Perl Orienté Objet
BioPerl

"There is more than one way to do it"

Bérénice Batut,

berenice.batut@udamail.fr

DUT Génie Biologique

Option Bioinformatique

Année 2014-2015

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

- Types de données
- Structures de contrôle
- Manipulation de fichiers
- Expressions régulières
- Définitions de fonctions

Références

Modules

Programmation Orientée Objet

BioPerl

Perl Orienté Objet - BioPerl

- Rappels de Perl — http://formation-perl.fr/guide-perl.pdf
  - Types de données
  - Structures de contrôle
  - Manipulation de fichiers
  - Expressions régulières
  - Définitions de fonctions
- Références
- Modules
- Programmation Orientée Objet
- BioPerl

Scalaires

Définition des variables `$chaine = ”une chaine\n”`

Nombres (entiers ou réels)

- Opérateurs `+ - * / ** %`
- Comparaison `< > <= >= == !=`

Chaînes de caractères

- Simple ou double quote selon interprétation ou non
- Opérateurs `. (concaténation)`
- Comparaison `lt gt le ge eq ne`
- Fonctions de manipulations

```
chomp($chaine); # retire le caractère de fin de ligne
split(/ /, $chaine);# découpe la chaine
length($chaine);# retourne la longueur de la chaine
substr($chaine, 1, 3);# retourne la sous-chaine
index($chaine,” ”, 1);# retourne l’indice de la 1e
occurrence
rindex($chaine,” ”);# retourne l’indice de la dernière
occurrence
```

Tableaux

Définition des variables `@tab = (3, ”chaine”);`

Accès aux données

```
print $tab[1];
$tab[0] = 10; $tab[-1] = 2;
print scalar(@tab);# affiche la taille du tableau
```

Fonctions de manipulation

```
push(@tab, 20);# ajoute en fin
pop(@tab);# supprime en fin
unshift(@tab, 0);# ajoute en tête
shift(@tab);# supprime en tête
join(” ”,@tab);# concatène les éléments
```

Tableaux associatifs

Définition des variables

```
%hash = (”id1”=>1, “id2”=>“chaine”,“id3”=>(1,2));
```

Accès aux données

```
$hash{‘id4’}=250;
```

Fonction de manipulation

```
keys(%hash);# renvoie les clés
values(%hash);# renvoie les valeurs
each(%hash);# renvoie les couples (clé, valeur)
delete(%hash,”id1”);# supprime la valeur liée à
   la clé
exists($hash{cle});# test si clé existe dans la
   table
```

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

- Types de données
- Structures de contrôle
- Manipulation de fichiers
- Expressions régulières
- Définitions de fonctions

Références

Modules

Programmation Orientée Objet

BioPerl

Tests

```
if (expression booléenne) {
   instructions;
}elsif (exppression booléenne) {
   instructions;
}else{
   instructions;
}

unless (expression booléenne){
   instructions;
}else{
   instructions;
}
```

Boucles

```
while (condition){
    instructions;
}

until (condition){
    instructions;
}

for(initialisation; condition; incrément){
    instructions;
}

foreach variable (liste) {
    instructions;
}

next # fin d'exécution du bloc d'instructions
last # fin de la boucle
redo # redémarrage du bloc d'instructions
```

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

- Types de données
- Structures de contrôle
- Manipulation de fichiers
- Expressions régulières
- Définitions de fonctions

Références

Modules

Programmation Orientée Objet

BioPerl

Fichiers

Ouverture

```
open(FILE, ”nom du fichier”) or die(“open: $!”);# ouvre en lecture ou met fin du programme
open(FILE, “>nom du fichier”);# ouvre en écriture
```

Lecture

```
while($ligne = <FILE>){
    instructions;
}
```

Écriture

```
print FILE ”chaine à écrire\n”;
```

Fermeture

```
close(FILE);
```

Opérateurs sur les noms de fichiers/dossiers

```
if(-e chemin_fichier) # teste si chemin_fichier est un chemin valable dans le système de fichier
if(-f chemin_fichier) # teste si c'est un fichier normal
if(-d chemin) # teste si c'est un répertoire
if(-r chemin) # teste s'il y a les droits de lecture
if(-w chemin) # teste s'il y a les droits d'écriture
if(-x chemin) # teste s'il y a les droits d'exécution
if(-z chemin) # teste si le fichier est vide
if(-s chemin) # teste si le fichier est non vide et renvoie sa taille
```

Perl Orienté Objet - BioPerl

- Rappels de Perl — http://formation-perl.fr/guide-perl.pdf
  - Types de données
  - Structures de contrôle
  - Manipulation de fichiers
  - Expressions régulières
  - Définitions de fonctions
- Références
- Modules
- Programmation Orientée Objet
- BioPerl

