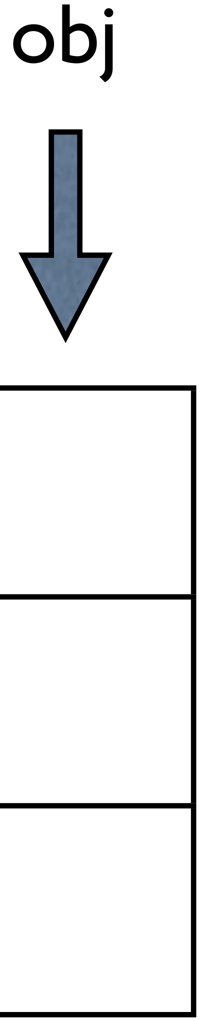
The CLOS Metaobject Protocol

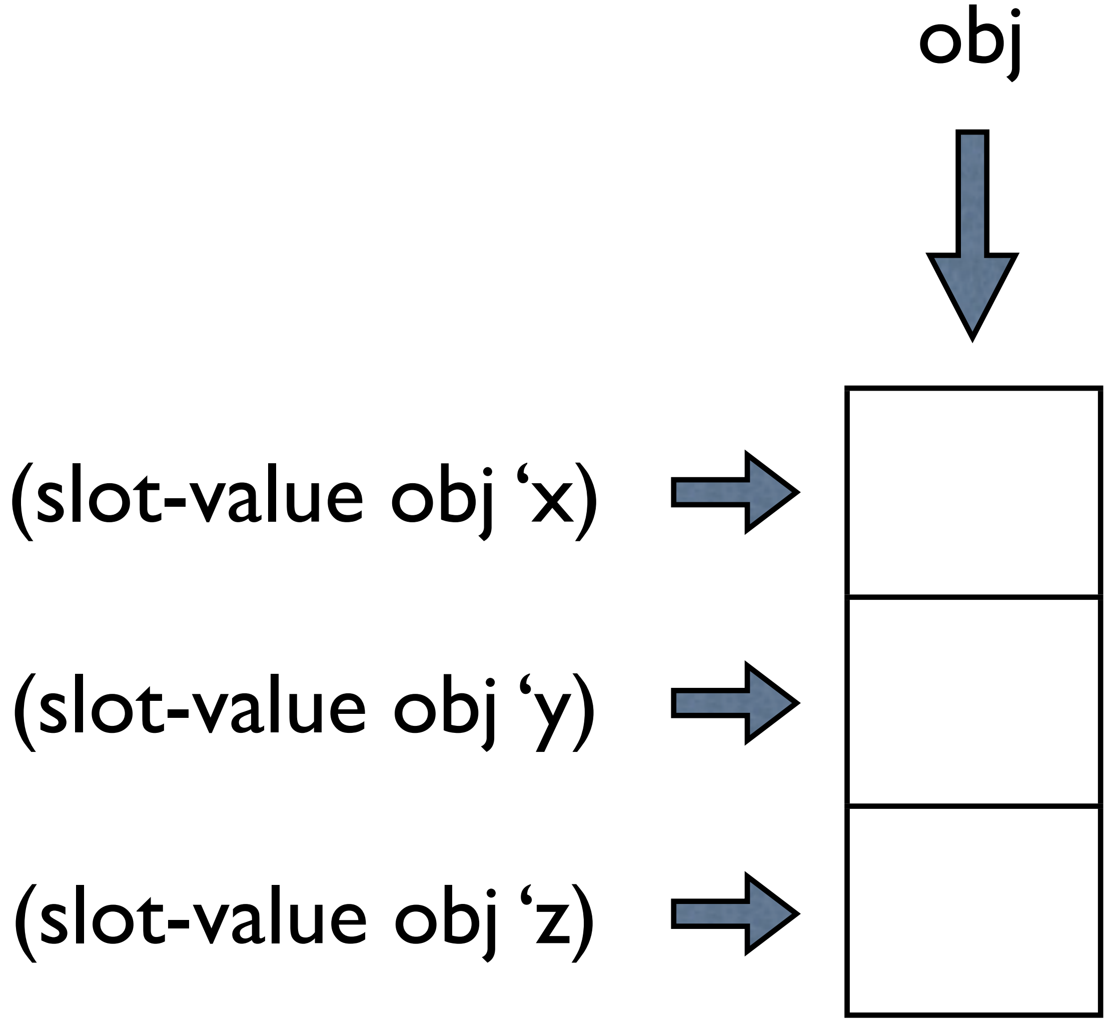
OOP: What is an object?

