

# Perl orienté objet

Février 2016

*Bérénice Batut*

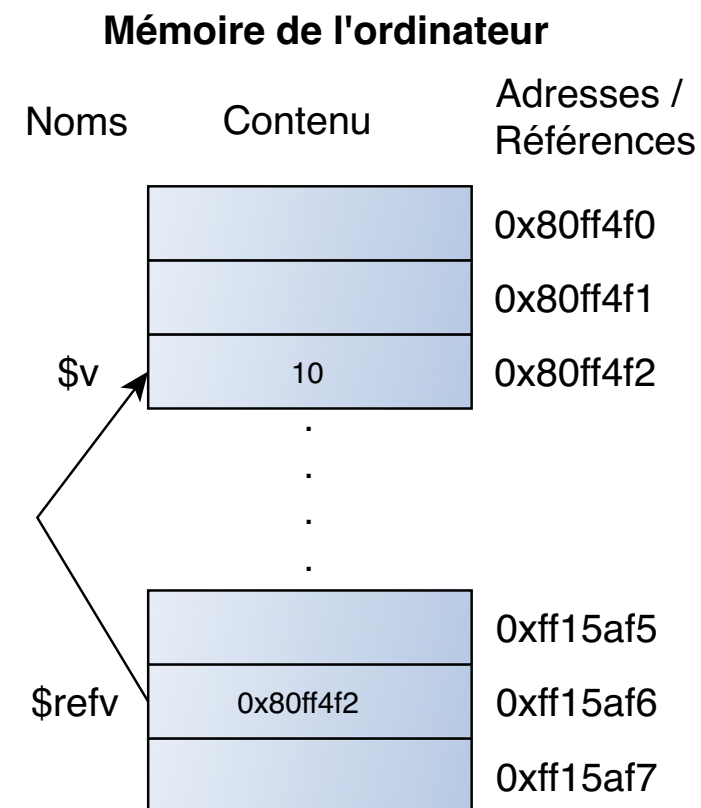
✉ [berenice.batut@udamail.fr](mailto:berenice.batut@udamail.fr)

 PDF

# Références

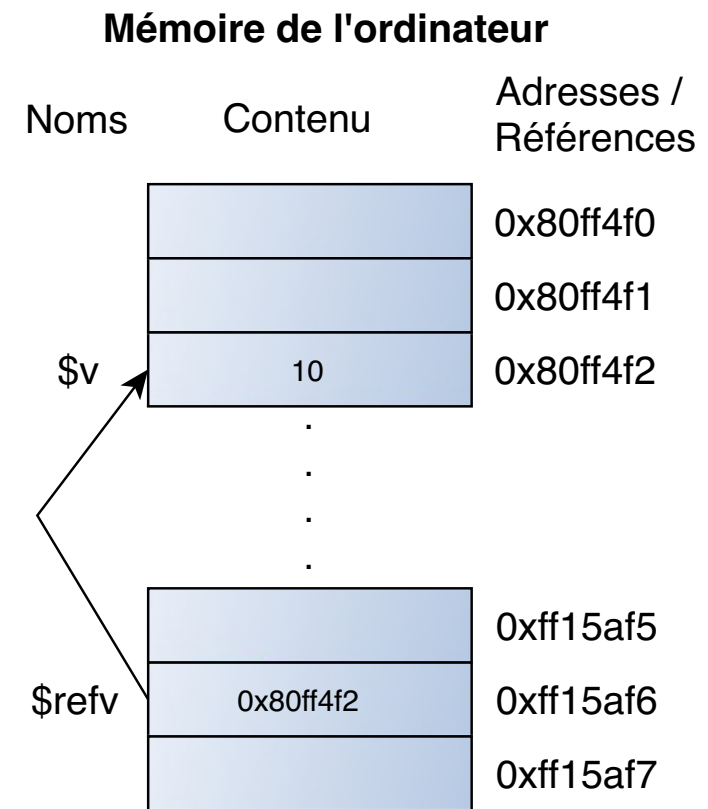
# Références

- Objectif :
  - Construire des structures complexes
- ~ pointeurs en C/C++ et références en Java