Expressions régulières

Vérification de la présence d'un motif `m/motif/`

Substitution d'un motif `s/motif/chaine_rempl/`

Liaison entre une variable et une expression `=~`

```
$v =~ m/motif/ # vérifie si la variable contient
   le motif
$v =~ s/motif/rempl # remplace la première
   occurrence du motif par le remplaçant
```

Ensembles

```
.           # caractère quel qu'il soit (sauf \n)
[caractères]# un caractère parmi ceux entre crochets
[c1-c2]     # intervalles de caractères entre c1 et c2
[^ensemble] # complémentaire de l'ensemble
```

Exemples

```
[qjk]    # soit q, soit j, soit k
[^qjk]   # ni q, ni j, ni k
[a-z]    # tout caractère compris entre a et z
[^a-z]   # aucun caractère compris entre a et z
[a-zA-Z] # tous les caractères Alpha, sans accents
[a-z]+   # toute chaîne de a-z non vide
```

Quantificateurs

Application au motif atomique précédent

Spécification du nombre de fois où le motif peut/doit être présent

```
+      # une occurrence ou plus de l'expression
*      # zéro ou plusieurs occurrences
?      # zéro ou une occurrences
{n}    # n fois exactement
{n,}   # au moins n fois
{,n}   # au plus n fois
{n,m}  # entre n et m fois
```

Raccourcis pour des ensembles courants

```
\d    #tout chiffre, équivalent à [0-9]
\D    #aucun chiffre, équivalent à [^0-9]
\w    #tout caractère alphanumérique, équivalent
         à [0-9a-zA-Z_]
\W    #aucun caractère alphanumérique, équivalent
         à [^0-9a-zA-Z_]
\s    #tout espacement, équivalent à [ \n\t\r\f]
\S    #aucun séparateur, équivalent à [^ \n\t\r\
```

Divers

Caractères spéciaux : besoin de despécifier avec \

```
\ | ( ) [ ] { } ^ $ * + ?
```

Alternatives

```
option1|option2   # choix entre option1 et
                  option2
```

Assertions : position dans l'expression

```
^  # début de la chaine
$  # fin de la chaine
```

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

- Types de données
- Structures de contrôle
- Manipulation de fichiers
- Expressions régulières
- Définitions de fonctions

Références

Modules

Programmation Orientée Objet

BioPerl

Fonctions

Définition

```
sub nom_fonction{
   instructions;
}
```

Arguments d'une fonction contenues dans la variable `@_`

Valeur de retour d'une fonction précédée de l'instruction `return`

Appel de fonction

```
$resultat = nom_fonction(arguments)
```

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

- Références sur scalaire
- Références sur tableau
- Références sur tableau d'association
- Passage de référence à une fonction

Modules

Programmation Orientée Objet

BioPerl

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

- Références sur scalaire
- Références sur tableau
- Références sur tableau d'association
- Passage de référence à une fonction

