CLOS

Classes

```
(defclass person (standard-object)
    ((name :accessor person-name
           :initarg :name
           :allocation :instance)
     (address :accessor person-address
              :initarg :address
              :allocation :instance))
    (:documentation
     "This is a class for person objects."))
```

Class and Superclasses

```
(defclass person (standard-object)
    ((name :accessor person-name
           :initarg :name
           :allocation :instance)
     (address :accessor person-address
              :initarg :address
              :allocation :instance))
    (:documentation
     "This is a class for person objects."))
```

Slots and Slot Options

```
(defclass person (standard-object)
    ((name :accessor person-name
           :initarg :name
           :allocation :instance)
     (address :accessor person-address
              :initarg :address
              :allocation :instance))
    (:documentation
     "This is a class for person objects."))
```

Class Options

```
(defclass person (standard-object)
    ((name :accessor person-name
           :initarg :name
           :allocation :instance)
     (address :accessor person-address
              :initarg :address
              :allocation :instance))
    (:documentation
     "This is a class for person objects."))
```

CLOS: Instances and Accessors

- (defparameter pascal
  (make-instance ‘person
  :name “Pascal”
  :address “Brussels”))
- (person-name pascal)
  => “Pascal”

Generic Functions and Methods

- (defgeneric display (object))
- (defmethod display ((object person))
  (print (person-name object))
  (print (person-address object)))
- (display pascal)

Inheritance

- (defclass employee (person)
  ((employer :accessor person-employer))))
- (defmethod display ((object employee))
  (call-next-method)
  (print (person-employer object)))

Sidenote: Keywords and Property Lists

- (getf '(:x 5 :y 6) :y)
  => 6
- (getf '(:x 5 :y 6 :x 11) :x)
  => 5
- (make-instance 'person
  :name "Pascal"
  :address "Brussels")

Sidenote: Keywords and Property Lists

- (defmethod make-person
  (&key name address)
  (make-instance 'person
  :name name
  :address address))
- (make-person :name "Pascal"
  :address "Brussels")

Sidenote: apply

- The following forms are all equivalent...
  - (+ 3 4 5)
  - (apply (function +) (list 3 4 5))
  - (apply (function +) 3 (list 4 5))
  - (apply (function +) 3 4 (list 5))
  - (apply (function +) 3 4 5 '())

Sidenote: Keywords and apply

- (defmethod make-person
  (&rest args
  &key name address)
  (apply (function make-instance)
  'person
  args))

Sidenote: Keywords and apply

- (defmethod make-person
  (&rest args
  &key name address)
  (apply (function make-instance)
  'person
  :name (concatenate 'string
  "" name "")
  args))