# Références sur les scalaires

# Principe



```
my $v = 10;  
my $refv = \ $v;
```

# Principe

```
my $v = 10;  
my $refv = \$v;  
print "$refv\n";
```

```
print "$$refv\n";
```

```
$$refv = 56;  
print "$$refv\n";  
print "$v\n";
```

# Utilisation

## Modification dans une fonction

```
sub f1{  
    my ($ref) = @_;  
    $$ref = $$ref + 10;  
}  
my $v = 20;  
my $refv = \ $v;  
f1( $refv );  
print "$v\n";  
f1 ( \ $v );  
print "$v\n";
```

# Utilisation

## Renvoi d'une variable par une fonction

```
sub f2{  
    my $v = 20;  
    return \$v;  
}  
my $ref = f2();  
print "$$ref\n";
```



# QCM

```
sub f{  
  my ($x,$y) = @_;  
  $$x = 20;  
  $z = $y+10;  
  return (\$z);  
}  
my $v = 10;  
my $w = 20;  
my $x = f(\$v,$w);  
my $y = $$x + $v;
```

Que contient \$y?

☐ 30

☐ 40

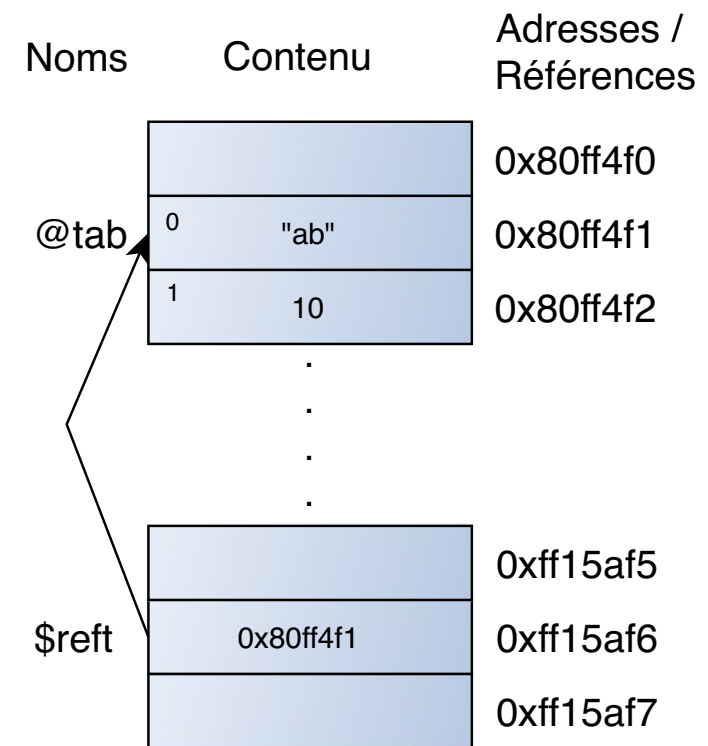
☐ 50

☐ 60

# Références sur les tableaux

# Principe

## Mémoire de l'ordinateur



```
my @tab = ("ab",10);  
my $reft = \@tab;
```

# Principe

```
my @tab = ("ab",10);  
my $reft = \@tab;
```

```
my @tab2 = @$reft;  
print "$tab2[1]\n";  
print "$$reft[1]\n";
```

```
@$reft = ("bc",11);  
print "$tab[0],$tab[1]\n";
```

```
$reft->[1] = 12;  
print "$tab[1]\n";
```

# Principe

Tableau	Référence
tab	\$reft
@tab	@\$reft
\$tab[i]	\$\$reft[i]
\$tab[i]	\$reft->[i]