Modules

Programmation Orientée Objet

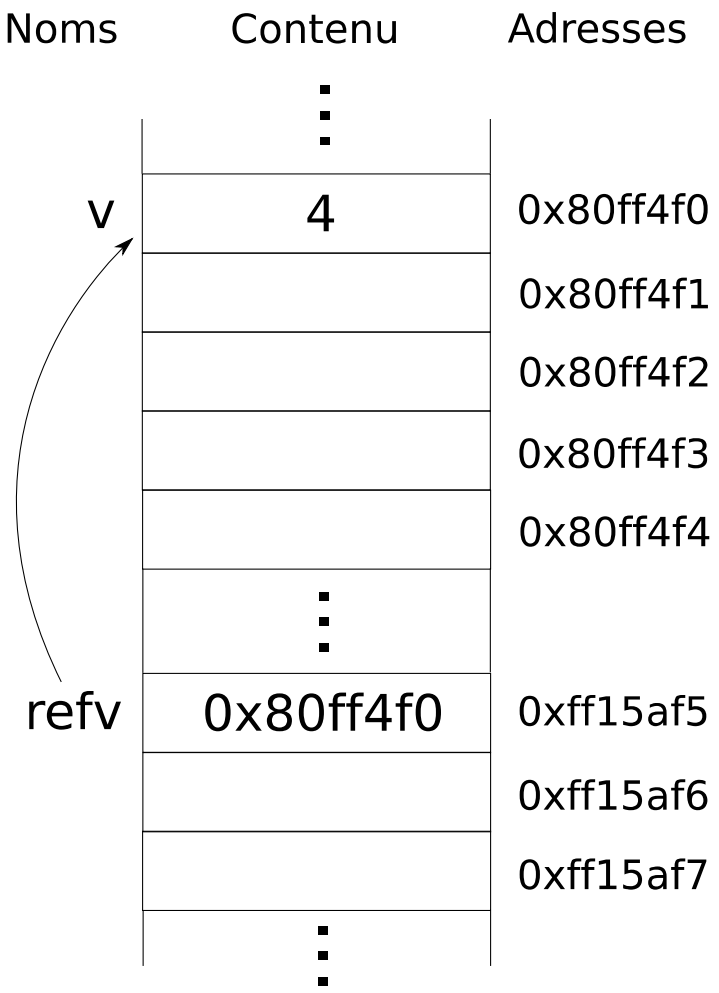
BioPerl

Références sur les scalaires

Programme

```
my $v = 4;
my $refv = \$v;

print "$refv\n";
  #affiche SCALAR(0x80ff4f0)

print "$$refv\n";
  #affiche 4

$$refv = 9;

print "$$refv\n";
  # affiche 9
print "$v\n";
  #affiche 9
```

Mémoire de l'ordinateur



Utilisation des références vers les scalaires

Modification dans une fonction

```
sub fonction{
    my ($ref) = @_;
    $$ref = 0;
}
fonction($refv); # ou fonction(\$v)
```

Génération de référence sur scalaire

```
sub fonction2{
    my $w = 43;
    return \$w;
}
my $reff = fonction2(); # référence vers une
    variable scalaire valant 43
```

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

- Références sur scalaire
- Références sur tableau
- Références sur tableau d'association
- Passage de référence à une fonction

Modules

Programmation Orientée Objet

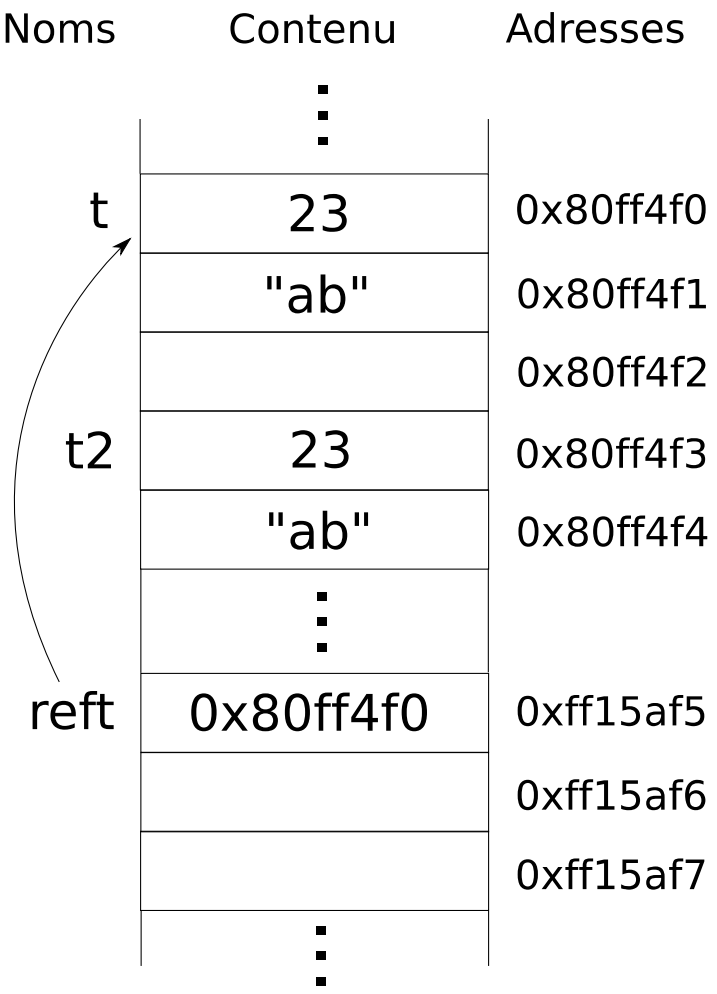
BioPerl

Références sur les tableaux

Programme

```
my @t = (23,"ab");
my $reft = \@t;

my @t2 = @$reft;

print "$$reft[1]\n";
  #affiche "ab"

print "$reft->[1]\n";
  # affiche "ab"
```

