MATLAB et l'orienté objet

Alain Coron <alain.coron@upmc.fr>
Laboratoire d'Imagerie Paramétrique UMR7623
CNRS - Université Pierre et Marie Curie - Paris

Introduction

MATLAB (contraction de Matrix Laboratory) est un langage de programmation et un environnement de développement édité par Mathworks http://www.mathworks.com.

Avec ce langage de script, il est aisé de

- manipuler des matrices et tableaux
- créer des graphiques.

Avant la version 2008a, une notion de classe existait mais était peu pratique.

Avec la version 2008a, l'orienté objet a été considérablement amélioré.

1er exemple: une classe par valeur

```
%% Mot clé classdef pour définir une classe
classdef classA
  %% Déclaration d'un bloc de 2 propriétés avec
  % leurs attributs optionnels.  Par défaut
  % l'accès est public.  Il peut être protégé
  % (protected) ou privé (private).
  properties (Access=public)
    aleatoire
    v = 5; % attribut avec une valeur par défaut
  end
  %% Déclaration d'un bloc pour les méthodes avec
  % leurs attributs optionnels.
  methods
    %% Constructeur de la classe
    function o = classA(arg1)
      o.aleatoire=rand(1); %rand:tirage aléatoire
      if nargin>=1 % Au - un argument en entrée?
        o.v = arg1;
        o.aleatoire = rand(1);
      end
    end
    %% set.v: appelé à toute affectation de v
    function o = set.v(o, v)
      % disp: fonction pour afficher un message.
      disp('Ici vérifications possibles...')
      o.v = v;
    end
  end
end
```

classA est une classe par valeur. Une fonction recevant en entrée un objet classe par valeur recevra une copie de cet objet. L'original n'est pas modifiable par la fonction.

Le constructeur doit pouvoir être appelé sans argument.

Les classes par valeur n'appellent pas de destructeur.

Exécutons quelques commandes dans MATLAB :

```
>> A = classA(5)  % Création et affichage de l'objet
Ici vérifications possibles...

A =
  classA
  Properties:
    aleatoire: 0.1869
            v: 5
  Methods

>> A.v='c';  %Propriété non typée.  Exécution de A.set.v()
Ici vérifications possibles...
```

2e exemple: une classe par référence et les événements

Une fonction recevant en entrée un objet classe par référence peut modifier directement les propriétés de cet objet. Jusqu'à présent, ce comportement n'était possible que pour les éléments graphiques, les fichiers et les variables globales.

Une classe est par référence si elle hérite de la classe handle en utilisant <. Une classe par référence peut définir un destructeur (delete), et peut utiliser le mécanisme des événements.

```
%% classB hérite de la classe handle et
% a un constructeur par défaut
classdef classB < handle
  properties
    aleatoire = rand(1);
    v=5
  end
  %% Bloc pour déclarer des événements
  events
    evnt_f
  end
  methods
    %% Déclaration d'un destructeur
    function delete(o) % o : l'objet à détruire
      disp('Dans destructeur de classe B.');
    end
    function f(o)
      disp('Avant annonce de l''événement');
      o.notify('evnt_f'); % notify hérité de
                          % handle
      disp('Après annonce de l''événement.');
    end
  end
end
```

```
>> B = classB()
B =
  classB handle
  Properties:
    v: 5
>> % Traitons l'événement par une fonction anonyme
>> % qui affiche un message.
>> h=addlistener(B, 'evnt_f', @(src,
event)disp('Evénement traité par fonction anonyme.'));
>> B.f();
Avant annonce de l'événement
Evénement traité par fonction anonyme.
Après annonce de l'événement.
>> clear B  % Supprime B de l'espace de travail.
            % Mais l'objet existe tjs grâce à h
>> clear h  % Cette fois-ci, le destructeur est appelé.
Dans destructeur de class B.
```

L'appel au constructeur: pas si simple...

```
>> C(1,3) = classA(4) % C est un  tableau de 3 objets
Ici vérifications possibles...
>> for c = C % Itération sur le tableau.
>>   fprintf(1, 'c.aleatoire=%f c.v=%f\n', c.aleatoire, c.v);
>> end
c.aleatoire=0.709365 c.v=5.000000
c.aleatoire=0.709365 c.v=5.000000
c.aleatoire=0.646313 c.v=4.000000
```

3 objets sont créés, mais le constructeur n'a été appelé que 2 fois, une fois sans argument et une fois avec l'argument 4: Le 2e objet est créé en copiant le 1er.

Les principales fonctionnalités

L'orienté objet avec MATLAB comprend:

- un contrôle fin de la visibilité des propriétés et méthodes
- le contrôle possible de l'accès à chaque propriété via les méthodes set.NomPropriété et get.NomPropriété
- la gestion de l'héritage multiple
- la surcharge des opérateurs +, -,:,*, .*,', ...
- l'introduction des références
- l'introduction d'événements
- l'introspection des classes et objets
- l'ajout dynamique de propriétés

Plus simplement, utiliser les classes permet un contrôle sur le nom des propriétés à l'exécution. C'est un moyen de détecter des fautes de frappe.

Depuis la version 2008a, l'analyseur de code a évolué. Certaines lignes acceptées par les premières versions sont maintenant (et à juste titre) refusées.

MATLAB vs C++

Avec MATLAB

- la classe handle cache les pointeurs.
- l'exécution d'une méthode n'est pas basée sur sa signature
- pas de paramètres implicites à une méthode
- pas de programmation générique

Conclusions

L'orienté objet est tardivement devenu une fonctionnalité réellement utilisable dans MATLAB. En ayant en plus recours aux patrons de conception, la conception de traitements et d'interfaces graphiques est considérablement renouvelée.