# Tableaux de tableaux

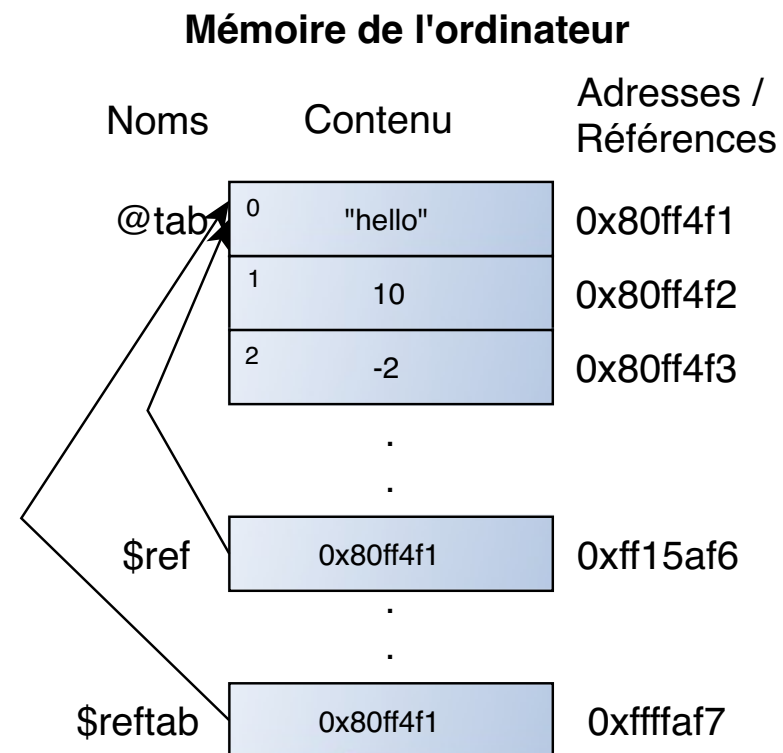
```
my @t1 = ("tutu", 10);  
my @t2 = (20, 30);  
my @t = (2, \@t2, \@t1, ("et", 20));
```

## Mémoire de l'ordinateur

Noms	Contenu	Adresses / Références
@t1	0 "tutu"	0x80ff4f1
	1 10	0x80ff4f2
.		.
@t2	0 20	0xff15af5
	1 30	0xff15af6
.		.
@t	0 2	0xff75121
	1 0xff15af5	0xff75122
	2 0x80ff4f1	0xff75123
	3 "et"	0xff75124
	4 20	0xff75125

# Référence à un tableaux dans une fonction

```
sub f{  
  my ($reftab) = @_;  
  $reftab->[2] = 40;  
}  
my @tab = ( "hello", 10, -2 );  
my $ref = \@tab;  
f( $ref );
```



# QCM

```
my $v = 10;  
my @t1 = (1,2, \ $v);  
my @t2 = (20, \@t1);
```

Comment accéder à \$v depuis @t2?

☐ \$@\$t2[1][2]

☐ \${\$t2[1]->[2]}

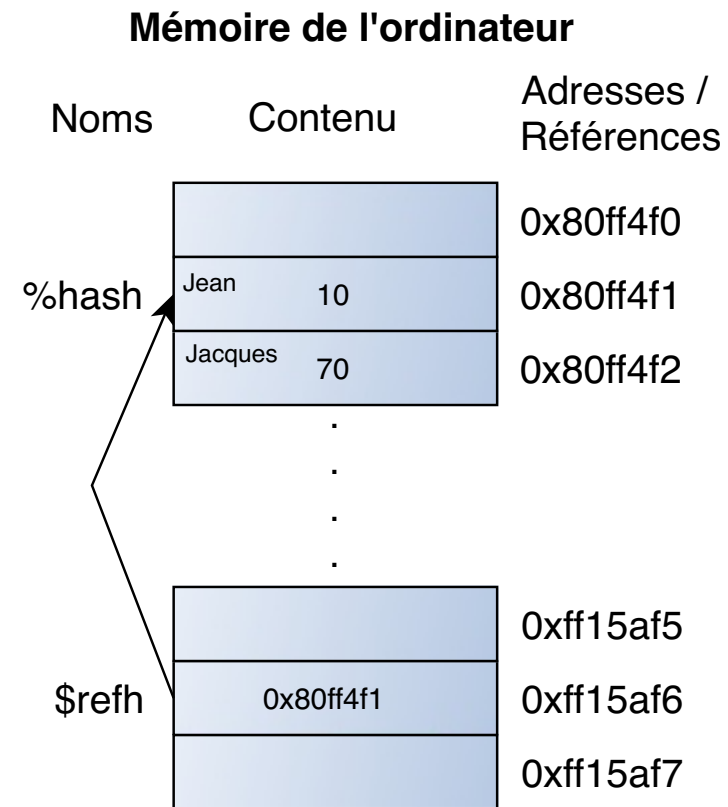
☐ {@{\$t2[1]}[2]}

☐ \$t2[1][2]