Mémoire de l'ordinateur



Equivalence de notations

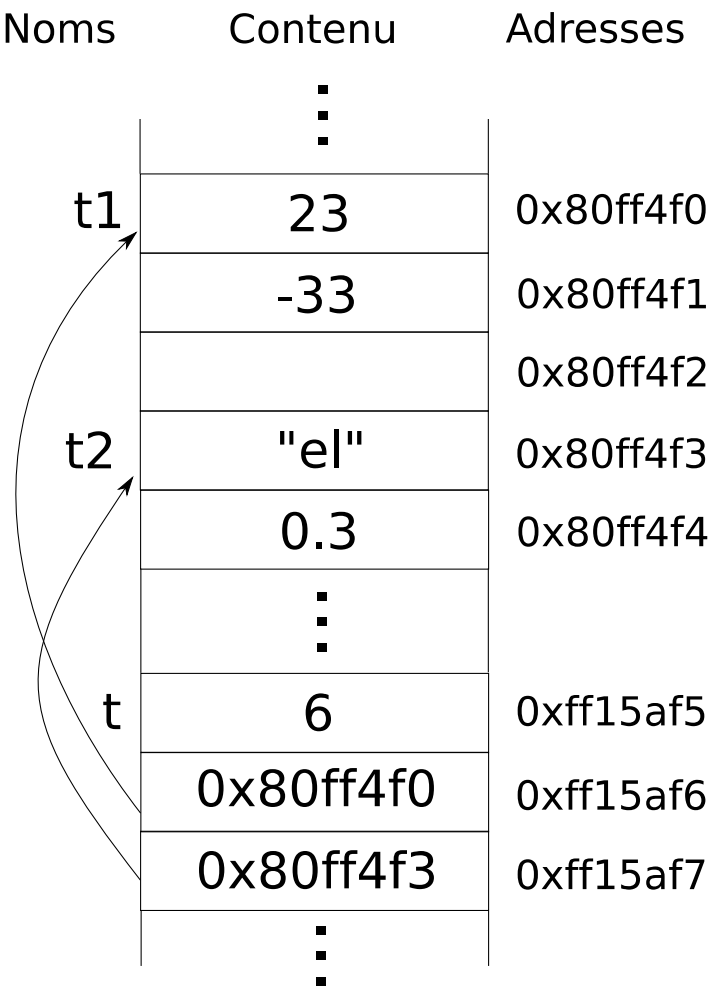
Tableau	Référence
`\@t`	`$reft`
`@t`	`@$reft`
`$t[i]`	`$$reft[i]`
	`$reft->[i]`

Tableaux de tableaux

Programme

```
my @t1 = (23,-33);
my @t2 = ("el",0.3);
my @t = (6,\@t1,\@t2);
```

Mémoire de l'ordinateur



Perl Orienté Objet - BioPerl

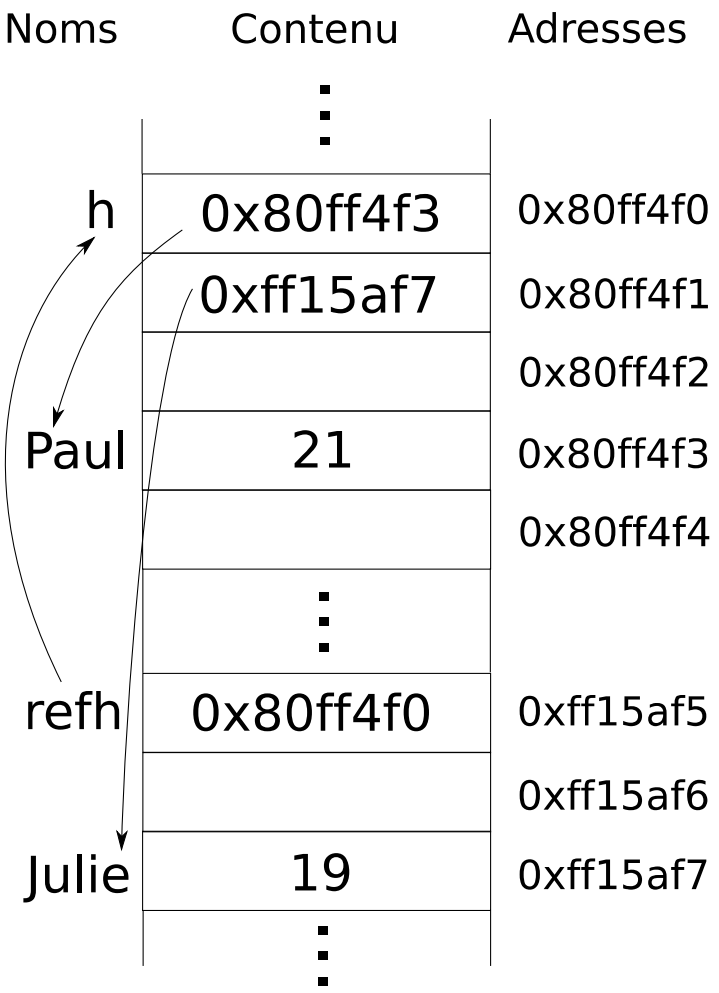
- Rappels de Perl — http://formation-perl.fr/guide-perl.pdf
- Références
  - Références sur scalaire
  - Références sur tableau
  - Références sur tableau d'association
  - Passage de référence à une fonction
- Modules
- Programmation Orientée Objet
- BioPerl

Références sur les tableaux d'association

Programme

```
my %h = ('Paul' => 21,
         'Julie' => 19);
my $refh = \%h;

my %h2 = %$refh;
```

