1
  #_=> '(my-if (> a 200)
  #_=> (do
  #_=> (println "Bigger than 200")
  #_=> :bigger)
  #_=> (do
  #_=> (println "Smaller than 200")
  #_=> :smaller)))
(if (> a 200) (do (println "Bigger than 200") :bigger) (do (println "Smaller than 200") :smaller))
```

```

user=> (defmacro >-macro [& params]
  #_=> `(> ~params))
#'user/>-macro
user=>

user=> (macroexpand '(>-macro 5 4 3))
(clojure.core/> (5 4 3))
user=> (>-macro 5 4 3)
ClassCastException java.lang.Long cannot be cast to clojure.lang.IFn user/eval1219 (form-init7702615529919007918.clj:1)

```

```

user=> (defmacro >-macro [& params]
  #_=> `(> ~@params)) ;; In the end this works as if you had written
#'user/>-macro
user=>                ;; (> 5 4 3)

user=>

user=> (macroexpand '(>-macro 5 4 3))
(clojure.core/> 5 4 3)
user=> (>-macro 5 4 3)
true

```

```

user=> (def a-var "hello world")
#'user/a-var
user=> (defmacro error-macro [& params]
  #_=> `(let [a-var "bye world"]
  #_=>      (println a-var)))
#'user/error-macro
user=> (macroexpand-1 '(error-macro))
(clojure.core/let [user/a-var "bye world"] (clojure.core/println user/a-var))
user=> (error-macro)

CompilerException java.lang.RuntimeException: Can't let qualified name: user/a-var, compiling: (/private/var/folders/4s/yxd1cnqn17dd30b_z4b27kyw0000gn/T/form-init7702615529919007918.clj:1:1)

```

```

user=> (def a-var "hello world")
#'user/a-var
user=> (defmacro error-macro [& params]
  #_=> (let [a-var-name (gensym 'a-var)]
  #_=>   `(let [~a-var-name "bye world"]
  #_=>     (println ~a-var-name))))
#'user/error-macro
user=> (macroexpand-1 '(error-macro))
(clojure.core/let [a-var1274 "bye world"] (clojure.core/println a-var1274))
user=> (error-
error-handler error-macro error-mode
user=> (error-m
error-macro error-mode
user=> (error-macro)
bye world
nil

```