# Références sur les tableaux associatifs

# Principe



```
my %hash = ("Jean" => 10, "Jacques" => 70);  
my $refh = \%hash;
```

# Principe

```
my %hash = ('Jean' => 10, 'Jacques' => 70);  
my $refh = \%hash;
```

```
my %hash2 = %$refh;  
print "$hash2{'Jean'}\n";
```

```
print "$$refh{'Jacques'}\n";  
print "$refh->{'Jacques'}\n";
```

# Principe

Tableau associatif	Référence
hash	\$refh
%hash	%%\$refh
\$hash{Paul}	\$\$refh{Paul}
\$hash{Paul}	\$refh->{Paul}

# QCM

```
my $v = 10;  
my %hash1 = ('tata' => \$v, 'titi' => $v, 'toto' => $v);  
my %hash2 = ('toto' => 3, 'tutu' => \%hash1);  
my $refh = \%hash2;  
my $x = $refh->{'tutu'}->{'tata'}+1;
```

Quel est le contenu de \$x?

- ☐ SCALAR(0x7ff615809458)
- ☐ 11
- ☐ 140694899430489
- ☐ Erreur

# Modules en Perl

# Modules

Fichier Perl regroupant un ensemble de variables et de fonctions touchant un à même domaine

~ Bibliothèque, librairie

# Liste des répertoires contenant des modules

```
$ perl -V
Summary of my perl5 (revision 5 version 16 subversion 0) configuration:

...

@INC:
/Users/bbatut/perl5/perlbrew/perls/perl-5.16.0/lib/site_perl/5.16.0/
/Users/bbatut/perl5/perlbrew/perls/perl-5.16.0/lib/site_perl/5.16.0/
/Users/bbatut/perl5/perlbrew/perls/perl-5.16.0/lib/5.16.0/darwin-2le
/Users/bbatut/perl5/perlbrew/perls/perl-5.16.0/lib/5.16.0
.
```



# Utilisation d'un module

```
use NomModule;
```

# Documentation

```
$ perldoc File::Copy
```

NAME

File::Copy - Copy files or filehandles

SYNOPSIS

```
use File::Copy;
```

```
copy("file1","file2") or die "Copy failed: $!";  
copy("Copy.pm",\*STDOUT);  
move("/dev1/fileA","/dev2/fileB");
```

```
use File::Copy "cp";
```