Mémoire de l'ordinateur



Perl Orienté Objet - BioPerl

- Rappels de Perl — http://formation-perl.fr/guide-perl.pdf
- Références
  - Références sur scalaire
  - Références sur tableau
  - Références sur tableau d'association
  - Passage de référence à une fonction
- Modules
- Programmation Orientée Objet
- BioPerl

Equivalence de notations

Tableau	Référence
`\%h`	`$refh`
`%h`	`%$refh`
`$h{Paul}`	`$$refh{Paul}`
	`$refh->{Paul}`

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

- Références sur scalaire
- Références sur tableau
- Références sur tableau d'association
- Passage de référence à une fonction

Modules

Programmation Orientée Objet

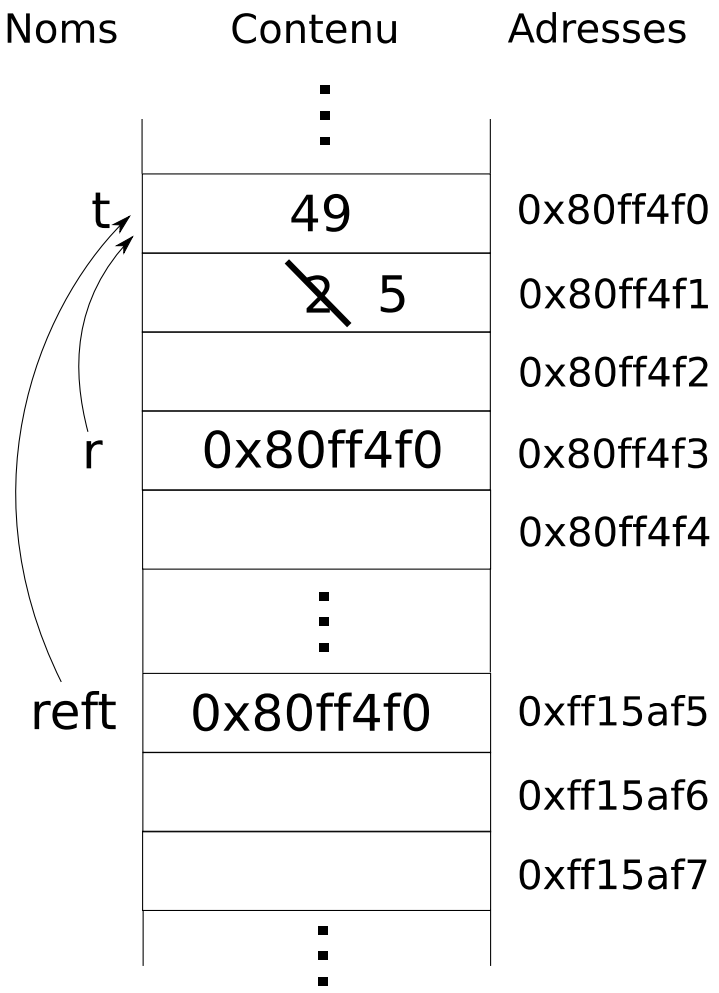
BioPerl

Passage de référence à une fonction

Programme

```
sub fonction{
  my ($reft) = @_;
  $reft->[1] = 5;
}

my @t = (49,2);
my $r = \@t;
fonction($r);
```

Mémoire de l'ordinateur



Perl Orienté Objet - BioPerl

Rappels de Perl

Références

Modules

Programmation Orientée Objet

BioPerl

http://formation-perl.fr/guide-perl.pdf

Module

Fichier Perl regroupant un ensemble de variables et de fonctions

Nommé `Nom_du_module.pm`

Structure

```
# --fichier Nom_du_module.pm
package Nom_du_module;
sub fonction1 {…
}
our $variable;
1;
```

Utilisation

```
use Nom_du_module;
Nom_du_module::fonction1(Nom_du_module::$variable);
```

Perl Orienté Objet - BioPerl

Rappels de Perl

Références

Modules

Programmation Orientée Objet

BioPerl

http://formation-perl.fr/guide-perl.pdf

Programmation orientée objet

Question de bases

- Quelles sont les données du problème?