"An object has state, behavior, and identity."
(Grady Booch, 1991)

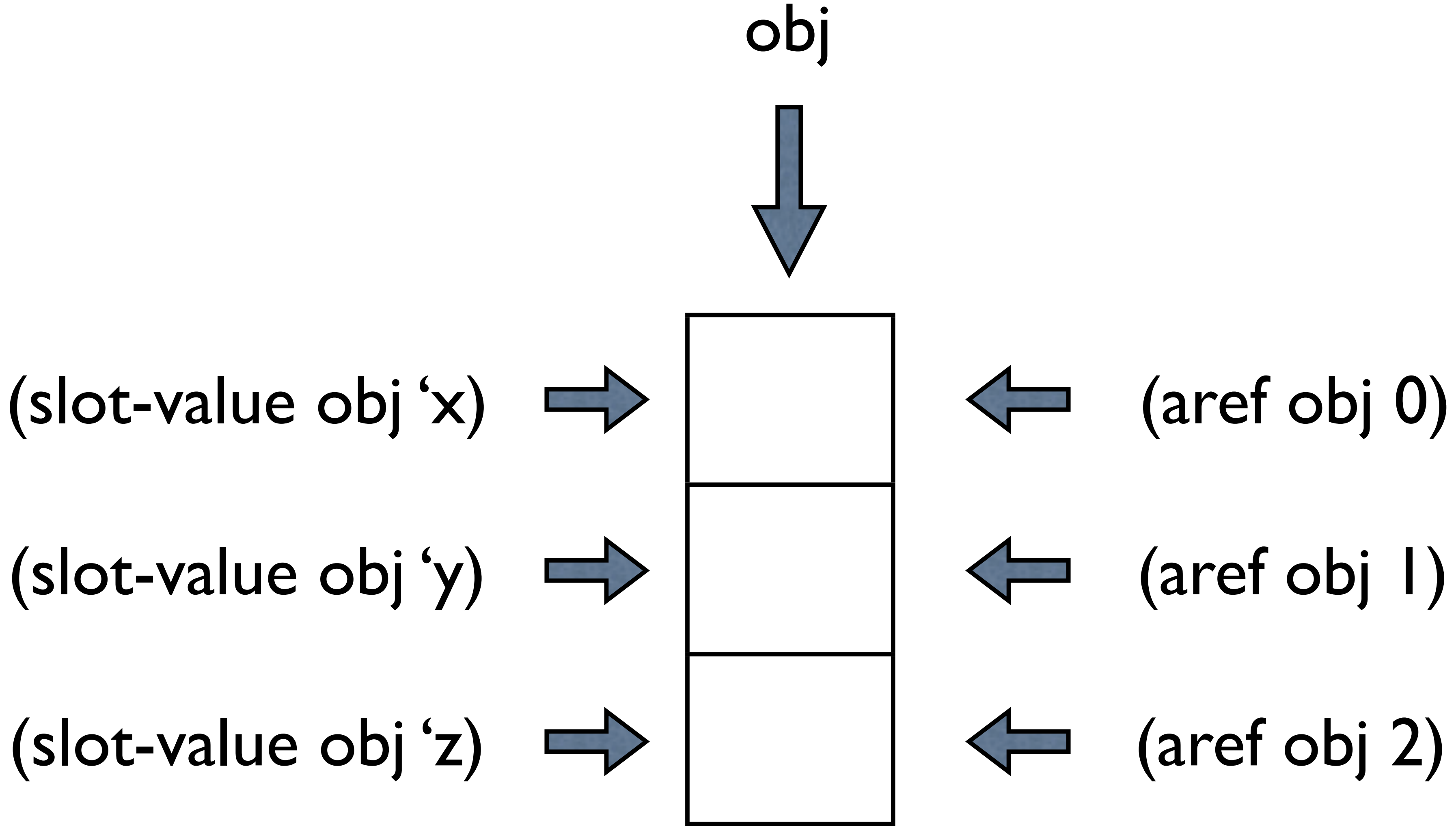
OOP: State



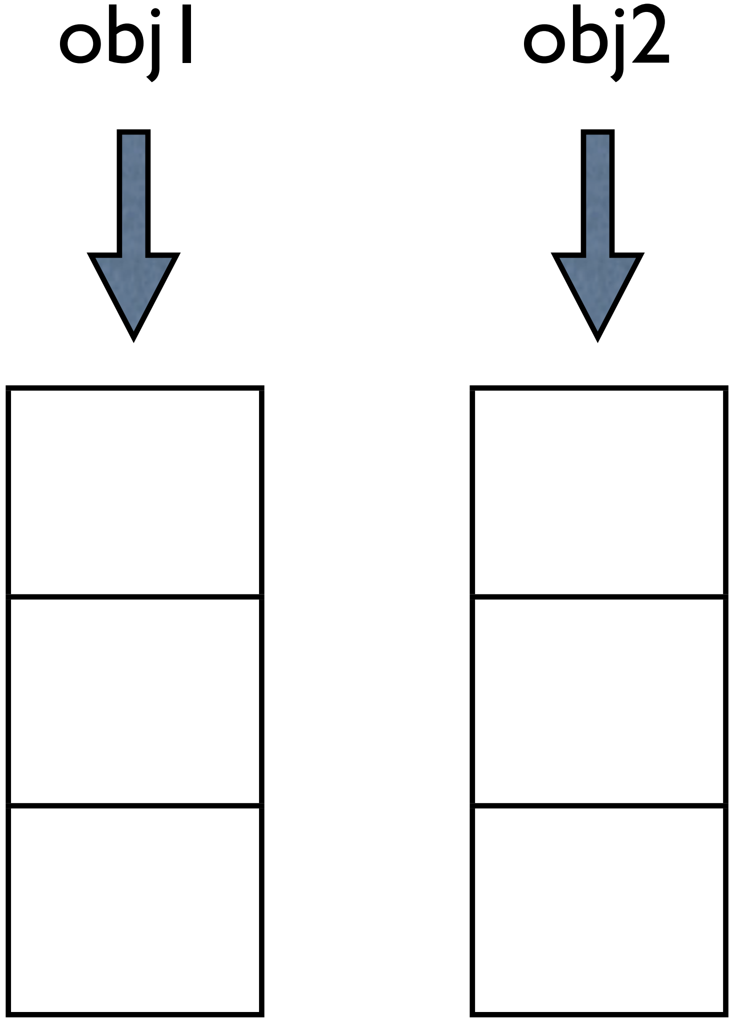