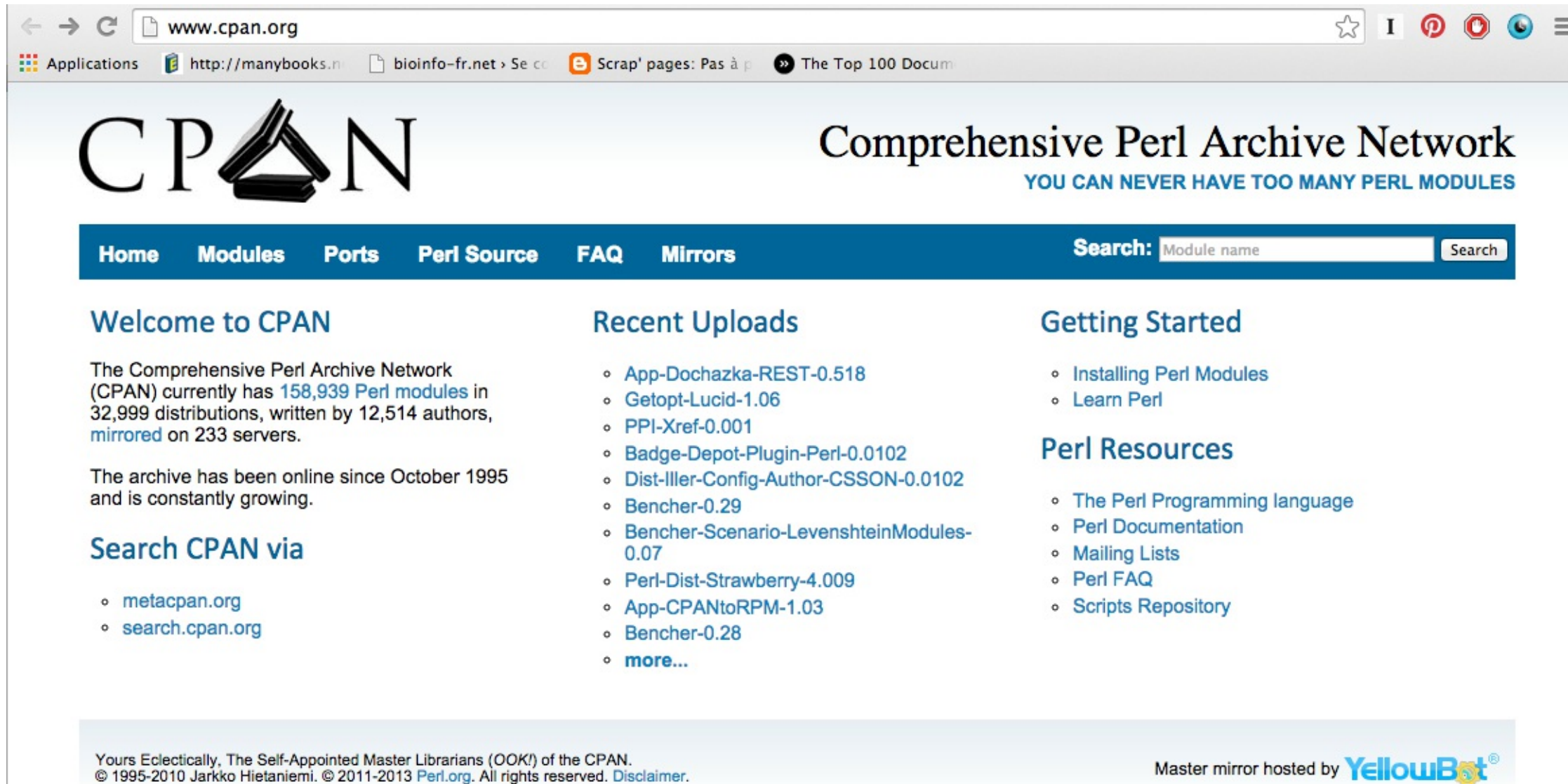
```
$n = FileHandle->new("/a/file","r");  
cp($n,"x");
```

# Utilisation

```
use File::Copy;  
copy("files/file1.txt","files/file2.txt") or die "Copy failed: $!";  
copy("files/file1.txt","\*STDOUT);  
move("files/file2.txt","files/file3.txt");
```

# Où trouver des modules?

## CPAN



The screenshot shows the CPAN (Comprehensive Perl Archive Network) website. The browser's address bar displays 'www.cpan.org'. The website's header features the CPAN logo and the text 'Comprehensive Perl Archive Network' with the tagline 'YOU CAN NEVER HAVE TOO MANY PERL MODULES'. A navigation bar includes links for Home, Modules, Ports, Perl Source, FAQ, and Mirrors, along with a search bar. The main content area is divided into three columns: 'Welcome to CPAN' (providing statistics on modules and distributions), 'Recent Uploads' (listing new module versions), and 'Getting Started' (linking to installation and learning resources). A 'Perl Resources' section is also present. The footer contains copyright information and a note about the master mirror being hosted by YellowBot.

www.cpan.org

Applications http://manybooks.n bioinfo-fr.net > Se cc Scrap' pages: Pas à p The Top 100 Docum

# CPAN

## Comprehensive Perl Archive Network

YOU CAN NEVER HAVE TOO MANY PERL MODULES

Home Modules Ports Perl Source FAQ Mirrors

Search:  Module name

### Welcome to CPAN

The Comprehensive Perl Archive Network (CPAN) currently has [158,939 Perl modules](#) in 32,999 distributions, written by 12,514 authors, [mirrored](#) on 233 servers.

