# 多态 & Java Interop

刘家财
http://liujiacai.net/

# 提纲

- 多态
  - 基于dispatch函数：multimethod
  - 基于类型：record/protocol/type
- Java Interop

# multimethods 多**态**方法

# multimethods 多**态**方法

- 基于 dispatch 函数确定函数调用

```
(defmulti compiler :os)
(defmethod compiler ::unix [m] (get m :c-compiler))
(defmethod compiler ::osx  [m] (get m :llvm-compiler))

(compiler {:os :unix :c-compiler "gcc" :home "/home"})
;; => gcc
(compiler {:os :osx :llvm-compiler "clang" :home "/Users"})
;; => clang
```

# dispatch 时的继承

```
(defmulti home :os)
(defmethod home ::unix [m] (get m :home))
(home unix)
;=> "/home"
(home osx)
;   No method in multimethod 'home' for dispatch value: :user/osx

(derive ::osx ::unix)
(home osx)
;=> "/Users"

(isa? ::osx ::unix)
;=> true
(isa? ::unix ::osx)
;=> false
```

# 解决**继**承**时**的冲突

```
(derive ::osx ::bsd)
(defmethod home ::bsd [m] "/home")
(home osx)
; java.lang.IllegalArgumentException: Multiple methods in multimethod

(prefer-method home ::unix ::bsd)
(home osx)
;=> "/Users"
```

record/protocol/type

abstraction-oriented programming

# Record

具有 type 的 map

```
(defrecord TCPServer [^String host
                      ^int port
                      ^int backlog])

(TCPServer. "localhost" 8080 5)
;;=> #user.TCPServer{:host "localhost", :port 8080, :backlog 5}

(assoc (TCPServer. "localhost" 8080 5) :port 9090)
;;=> #user.TCPServer{:host "localhost", :port 9090, :backlog 5}

(dissoc (TCPServer. "localhost" 8080 5) :port)
;; ⚠️这里返回 map
;;=> {:host "localhost", :backlog 5}
```

# record 优势

- 明确需要的 keys
- 性能提升
  - 创建速度更快
  - 占用内存更少（可以存 primitives）
  - 查询速度更快

# 与 map 的区别

- record 不能作为 function 使用
- record != 具有相同 k/v 的 map

# protocol

## 让 record 动起来

类似：java 的 interface、C++ 的 virtual class

```
(defprotocol Service
  (reload! [service core]
           "Informs the service of a change in core.")
  (start! [service]
           "Starts a service. Must be idempotent.")
  (stop!  [service]
          "Stops a service. Must be idempotent.")
  (conflict? [service1 service2]
             "Do these two services conflict with one another? Adding
             a service to a core *replaces* any conflicting services."))
```

```
(defrecord TCPServer
  [^String host
   ^int port
   ^int backlog]
  Service
  (conflict? [this other]
   (and (instance? TCPServer other)
        (= host (:host other))
        (= port (:port other))))

  (reload! [this new-core]
    (reset! core new-core))

  (stop! [this]
    (.close ServerManager host port)
    (info "TCP server" host port "closed"))

  (start! [this]
    (->> (InetSocketAddress. host port)
     (.bind bootstrap)
     (.sync)
     (.channel)
     (.add channel-group))
    (info "TCP server" host port "online")))
```

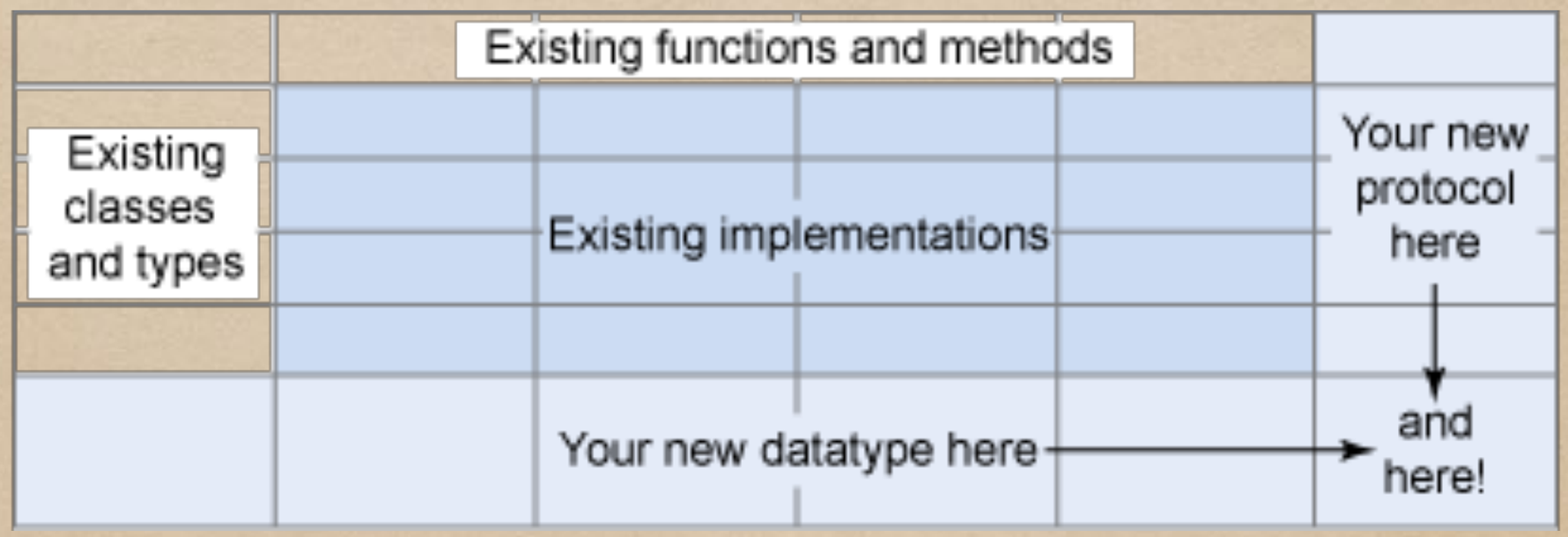
# extend-type

```
(defprotocol StringOps
  (rev [s]))

(extend-type String
  StringOps
  (rev [s] (clojure.string/reverse s)))

(rev "Works")
;=> "skroW"
```

为已有类添加新操作

# Record + Protocol

- Clojure's answer to Expression Problem



# type

## 轻量级的 record

# defrecord 实现细节

默认继承以下类：

```
[clojure.lang.IRecord
 clojure.lang.IHashEq
 clojure.lang.IObj
 clojure.lang.ILookup
 clojure.lang.IKeywordLookup
 clojure.lang.IPersistentMap
 java.util.Map
 java.io.Serializable]
```

# deftype

- 只继承 clojure.lang.IType
- 使用场景：实现不是 map-like 的数据结构
- Why have both deftype and defrecord?

# deftype

```
(deftype InfiniteConstant [i]
  clojure.lang.ISeq
  (seq [this]
    (lazy-seq (cons i (seq this)))))

(take 3 (InfiniteConstant. 5))
;=> (5 5 5)
```

# Java Interop

# Java Interop

- doto （**调**用同一**对**象的多个方法）
- .. （**级联调**用）
- case 不能与 enum 一起使用
- varargs 可**变**参数 （make-array 解决）
- reify vs. proxy

# Thank You.

群名称：SICP读书群

群　号：119845407

公众号