CLOS-TD3
Composition et délégation

Mireille Blay et Stéphane Ducasse
ESSI3-DESS-DEA

December 19, 1995

Avant toute définition d'une métaclasse, vous ajouterez les lignes suivantes en tête de votre fichier.

```
(defmethod PCL::validate-superclass ((c PCL::class) (new-super PCL::class)) t)
```

1 Création automatique d'objets composites

Soit une classe *composition* qui définit un slot *composants*. On voudrait que la création d'une instance d'une classe sous-classe de composition implique automatiquement la création des composants.

Par exemple, toute voiture est composée de 4 roues et d'un moteur. On voudrait donc que la création d'une instance de voiture implique la création automatique des roues et du moteur.

Que proposez-vous?

Un moteur est également constitué d'un carburateur et de soupapes. La création d'une voiture induit la création d'un moteur qui induit la création du carburateur etc.... Ca marche? (la réponse doit être oui!)

Des indications pour une solution:

- Définissez la classe **meta-composition** qui hérite de **standard-class** ayant un slot **composants-name**. Ce slot contiendra ainsi la liste des symboles représentant les classes des composants d'un objet composites (cf. exemple). Pensez aux possibilités d'initialisations et d'accès à ce slot.
- Définissez la classe **composition** (qui hérite de **standard-object**) ayant un slot **composants**.
- Définissez le comportement nécessaire de la méthode `initialize-instance` pour les instance de la classe **composition**.

```
(defclass moteur (composition)
  ((numero :initform "1000")
   (puissance :initform 50))
   (:metaclass meta-composition)
   (:composant-name carburant soupape soupape))

;; derniere ligne equivalente a
;; (setf (composants (find-class 'moteur)) '(carburateur soupape soupape))
;; une instance de moteur sera composée d'une instance de carburateur, de deux
;; instances de soupapes

(defclass soupape () ((date :initform "origine")))

(defclass carburateur () ((energie :initform "super")))

(defclass roue () ((marque :initform "michelin")))
```

```
(defclass voiture (composition)
  ((marque :accessor marque))
  (:metaclass meta-composition)
  (:composant-name moteur roue roue roue roue ))

;; derniere ligne equivalente a
;; (setf (composants (find-class 'voiture)) '(moteur roue roue roue roue))

(setq cocinelle (make-instance 'voiture))
(describe (car (composants (car (composants cocinelle))))))
```

Eléments de cours

La fonction générique *compute-applicable-methods* détermine l'ensemble des méthodes applicables correspondant à une fonction générique donnée en fonction d'une liste d'arguments requis.
Elle retourne une liste d'objets méthode triée de la plus à la moins spécifique.
compute-applicable-methods((*generic-function standard-generic-function*) arguments)

2 Composition d'objets et délégation

Maintenant nous voulons permettre qu'un message envoyé à un objet (dit composite) composé de plusieurs autres objets puisse être renvoyé à ceux-ci, lorsque l'objet composite ne comprend pas le message en question.
Définir :

- *deleg-composition* qui a un slot *deleg-composants*.
- Définir la méthode *no-applicable-method* de telle sorte qu'un appel à une fonction générique prenant comme argument une instance de la classe *deleg-composition* soit éventuellement redirigé sur les composants de cette instance. La valeur retournée sera alors éventuellement une liste des valeurs retournées par les différents composants.

  Dans cet exercice, vous vous limiterez au cas où l'objet composite est le premier de la liste des arguments de l'appel.
- De même définir la méthode *slot-missing* de telle sorte qu'une tentative d'accès à un des slots d'un objet instance de la classe *deleg-composition* soit éventuellement redirigé sur les composants de cette instance.

```
(defclass oeil () ())
(defmethod regarde ((o oeil) truc) (format t "je le vois...~%"))

(defclass oreille () ())
(defmethod ecoute ((o oreille) truc)  (format t "Je l'entend~%"))

(setq oeil-droit (make-instance 'oeil))
(setq oeil-gauche (make-instance 'oeil))

(setq zozo
    (make-instance 'deleg-composition
                   :deleg-composants (list oeil-droit oeil-gauche)))

(regarde zozo 'kiko)
; -> ("je le vois..." "je le vois...")
(ecoute zozo 'lili)
; -> erreur
```

3 Remarques

Les deux aspects présentés de composition et délégation pourraient être fusionnés. De plus, pour la délégation, vous pourriez essayer d'étendre le mécanisme à tous les objets de la liste des arguments d'appels et pensez que ce mécanisme peut-être récursif. Mais heureusement, nous ne vous le demandons pas.