The archive has been online since October 1995 and is constantly growing.

### Search CPAN via

- [metacpan.org](#)
- [search.cpan.org](#)

### Recent Uploads

- [App-Dochazka-REST-0.518](#)
- [Getopt-Lucid-1.06](#)
- [PPI-Xref-0.001](#)
- [Badge-Depot-Plugin-Perl-0.0102](#)
- [Dist-iller-Config-Author-CSSON-0.0102](#)
- [Bench-0.29](#)
- [Bench-Scenario-LevenshteinModules-0.07](#)
- [Perl-Dist-Strawberry-4.009](#)
- [App-CPANtoRPM-1.03](#)
- [Bench-0.28](#)
- [more...](#)

### Getting Started

- [Installing Perl Modules](#)
- [Learn Perl](#)

### Perl Resources

- [The Perl Programming language](#)
- [Perl Documentation](#)
- [Mailing Lists](#)
- [Perl FAQ](#)
- [Scripts Repository](#)

Yours Eclectically, The Self-Appointed Master Librarians (OOK!) of the CPAN.  
© 1995-2010 Jarkko Hietaniemi. © 2011-2013 [Perl.org](#). All rights reserved. [Disclaimer](#).

Master mirror hosted by [YellowBot](#)

# Écriture d'un module

# Principe

1 fichier

- Indépendant des scripts qui l'utilise
- Extension : .pm
- Dans un des répertoires de la variable @INC

# Structure du fichier

```
# --- MonModule.pm ---  
  
package MonModule;  
  
use strict;  
use warnings;  
  
sub bonjour {  
    my ($prenom) = @_;  
    print "Bonjour $prenom\n";  
}  
  
1;
```

# Utilisation du module

```
# --- script.pl ---  
#!/usr/bin/perl  
  
use strict;  
use warnings;  
  
use MonModule;  
  
MonModule::bonjour( "Paul" );
```



# Variables dans un module

- Accessibles seulement aux fonctions dans le module : `my`
- Accessibles depuis l'extérieur du module : `our`

# Variables dans un module

```
# --- MonModule.pm ---
package MonModule;
use strict;
use warnings;

our $x = 'toto';
my $y = 'toto';
sub bonjour {
    # Variable locale
    my ($prenom) = @_;
    print "$x $y\n";
}

1;
```

```
# --- script.pl ---
#!/usr/bin/perl
use strict;
use warnings;
use Utils;
Utils::bonjour("Paul");
print "$Utils::x\n";
print "$Utils::y\n";
# Erreur
```

# Dernière ligne d'un module

1;

Valeur de chargement du module

# Documentation

```
# --- MonModule.pm ---  
  
=head1 NAME  
  
MonModule.pm - Useful functions  
  
=head1 SYNOPSIS  
  
    use MonModule;  
    bonjour("Paul");  
  
=head1 DESCRIPTION  
  
Blabla blabla  
  
=cut
```

# Documentation

```
# --- MonModule.pm (suite) ---  
package MonModule;  
use strict;  
use warnings;  
  
=head1 FUNCTION bonjour  
  
    This function prints hello in french  
  
=cut  
  
sub bonjour {  
    my ($prenom) = @_;  
    print "Bonjour $prenom\n";  
}  
  
1;
```

# Documentation

```
$ perldoc MonModule.pm
```

NAME

MonModule.pm - Useful functions

SYNOPSIS

```
use MonModule;  
bonjour("Paul");
```

DESCRIPTION

Blabla blabla

FUNCTION `bonjour`

This function prints hello in french

# QCM

```
package UnModule;  
use strict;  
use warnings;  
my @t = (13,24);  
our $reft = \@t;
```

Comment accéder au premier élément du tableau t depuis un script invoquant le module?

