

Introduction Classes et Objets Héritage Polymorphisme

Approches Objet de la Programmation

~ CLOS : Common Lisp Object System ~

Didier Verna
EPITA / LRDE

didier@lrde.epita.fr

 lrde/~didier
  @didievierna
  didier.verna
  in/didievierna

Introduction Classes et Objets Héritage Polymorphisme

Plan

- Introduction
- Classes et Objets
 - Classes et Objets
 - Portée et Accessibilité de l'Information
 - Intégration Classes / Types
- Héritage
 - Rappels
 - Modèle d'héritage
 - Problèmes Liés à l'Héritage
- Polymorphisme
 - Rappels
 - Fonctions Génériques et Méthodes
 - Relation (sous-)Classe / (sous-)Type

AOP / CLOS : Common Lisp Object System – Didier Verna 2/31

Introduction Classes et Objets Héritage Polymorphisme

Plan

- Introduction
- Classes et Objets
- Héritage
- Polymorphisme

AOP / CLOS : Common Lisp Object System – Didier Verna 3/31

Introduction Classes et Objets Héritage Polymorphisme

Origine

- ▶ Expérimentations diverses depuis les années 70
 - ▶ Smalltalk, idées originales etc.
- ▶ 1986 : ACM Lisp and Functional Programming Conference
 - ▶ Groupe informel pour standardiser un système objet
 - Forte pression pour la standardisation du langage*
- ▶ Comité X3J13 pour la standardisation de Common Lisp
 - ▶ ANSI standard X3.226:1994 (R1999)
- ▶ Résultat : un « best-of » des idées / systèmes de l'époque
 - ▶ [New] Flavors (Symbolics Inc., MIT Lisp Machines)
 - ▶ [Portable] Common Loops (Xerox PARC, Interlisp-D)
 - ▶ Lucid

AOP / CLOS : Common Lisp Object System – Didier Verna 4/31

Introduction Classes et Objets Héritage Polymorphisme

Caractéristiques I

- ▶ Stratifié, flexible
 - ▶ API (syntaxique) $\Rightarrow$ API (fonctionnelle) $\Rightarrow$ Implémentation
- ▶ Fonctions génériques ($\neq$ envoi de message)
 - ▶ Envoi de message inadapté aux opérations n-aires
 - ▶ Multi-méthodes
 - ▶ Extension naturelle des fonctions classiques
- ▶ Héritage multiple
 - ▶ Système de précédence de classe linéarisé
- ▶ Combinaison de méthodes
 - ▶ Invocation simple / multiple
 - ▶ Plusieurs en standard, programmable

AOP / CLOS : Common Lisp Object System – Didier Verna 5/31

Introduction Classes et Objets Héritage Polymorphisme

Caractéristiques II

- ▶ Ordre supérieur (MOP)
 - ▶ Méta-objets : fonctions génériques, classes *etc.*
 - ▶ 1^{re} classe : manipulation anonyme *etc.*
- ▶ Typage dynamique
 - ▶ C'est du Lisp!
- ▶ Pas d'encapsulation ni de protection
 - ▶ Orthogonal à la programmation orientée-objet
 - ▶ Cf. accesseurs et packages

AOP / CLOS : Common Lisp Object System – Didier Verna 6/31

Introduction **Classes et Objets** Héritage Polymorphisme

Plan

Introduction

Classes et Objets

- Classes et Objets
- Portée et Accessibilité de l'Information
- Intégration Classes / Types

Héritage

Polymorphisme

AOP / CLOS : Common Lisp Object System – Didier Verna 7/31

Introduction **Classes et Objets** Héritage Polymorphisme

Classes

La classe human

```
(defclass human () (name size birth-year))
```

- ▶ Remarques
 - ▶ Définition fonctionnelle (defclass est une macro)
 - ▶ Cycle de vie dynamique (à la fois type et objet)
 - ▶ Typage dynamique
 - ▶ Pas de méthodes membre
 - ▶ Pas de mécanisme de protection
 - ▶ Pas de classes abstraites, finales *etc.*

AOP / CLOS : Common Lisp Object System – Didier Verna 8/31

Introduction Classes et Objets Héritage Polymorphisme

Instanciation : Allocation

Mécanisme général

```
(make-instance 'human)
```

- ▶ **Remarques**
 - ▶ Instanciation fonctionnelle ($\neq$ constructeur)
 - ▶ Fonction commune à toutes les classes
 - ▶ Rappel : ramasse-miettes $\Rightarrow$ pas de destructeur
- ▶ **Problème** : initialisation des slots