Notion d'objets auxquelles sont affectés

- Variables/attributs (propriétés)
- Fonctions/actions (méthodes)

Paradigme de programmation qui consiste à

- Encapsuler les données dans une structure appelée **classe**
- Associer des **méthodes** (sous-programmes) de traitement des **attributs** de la classe

Exemple de classe

Classe « Employe »

- Propriétés
  - Nom
  - Salaire, …
- Méthodes
  - Récupération des informations
  - Augmentation salaire, …

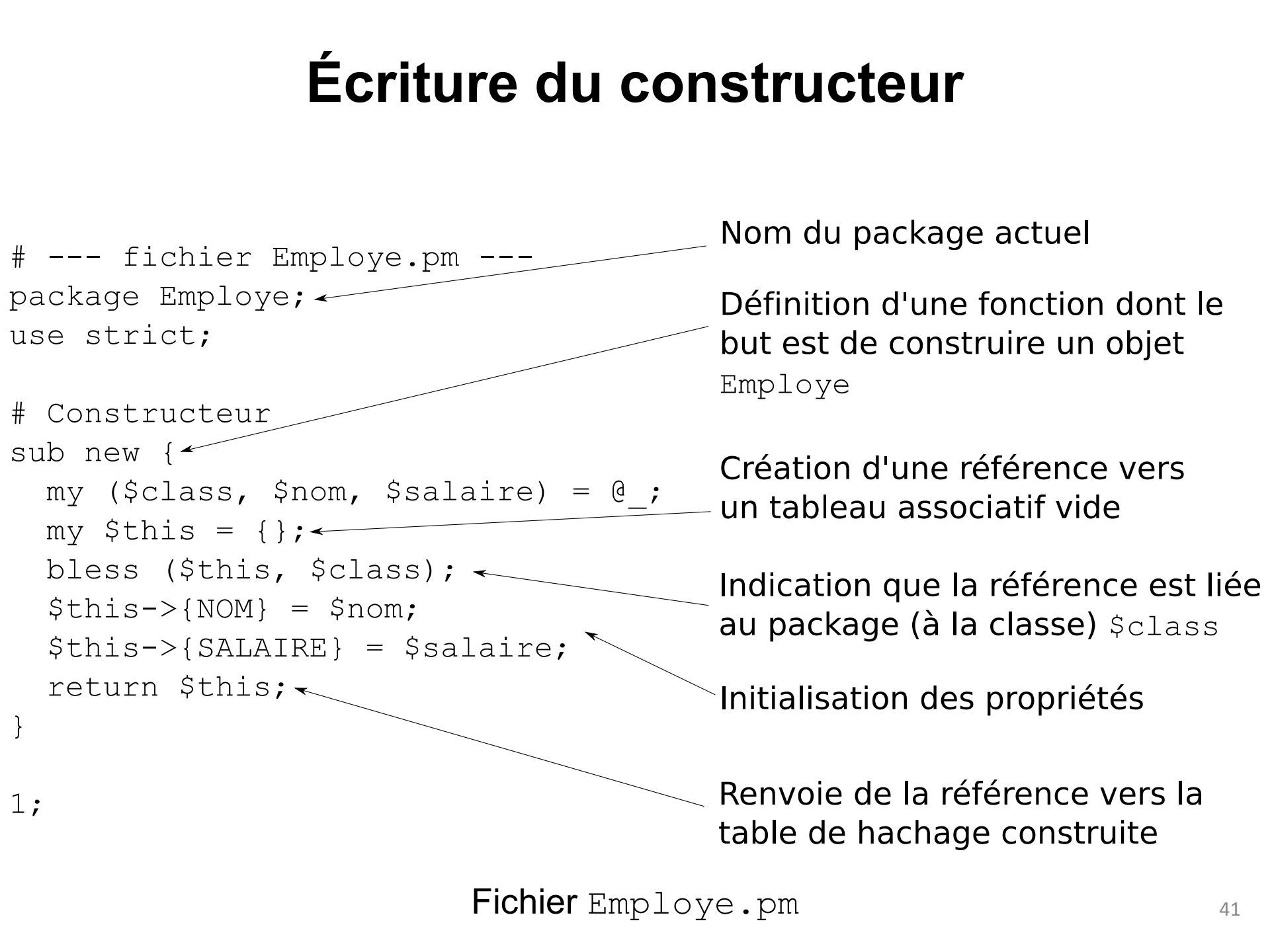
Embauche d'un nouvel employé : création d'une nouvelle instance (ou nouvel objet) de la classe en renseignant les différentes propriétés

Création d'une classe en Perl

- Choix du nom de la classe
  - Définition du module correspondant
- Définition des attributs de la classe
- Définition des constructeurs
  - Création d'une référence vers un tableau associatif
  - Stockage des attributs de l'objet dans le tableau associatif
    - Clé : nom de l'attribut
    - Valeur : valeur de l'attribut
  - Association de cette référence au module avec `bless`
- Définition des autres méthodes de la classe
  - Interface entre les objets et le programme Perl de l'utilisateur

Écriture du constructeur

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

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

1;
```



Fichier `Employe.pm`

Appel du constructeur

```
#! /usr/bin/perl -w
use strict;

use Employe;

my $e1=Employe->new("Jean Dupont", 2000);
my $e2=Employe->new("Robert Duval",1500);
```

Appel du module correspondant à la classe d'intérêt

Appel au constructeur

Création de deux instances indépendantes de la classe `Employe`

Référence vers un tableau d'association dont les champs ont été initialisés et qui a été "bénie" en `Employe`

Fichier `script.pl`

Écriture de méthodes