- ☐ @{\$UnModule::reft}[0]
- ☐ Impossible
- ☐ @UnModule::t[0]
- ☐ @UnModule::reft[0]

# Programmation orientée objet



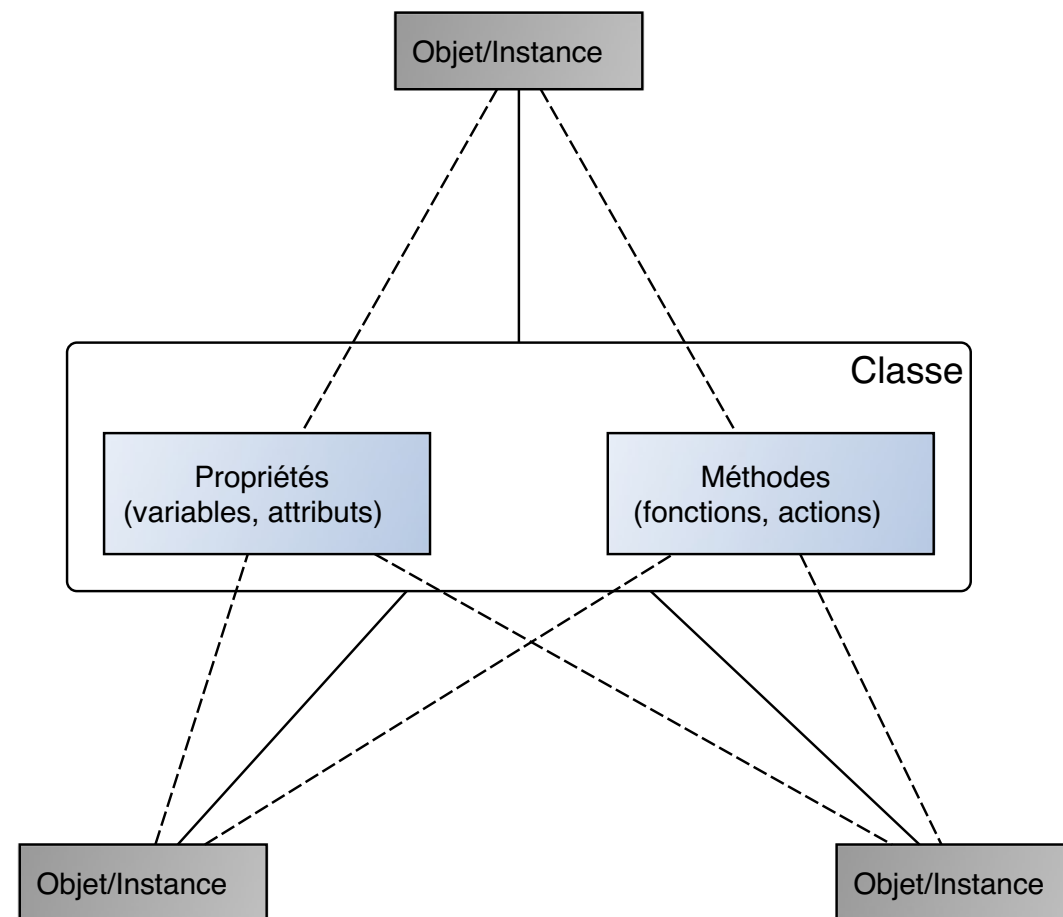
Programmation procédurale

**Quelles sont les fonctions/actions à faire ?**

Programmation orientée objet

**Quelles sont les données du problème ?**

# Principe



# Exemple : Employe

- Classe
  - Propriétés
    - Nom
    - Date de naissance
    - Salaire
  - Méthodes
    - Récupération des informations
    - Augmentation de salaire
- Embauche d'un nouvel employé
  - Création d'un nouvel objet
  - Remplissage des différentes propriétés

# Principe

- Classe = Module
- Objet/Instance = Référence associée à la classe

# Construction d'une classe et des objets

# Création d'un fichier `EmpLoye.pm`

```
# --- fichier Employe.pm ---  
package Employe;  
use strict;  
  
...  
1;
```

# Ecriture du constructeur

- Création d'une référence vers un tableau associatif
- Association de la référence au package (*bless*)
- Remplissage du tableau associatif avec les propriétés