AOP / CLOS : Common Lisp Object System – Didier Verna 9/31

Introduction Classes et Objets Héritage Polymorphisme

Instanciation : Initialisation I

Arguments d'initialisation

```
(defclass human ()
  ((name :initarg :name)
   (size :initarg :size)
   (birth-year :initarg :birth-year)))

(make-instance 'human
  :name "Alain Térieur" :size 1.80 :birth-year 1970)
```

- ▶ **Problème** : initialisation par défaut

AOP / CLOS : Common Lisp Object System – Didier Verna 10/31

Introduction Classes et Objets Héritage Polymorphisme

Instanciation : Initialisation II

Valeurs d'initialisation

```
(defclass human ()
  ((name :initarg :name)
   (size :initarg :size)
   (birth-year :initarg :birth-year :initform 1970)))

(make-instance 'human :name "Alain Térieur" :size 1.80)
```

- ▶ **Problème** : initialisation obligatoire

AOP / CLOS : Common Lisp Object System – Didier Verna 11/31

Introduction Classes et Objets Héritage Polymorphisme

Instanciation : Initialisation III

Fonction d'instanciation

```
(defun make-human (name size &rest keys &key birth-year)
  (apply #'make-instance 'human :name name :size size
    keys))

;; (make-human "Alain Térieur" 1.80)
;; (make-human "Alex Térieur" 1.80 :birth-year 1939)
```

- ▶ **Remarques**
 - ▶ make-<class> est conventionnel
 - ▶ Privilégier les *keywords* aux paramètres optionnels
 - ▶ Sémantique d'appel $\neq$ Surcharge

AOP / CLOS : Common Lisp Object System – Didier Verna 12/31

Introduction Classes et Objets Héritage Polymorphisme

Portée de l'Information

Slots locaux vs. partagés

```
(defclass human ()
  ((population :allocation :class :initform 0)
   (name :initarg :name)
   (size :initarg :size)
   (birth-year :initarg :birth-year :initform 1970)))
```

- **Remarques**
 - Par défaut : `:allocation :instance`
 - Pas d'accès standard aux slots partagés à travers les classes
 - Pas d'équivalent direct des méthodes de classe (statiques)

AOP / CLOS : Common Lisp Object System – Didier Verna 13/31

Introduction Classes et Objets Héritage Polymorphisme

Accessibilité de l'Information I

Mécanisme Général

```
(slot-value alain 'birth-year)
```

- **Remarques**
 - Accès fonctionnel ($\neq$ syntaxique)
 - Fonction commune à toutes les classes
 - Pas de mécanisme de protection (peu de sens)
Pas même les packages, pas d'amitié etc.
- **Problème** : manque d'abstraction

AOP / CLOS : Common Lisp Object System – Didier Verna 14/31

Introduction Classes et Objets Héritage Polymorphisme

Accessibilité de l'Information II

Accesseurs

```
(defclass human ()
  ((name :initarg :name :reader name :writer rename)
   (size :initarg :size :accessor size)
   (birth-year :initarg :birth-year :initform 1970 :reader birth-year)))

(name alain) ;; => "Alain Térieur"
(rename "Alain Verse" alain) ;; => "Alain Verse"

(size alain) ;; => 1.80
(setf (size alain) 1.78) ;; => 1.78

(birth-year alain) ;; => 1970
(setf (birth-year alain) 1971) ;; error
```

- **Remarque** : Génération automatique de fonctions (génériques)

AOP / CLOS : Common Lisp Object System – Didier Verna 15/31

Introduction Classes et Objets Héritage Polymorphisme

Comportement Hors Classe

Exemple

```
(defun hello (human)
  (format t "Hello! I'm ~A, ~Am, ~Ayo.~%"
    (name human)
    (size human)
    (age human))
  (values))
```

- **Remarques**
 - Méthode dans les systèmes traditionnels
 - Simple fonction ici (fonction générique plus tard)
 - Généricité quand même (typage dynamique)