OOP: State



OOP: State

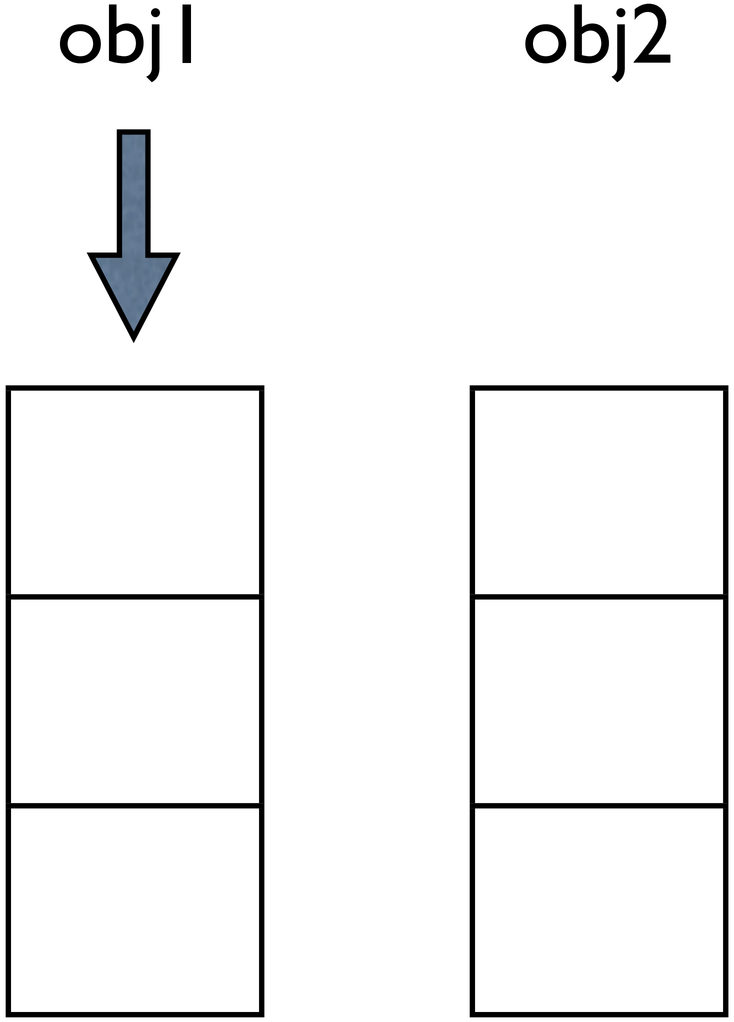

obj
(slot-value obj 'x)
(slot-value obj 'y)
(slot-value obj 'z)
(aref obj 0)
(aref obj 1)
(aref obj 2)