# Constructeur de la classe Employe

```
# --- Employe.pm ---  
...  
  
# Constructeur  
sub new {  
    my ($class, $nom, $salaire) = @_;  
    my $this = {};  
    bless ($this, $class);  
    $this->{NOM} = $nom;  
    $this->{SALAIRE} = $salaire;  
    return $this;  
}  
  
...
```



# Création d'objets

```
# --- script.pl ---  
#!/usr/bin/perl  
  
use strict;  
use warnings;  
  
use Employee;  
  
my $e1 = Employee->new("Jean Dupont", 2000);  
my $e2 = Employee->new("Robert Duval", 1500);
```

# Manipulation

# Connaissance de la classe

```
my $e1 = Employee->new("Jean Dupont", 2000);  
print "$e1\n";
```

```
Employee=HASH(0x7fc802b5e698)
```

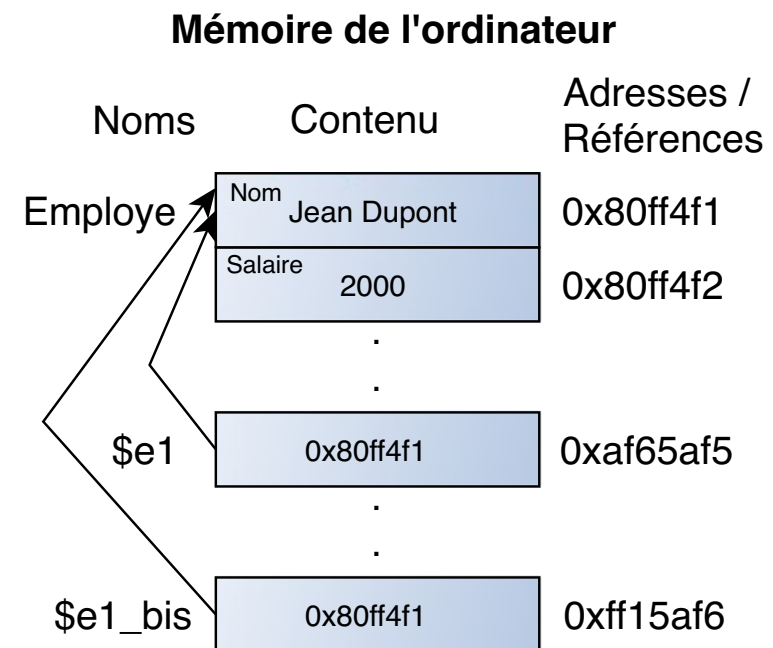
# Visualisation d'un objet

```
use Data::Dumper;
print Dumper($e1)."\n";

$VAR1 = bless( {
    'SALAIRE' => 2000,
    'NOM' => 'Jean Dupont'
}, 'Employe' );
```

# "Copie" d'un objet

```
my $e1_bis = $e1;  
print "$e1_bis\n"
```



# Méthodes

# Ecriture

```
# --- Employe.pm ---  
...  
  
sub recuperation_info{  
    my ($this) = @_;  
    return $this->{NOM}." ".$this->{SALAIRE};  
}  
  
sub augmentation_salaire {  
    my ($this, $pourcentage) = @_;  
    my $p = (100+$pourcentage)/100;  
    $this->{SALAIRE} = $this->{SALAIRE}*$p;  
}
```

# Utilisation

```
# --- script.pl ---  
  
#!/usr/bin/perl  
  
use strict;  
use warnings;  
  
use Employe;  
  
my $e1 = Employe->new("Jean Dupont", 2000);  
print "$e1->recuperation_info(2)\n";  
e1->augmentation_salaire(2);  
print "$e1->recuperation_info(2)\n";
```



# Question

```
# --- fichier Vehicule.pm ---  
use warnings;  
  
sub new{  
    my ($class, $nbRoues, $couleur) = @_;  
    my $this = {};  
    $this->{NB_ROUES} = $nbRoues;  
    return $this;  
}
```

Que manque-t-il pour bien déclarer la classe `Vehicule` avec deux propriétés (nombre de roues et couleur)?

# Références

- [Guide Perl](#)

# BioPerl