AOP / CLOS : Common Lisp Object System – Didier Verna 16/31

Introduction Classes et Objets Héritage Polymorphisme

Classes, Objets, Types

- ▶ Forte intégration type / classe
 - ▶ Hiérarchie unique : classe racine `t` (class-of)
 - ▶ Correspondance classe / type (type-of, typep)
 - ▶ Correspondance type natif / classe
Spécialisation possible, instantiation interdite
- ▶ Note : sous-classage $\iff$ sous-typage (subtypép)
 - ▶ Cf. systèmes traditionnels, mais en plus robuste
- ▶ MOP (réflexivité totale : introspection / intercession)
 - ▶ Les classes sont des objets (système de type de 1^{re} classe)
 - ▶ Métaclasse : classe d'une classe
 - ▶ `standard-class` : classe des classes utilisateurs (entre autres)
Classe des classes agrégatives, circularité

AOP / CLOS : Common Lisp Object System – Didier Verna 17/31

Introduction Classes et Objets Héritage Polymorphisme

Plan

Introduction

Classes et Objets

Héritage

- Rappels
- Modèle d'héritage
- Problèmes Liés à l'Héritage

Polymorphisme

AOP / CLOS : Common Lisp Object System – Didier Verna 18/31

Introduction Classes et Objets Héritage Polymorphisme

Rappels : Relations Entre Classes

- ▶ **Copier-Coller** : c'est mal !
- ▶ **Agrégation** : relation « ensemble / parties »
- ▶ **Composition** : agrégation plus forte
- ▶ **Héritage** : forme d'inclusion implicite
Héritage de structure et de comportement

Héritage

```
(defclass employee (human)
  ((company :initarg :company :reader company)
   (salary :initarg :salary :accessor salary)
   (hiring-year :initarg :hiring-year)))
```

AOP / CLOS : Common Lisp Object System – Didier Verna 19/31

Introduction Classes et Objets Héritage Polymorphisme

Spécificités

- ▶ **Héritage implicite** :
 - ▶ Hiérarchie de classes unique
 - ▶ `user-class` $\rightarrow$... $\rightarrow$ `standard-object` $\rightarrow$ `t`
 - ▶ Aucun slot, cf. `print-object` etc.
- ▶ **Héritage des slots** :
 - ▶ $\neq$ systèmes traditionnels
 - ▶ Un slot unique (pas d'ambiguïté)
- ▶ **Héritage des options** :
 - ▶ `initargs`, `initforms` etc.
 - ▶ Modalités différentes selon les options
- ▶ **Héritage des méthodes** :
 - ▶ Sous-classage $\iff$ sous-typage
- ▶ **Héritage Multiple**

AOP / CLOS : Common Lisp Object System – Didier Verna 20/31

Introduction Classes et Objets **Héritage** Polymorphisme

Instanciation en Présence d'Héritage

Exemple

```
(defun make-employee (name size company salary hiring-year
                      &rest keys &key birth-year)
  (let ((employee (apply #'make-instance 'employee
                           :name name :size size
                           :company company :salary salary
                           :hiring-year hiring-year
                           keys)))
    (incf (slot-value employee 'population))
    employee))
```

Remarques :

- ▶ Un seul point d'entrée (make-instance)
- ▶ Pas de chaîne de constructeurs

AOP / CLOS : Common Lisp Object System – Didier Verna 21/31

Introduction Classes et Objets **Héritage** Polymorphisme

Problèmes Liés à l'Héritage

- ▶ **Persistants :**
 - ▶ Héritage vs. Instantiation (« est un »)
Système à base de classes
 - ▶ Ambivalence de l'héritage (interface / implémentation)
Sous-classement $\iff$ sous-typage
- ▶ **Alternatives :**
 - ▶ Héritage par restriction / Programmation différentielle
Cf. chapitre suivant : aspects dynamiques de CLOS
 - ▶ Héritage multiple / en diamant
 - ▶ Fusion des définitions / packages $\neq$
 - ▶ *Cf. chapitre suivant : combinaisons de méthodes*

AOP / CLOS : Common Lisp Object System – Didier Verna 22/31

Introduction Classes et Objets Héritage **Polymorphisme**

Plan

- Introduction
- Classes et Objets
- Héritage
- Polymorphisme**
 - Rappels
 - Fonctions Génériques et Méthodes
 - Relation (sous-)Classe / (sous-)Type