OOP: Identity



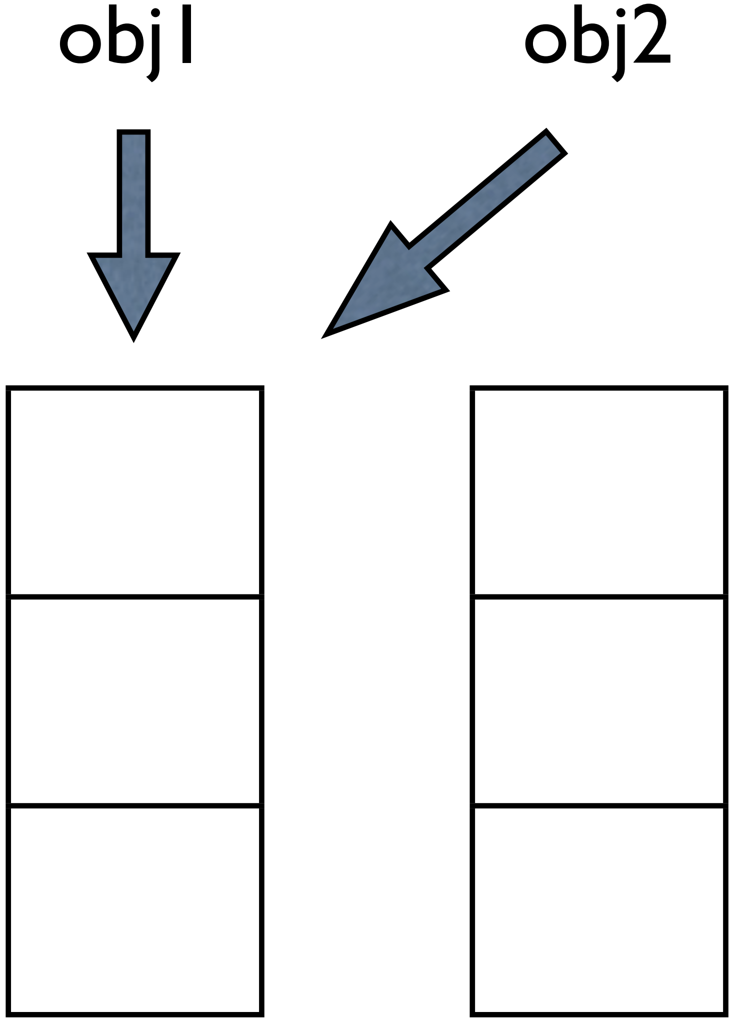
```
(eq obj1 obj2)
=> nil
```

OOP: Identity



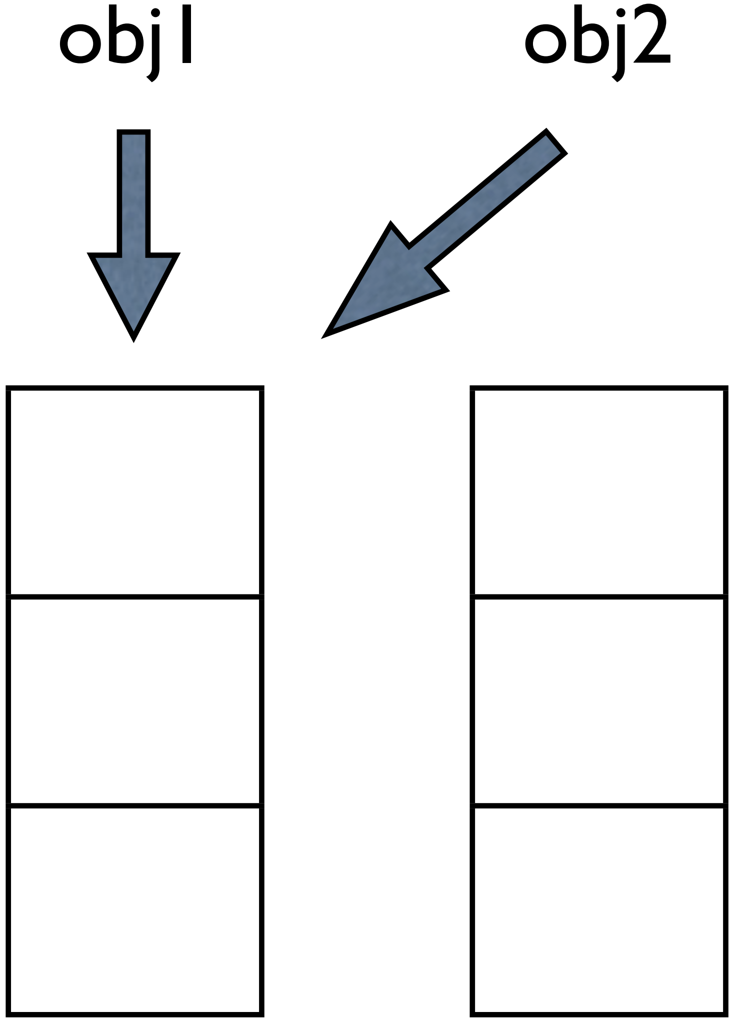
```
(eq obj1 obj2)
=> nil

(setf obj2 obj1)
```