```
# Augmentation de salaire
sub augmentation_salaire {
  my ($this, $pourcentage) = @_;
  my $p = (100+$pourcentage)/100;
  $this->{SALAIRE} = $this->{SALAIRE}*$p;
}

# Récupération des informations sous forme de chaine
sub recuperation_info{
  my ($this) = @_;
  return $this->{NOM}." ".$this->{SALAIRE};
}
```

Fichier `Employe.pm`

```
print $e1->recuperation_info(), "\n";
$e1->augmentation_salaire(2.0);
print $e1->recuperation_info(), "\n";
```

Fichier `script.pl`

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

Modules

Programmation Orientée Objet

BioPerl

- Bio::Seq
- Bio::SeqIO
- Bio::DB
- Bio::SearchIO

BioPerl

Ensemble de modules Perl dédiés à la bioinformatique qui permettent de

- Lire, écrire, traduire, manipuler des séquences
- Accéder à des bases de données
- Rechercher des séquences
- Rechercher des gènes
- Manipuler des alignements
- Lire des structure 3D
- …

Documentation http://doc.bioperl.org/bioperl-live/

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

Modules

Programmation Orientée Objet

BioPerl

- Bio::Seq
- Bio::SeqIO
- Bio::DB
- Bio::SearchIO

Bio::Seq

Classe permettant de représenter une séquence de nucléotides ou d'acides aminés

```
use Bio::Seq;

my $sequence_objet = Bio::Seq->new(
    -seq => "ACTGTGTGTCC",
    -id => "Chlorella sorokiniana",
    -accession_number => "CAA41635"
);
```

Méthodes de Bio::Seq

```
$sequence_objet->seq();
$sequence_objet->subseq(5,10);
$sequence_objet->accession_number();#identifiant
$sequence_objet->alphabet(); #dna, rna ou protein
$sequence_objet->seq_version();
$sequence_objet->keywords();
$sequence_objet->length();
$sequence_objet->desc();#description
$sequence_objet->primary_id();#identifiant unique
$sequence_objet->display_id();#identifiant
$sequence_objet->revcom;#complement
$sequence_objet->translate;#traduction
$sequence_objet->get_SeqFeatures();#caractéristiques
```

Plusieurs types de séquences

`Bio::Seq::PrimarySeq`

Version simplifiée de `Bio::Seq`

`Bio::Seq::LocatableSeq`

`Bio::Seq::RelSegment`

`Bio::Seq::Quality`

`Bio::Seq::PrimaryQual`

Version simplifiée de `Bio::Seq::Quality`

…

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

Modules

Programmation Orientée Objet

BioPerl

- Bio::Seq
- Bio::SeqIO
- Bio::DB
- Bio::SearchIO

Bio::SeqIO

Classe permettant de lire (ou écrire) une séquence depuis (ou vers) un fichier

Plusieurs formats pris en compte

- clustalw
- emboss
- fasta
- mega
- nexus
- quality
- …

Exemple de lecture/écriture

```
use Bio::Seq;
use Bio::SeqIO;

my $input = Bio::SeqIO->new(
   -file => "glutamate.fasta",
   -format => "fasta");

my $output = Bio::SeqIO->new(
   -file => ">glutamate.gcg",
   -format => "gcg");

while($seq = $input->next_seq()){
   $output->write_seq($seq);
}
```

Ouverture du fichier fasta en lecture

Ouverture du fichier gcg en écriture

Parcours du fichier fasta

Écriture dans le fichier gcg

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

Modules

Programmation Orientée Objet

BioPerl

- Bio::Seq
- Bio::SeqIO
- Bio::DB
- Bio::SearchIO

Bio::DB

Classe permettant d'accéder aux bases de données

GenBank	`Bio::DB::GenBank`
GenPept	`Bio::DB::GenPept`
SwissProt	`Bio::DB::SwissProt`
RefSeq	`Bio::DB::RefSeq`
EMBL	`Bio::DB::EMBL`

Exemples d'accès à GenBank

```
use Bio::Seq;
use Bio::DB::GenBank;
use Bio::DB::Query::GenBank;s

my $genbank=new Bio::DB::GenBank;