AOP / CLOS : Common Lisp Object System – Didier Verna 23/31

Introduction Classes et Objets Héritage **Polymorphisme**

Rappel : 3 Formes de Polymorphisme

- ▶ **Polymorphisme statique**
 1. **Surcharge**
 - ▶ Types : aucun sens dans un langage dynamique
 - ▶ Cardinalité : trop ambigu / compliqué
 - ▶ Remplacée par une sémantique d'appel plus riche
 2. **Masquage**
 - ▶ Spécifique aux méthodes membres
 - ▶ Aucun sens avec des fonctions génériques
- ▶ **Polymorphisme dynamique**
 - 3 **Ré-écriture**
 - ▶ Fonctions génériques
 - ▶ Méthodes

AOP / CLOS : Common Lisp Object System – Didier Verna 24/31

Introduction Classes et Objets Héritage Polymorphisme

Fonctions Génériques

La fonction translate

```
(defgeneric translate (object x &optional y))
```

► **Remarques**

- Définition fonctionnelle (defgeneric est une macro)
- Cycle de vie dynamique (méta-objet)
- Typage dynamique
- Interface uniquement (≠ fonctions standard)
- Déclaration optionnelle
- Les accesseurs sont des fonctions génériques
- Comportement identique à celle d'une fonction standard

Même syntaxe d'appel, function-cell, fonctions génériques anonymes etc.

AOP / CLOS : Common Lisp Object System – Didier Verna 25/31

Introduction Classes et Objets Héritage Polymorphisme

Méthodes

Une méthode translate

```
(defmethod translate ((circle circle) x &optional (y 0))
  (translate (center circle) x y))
```

► **Remarques**

- Définition fonctionnelle (defmethod est une macro)
- Cycle de vie dynamique (méta-objet)
- Typage dynamique
- Méthode = une implémentation particulière
- Spécialisation sur les arguments obligatoires
- Multi-méthodes (spécialisation possible sur plusieurs arguments)
- Méthode par défaut = non spécialisée (classe t)
- Méthodes non exécutables et anonymes

AOP / CLOS : Common Lisp Object System – Didier Verna 26/31

Introduction Classes et Objets Héritage Polymorphisme

Chaînage de Méthodes

call-next-method

```
(defgeneric hello (object))

(defmethod hello ((human human))
  (format t "Hello! I'm ~A, ~Am, ~Ayo.~%"
    (name human)
    (size human)
    (age human))
  (values))

(defmethod hello ((employee employee))
  (call-next-method)
  (format t "Working at ~A for ~A euros, started at the age of ~A.~%"
    (company employee)
    (salary employee)
    (hiring-age employee))
  (values))
```

► **Remarque** : liste classée de méthodes applicables

AOP / CLOS : Common Lisp Object System – Didier Verna 27/31

Introduction Classes et Objets Héritage Polymorphisme

Relation (sous-)Classe / (sous-)Type

Rappel

Covariance

Contravariance

AOP / CLOS : Common Lisp Object System – Didier Verna 28/31

Introduction Classes et Objets Héritage Polymorphisme

Covariance / Contravariance

- ▶ **Tout est exprimable** ($\neq$ MOB1)
Exprimable $\Rightarrow$ *compilable*
 - ▶ Contravariance des valeurs retournées
 - ▶ Covariance des arguments
 - ▶ Raisons : langage dynamique / méthodes externes
- ▶ **Tout n'a pas nécessairement de sens** ($=$ MOB1)
 - ▶ Contravariance des valeurs retournées
 - ▶ Échec potentiel
 - ▶ Exemple : (drive DRIVER (vehicle TRUCK-DRIVER))
 - ▶ Covariance des arguments
 - ▶ Erreur potentiellement ignorée
 - ▶ Exemple : (offer TRUCK-DRIVER PARKING)
- ▶ $\Rightarrow$ Vérification dynamique grâce à de nouvelles (multi-)méthodes

AOP / CLOS : Common Lisp Object System – Didier Verna 29/31

Bibliographie

Plan

AOP / CLOS : Common Lisp Object System – Didier Verna 30/31

Bibliographie

- 🌐 Linda G. DeMichiel and Richard P. Gabriel
 The Common Lisp Object System : An Overview
ECOOP, 1987
- 📖 Sonja E. Keene
 Object-Oriented Programming in Common Lisp : a Programmer's Guide to CLOS
Addison-Wesley, 1989
- 📄 Daniel G. Bobrow, Linda G. DeMichiel, Richard P. Gabriel, Sonya E. Keene, Gregor Kiczales and David A. Moon.
 Common Lisp Object System specification
ACM SIGPLAN Notices, 1988.

AOP / CLOS : Common Lisp Object System – Didier Verna 31/31