OOP: Identity



```
(eq obj1 obj2)
=> nil

(setf obj2 obj1)
```

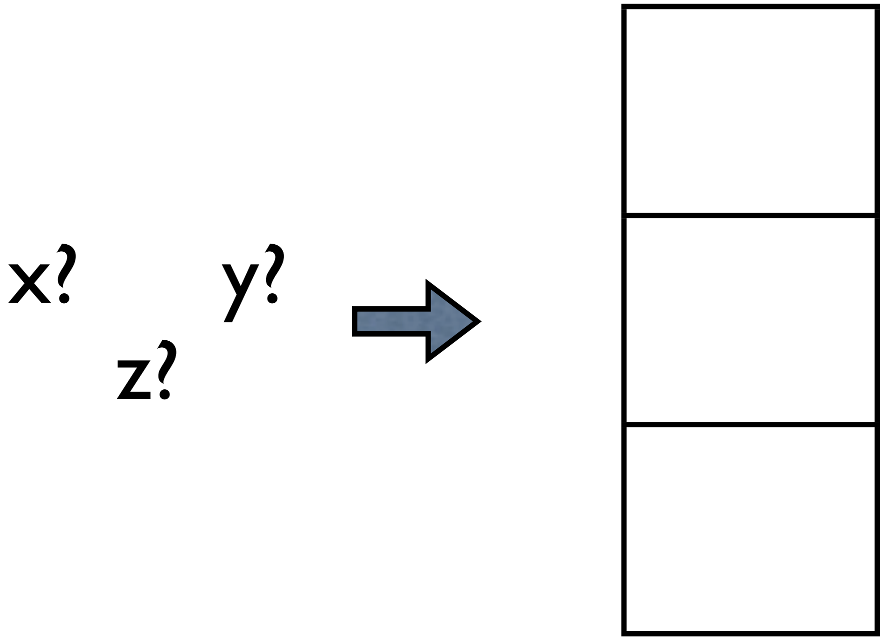
OOP: Identity



```
(eq obj1 obj2)
=> nil

(setf obj2 obj1)

(eq obj1 obj2)
=> t
```

