M1 Informatique - Programmation Orientée Objets

Pr. Hélène Paugam-Moisy

Département d'Informatique et Statistique - ICOM - Université Lumière Lyon 2

Année 2015-2016, semestre 1

UNIVERSITÉ LUMIÈRE LYON 2
UNIVERSITÉ DE LYON

Introduction au cours de POO

DIFFERENTES FAMILLES

de

LANGAGES de PROGRAMMATION

Procédural - Fonctionnel - Orienté objets - Logique

UNIVERSITÉ LUMIÈRE LYON 2
UNIVERSITÉ DE LYON

Programmation impérative

Programmation impérative

La programmation **impérative** ou **procédurale** est la plus courante, celle que l'on apprend généralement en premier (ex. les langages Pascal, Basic ou C).
Composant de base :

les **instructions** et leur organisation (structures de contrôle)

Quelques langages de cette famille :
FORTRAN, COBOL, BASIC, ADA, PASCAL, C
Une variante, les langages de bases de données :
SQL, ORACLE

LUMIERE
LYON 2
UNIVERSITÉ DE LYON

Programmation fonctionnelle

Programmation fonctionnelle

En programmation **fonctionnelle**, un programme est une suite de déclarations de fonctions. Celles-ci se construisent les unes à partir des autres, l'ordre d'exécution devenant ainsi implicite.

Composant de base : les **fonctions** et leur déclaration

Quelques langages de cette famille : LISP, SCHEME

Exemple de calculs, en LISP :

? (+ 5 3)

= 8

? (setq a 4)

= 4

? (* 3 a)

= 12

? (+ (* 2 4) 5 (* 3 2))

= 19

Définition de fonctions, en LISP :

? (de carre(u) (* u u))

= carre

? (carre (+ 2 5))

= 49

? (de cube(x) (* x (carre x)))

= cube

? (cube 3)

= 27

Programmation fonctionnelle

Comparaison : programme en SCHEME et en PASCAL

Cahier des charges : Calculer la somme des *n* premiers entiers non nuls.

Exemple, en PASCAL :

```
program Somme ;
var i, n, s : integer ;
begin
readln( n ) ;
s := 0 ;
for i :=1 to n
do s := s + i ;
writeln( s ) ;
end.
```

```
5
15
```

Exemple, en SCHEME :

```
? (define somme
(lamda ( n )
(if ( = n 0 )
0
( + n (somme( - n 1 ))))))
= fact
```

```
? (somme 5)
= 15
? (somme 0)
= 0
```

Programmation orientée objets

Programmation orientée objets (POO)

La programmation **orientée objets** permet une forte *modularité*, dans le but de faciliter la conception et la maintenance de très grands programmes.

La notion de *type abstrait* (généralisant la notion de type de variable) permet de définir des classes d'objet, en précisant les *attributs* et les *méthodes* qui leur correspondent.
Les classes peuvent hériter les unes des autres, de manière hiérarchique.

Composant de base :

les **classes d'objets**, qui contiennent données et programmes

Quelques langages de cette famille :

SMALLTALK, EIFFEL, PASCAL OBJECT, C++, JAVA

Programmation orientée objets

Exemple de programme en JAVA

Définition de la classe Rectangle

```
class Rectangle {
// attributs
int largeur, hauteur ;
// constructeur
Rectangle(int L, int H) {
largeur = L ;
hauteur = H ; }
//méthodes
int perimetre() {
return 2 * (largeur + hauteur) ; }
int surface() {
return largeur * hauteur ; } }
```

Utilisation de la classe Rectangle

On déclare deux objets de type Rectangle ; on les définit par la donnée de leurs *attributs*, largeur et hauteur ; on fait appel aux *méthodes* dont on a besoin.

```
Rectangle r1, r2 ;
r1 = new Rectangle(5, 4) ;
r2 = new Rectangle(3, 7) ;
perim1 = r1.perimetre() ;
System.out.println(r2.surface()) ;
```

Programmation logique

Programmation logique

La programmation **logique** est propice à l'écriture de *règles d'inférence*, c-à-d d'enchaînements de déductions logiques. Elle est basée sur la logique mathématique (prédicats du 1er ordre).

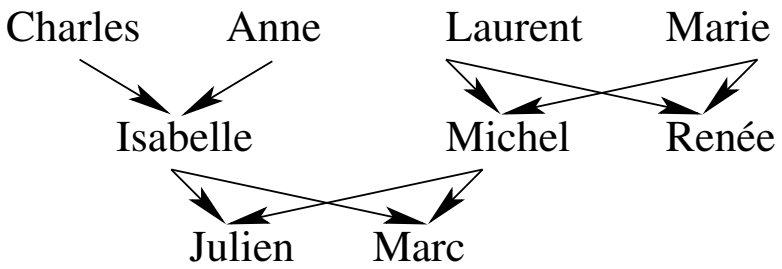
Composant de base : des **règles** du type " si (p ou non(q) ou r) alors w "

Exemple : le langage PROLOG

Ce type de langage peut être utilisé pour l'écriture de *systèmes experts* : à partir d'une base de faits, d'une base de règles et d'un moteur d'inférence, on déduit toutes les réponses valides à une question posée.
Exemples :

- relations de parentés entre individus
- diagnostic médical (symptômes $\rightarrow$ maladies)

Programmation logique



Exemple : **système expert** pour gérer un arbre généalogique

Base de faits :

père(Charles,Isabelle).
père(Laurent,Michel).
. . . etc . . .
mère(Isabelle,Julien).
mère(Isabelle,Marc).

Base de règles :

grand_père(X,Y) → père(X,**S**), père(**S**,Y).
grand_père(X,Y) → père(X,**S**), mère(**S**,Y).
. . . etc . . .
frère_ou_soeur(X,Y) → père(**R**,X), père(**R**,Y).
frère_ou_soeur(X,Y) → mère(**R**,X), mère(**R**,Y).

Interrogation par requêtes en langage PROLOG :

? père (Charles,Michel)
false

? père (Laurent,Y)
Y = Michel
Y = Renée

? grand_père (X,Marc)
X = Laurent
X = Charles