TÉCNICO LISBOA

A Class System for Common LISP

Group 3:

-> Luís Borges, 78349
-> Paulo Ritto, 78929
-> João Miguens, 79201

Representing an object

- An object is represented by a list containing
  - ❖ The name of its class
  - ❖ An hash table mapping every atribute to its value

```
[2]> (def-class person name age)
PERSON-SET-AGE!
[3]> (setf p (make-person :name "Luis" :age 21))
(PERSON #S(HASH-TABLE :TEST FASTHASH-EQL (AGE . 21) (NAME . "Luis")))
[4]>
```

Two global variables

```
(defvar *class-inheritance-lists* (make-hash-table))
```

Each class is mapped to a list of all its superclasses

```
(defvar *class-attributes-lists* (make-hash-table))
```

Each class is also mapped to a list of all its attributes (inherited as well)

Our main macro

- The first thing we do is to fill the global variables **class-inheritance-list** and **class-attributes-lists** according to the class we're defining, its superclasses and the slots we provide.

```
[2]> (gethash 'IST-STUDENT *class-attributes-lists*)
(COURSE NAME AGE ACTIVITY SCHEDULE) ;
```

```
Break 1 [4]> (gethash 'IST-STUDENT *class-inheritance-lists*)
(STUDENT SPORTSMAN PERSON) ;
```

- This is done iteratively using the functions *create-precedence-list* and *create-attributes-list*

- Our constructor will ask the user to input all of the class' slots, inherited as well (*attr-list* is obtained in **class-attributes-list**)
- It will return the list that represents the object
- *create-object* is a function that given the user's assignments, i.e. ("Luis", 21) constructs the hashmap.

```
(defun ,(intern (format nil "MAKE-~a" className)) (&key ,@attr-list) ;Constructor
    (list ',className (create-object ',className (vector ,@attr-list))))
```

Recognizer

- Given the representation of the object, an object belongs to a class if it is a direct instance of that class or if that class is in the corresponding *class-inheritance-lists* key.

```
(defun ,(intern (format nil "~a?" className)) (,className) ;Verifier
    (if (or (equalp ',className (nth 0 ,className)) (gethash (nth 0 ,className) *class-inheritance-lists*))
        T
        NIL))
```

Getters

- Loop through all the attributes to define the getters
  - ❖ If the argument of a getter is an instance of the class, then it's a simple hashmap access

```
,@(loop for i from 0 to (1- (length (gethash className *class-attributes-lists*))) collect ;Getters
    `(defun ,(intern (format nil "~a-~a" className (nth i (gethash className *class-attributes-lists*)))) (,className)
       (if (,(intern (format nil "~a?" className)) ,className)
           (gethash ',(nth i (gethash className *class-attributes-lists*)) (nth 1 ,className)) 'ERROR  )))
```

- **Extension:** setters are defined almost exactly the same as getters