OOP:
How to map slots?

```
(defclass point ()
   (x y))

(defclass point-3d (point)
   (z))
```



OOP: How to map slots?

1. compute class precedence list
2. compute slots
3. determine slot locations

OOP:
How to map slots?

1. (compute-class-precedence-list ...)
2. (compute-slots ...)
3. (slot-definition-location ...)

The Idea!

- Make compute-class-precedence-list, compute-slots, and so on, generic functions!
- Allow changes to the CLOS object model!
- Question: How to distinguish between standard and non-standard behavior?

Hierarchy for metaobject classes

$$\begin{tikzcd}
\text{t} \\
\text{standard-object} \arrow[u] \\
\text{class} \quad \text{slot-definition} \quad \text{generic-function} \quad \text{method} \arrow[u]
\end{tikzcd}$$

Hierarchy for metaobject classes

$$\begin{tikzcd}
 & & \text{t} & & \\
 & & \text{standard-object} \arrow[u] & & \\
\text{class} & \text{slot-definition} & {} \arrow[u] & \text{generic-function} & \text{method} \\
\text{standard-class} \arrow[u] & & & &
\end{tikzcd}$$

Hierarchy for metaobject classes

$$\begin{tikzcd}
 & t & & \\
 & \text{standard-object} \arrow[u] & & \\
\text{class} & \text{slot-definition} \arrow[u] & \text{generic-function} & \text{method} \\
\text{standard-class} \arrow[u] & & \text{standard-generic-function} \arrow[u] &
\end{tikzcd}$$

Hierarchy for metaobject classes

$$\begin{tikzcd}
 & & \text{t} & & \\
 & & \text{standard-object} \arrow[u] & & \\
\text{class} & \text{slot-definition} & {} \arrow[u] & \text{generic-function} & \text{method} \\
\text{standard-class} \arrow[u] & & & \text{standard-generic-function} \arrow[u] & \text{...}
\end{tikzcd}$$

Class metaobject classes

-
  ```
  (defclass persistent-class (standard-class)
     ((database-connection ...)))
  ```

-
  ```
  (defclass person ()
     ((name ...)
      (address ...))
     (:metaclass persistent-class))
  ```

The Instance Structure Protocol

-
  ```
  (defmethod person-name ((object person))
    (slot-value object ‘name))
  ```

-
  ```
  (defun slot-value (object slot)
    (slot-value-using-class
      (class-of object) object slot))
  ```

-
  ```
  (defmethod slot-value-using-class
    ((class standard-class) object slot)
    (aref ...))
  ```

The Instance Structure Protocol

- (defmethod slot-value-using-class
  ((class persistent-class) object slot)
  (fetch-slot-from-database ...))

Other Protocols

- Initialization protocols
- Class finalization protocol
- Instance structure protocol
- Funcallable instances
- Generic function invocation protocol
- Dependent maintenance protocol

Example: The Python object model

1. Define a mix-in for hashtable-based slots.
2. Ensure that this mix-in is used.
3. Modify the slot access protocol.

Links (Common Lisp)

- Andreas Paepcke,
  “User-level Language Crafting”
- G. Kiczales, J. des Rivieres, D. Bobrow,
  “The Art of the Metaobject Protocol”
- http://common-lisp.net/project/closer/

Links (Scheme)

- http://community.schemewiki.org/?object-systems

Links (Smalltalk)

- http://www.laputan.org/#Reflection

Links (C++)

- Ira Forman, Scott Danforth, "Putting Metaclasses to Work"

Links (Java)

- Ira Forman, Nate Forman, “Java Reflection in Action”