# Premier exemple de requete
my $seq=$genbank->get_Seq_by_acc("CAA41635");

# Second exemple de requete
my $query=Bio::DB::Query::GenBank->new(
  -query => "glutamate dehydrogenase",
  -db => "protein");
my $seqio = $genbank->get_Stream_by_query($query);
while(my $seq=$seqio->next_seq){
  print $seq->display_id()."\n";
}
```

Chargement de la banque

Requête directe par numéro d'accès (possible aussi par identifiant ou gi)

Création d'une requête complexe

Récupération des résultats de la requête

Parcours des résultats, séquence par séquence

Perl Orienté Objet - BioPerl

Rappels de Perl http://formation-perl.fr/guide-perl.pdf

Références

Modules

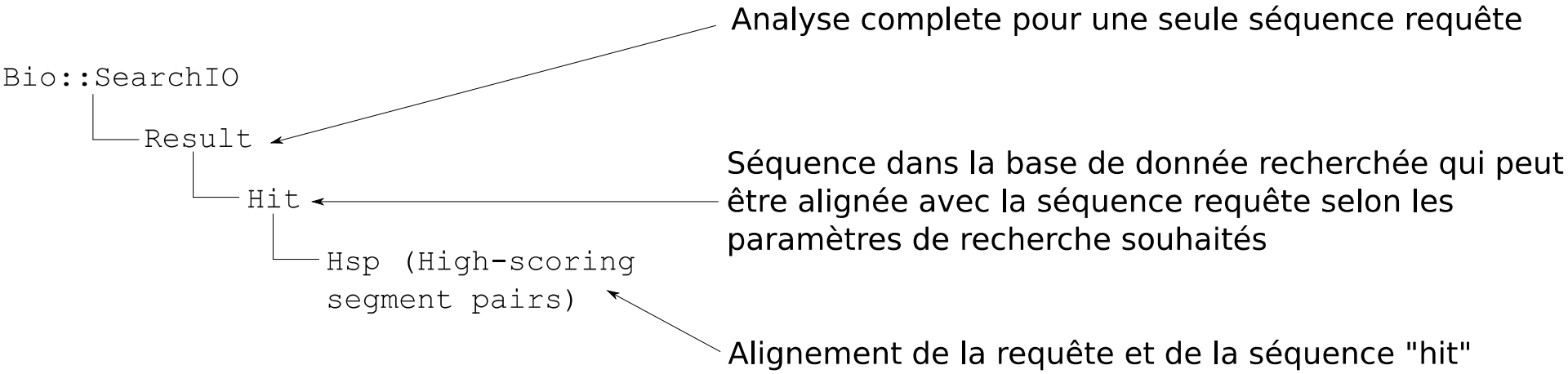
Programmation Orientée Objet

BioPerl

- Bio::Seq
- Bio::SeqIO
- Bio::DB
- Bio::SearchIO

Bio::SearchIO

Classe séparant les comptes-rendus de « recherche » en une hiérarchie de composants



Modules supportés

Blast, fasta, blasttable, blastxml, erpin, infernal, megablast, psl, waba, axt, …

Exemples de parcours d'un fichier issu d'une requête Blast

```
use strict;
use Bio::SearchIO;

my $in = new Bio::SearchIO(
  -format => "blast",
  -file => "report.bls");

while(my $result = $in->next_result){
  while(my $hit = $result->next_hit){
    while(my $hsp = $hit->next_hsp){
      print "Query=", $result->query_name,
        " Hit=", $hit->name,
        " Length=", $hsp->length('total'),
        " Percent_id=", $hsp->percent_identity,
        "\n";
    }
  }
}
```

Chargement du fichier de sortie Blast

Parcours des `Result`

Parcours des `Hit`

Parcours des `Hsp`

Méthodes des objets Result

```
$result->algorithm();#algorithme utilisé
$result->algorithm_version();#version de l'algorithme
$result->query_name();#nom de la requête
$result->query_accession();#numéro d'accession de la
requête
$result->query_length();#longueur de la requête
$result->query_description();#description de la requête
$result->database_name();#nom de la base de données
$result->database_letters();#nombre de résidus dans
la base de données
$result->database_entries();#nombre d'entrées dans
la base de données
$result->available_statistics();#statistiques
utilisées
$result->available_parameters();#paramètres utilisés
$result->num_hits();#nombre de hits
$result->hit();#liste des hits
```

Méthodes des objets Hit

```
$hit->name();#nom du hit
$hit->length();#longueur de la séquence hit
$hit->accession();#numéro d'accession
$hit->description();#description du hit
$hit->algorithm();#algorithme
$hit->raw_score();#score brut
$hit->significance();#signification
$hit->hsps();#liste des hsp
$hit->num_hsps();#nombre de hsp
$hit->locus();#nom du locus
$hit->accession_number();#numéro d'accession
```

Méthodes des objets HSP

```
$hsp->algorithm();#algorithme
$hsp->evalue();#e-value
$hsp->expect();#alias pour e-value
$hsp->frac_identical();#proportion d'identité
$hsp->frac_conserved();#proportion de conservation
$hsp->gaps();#nombre de gaps
$hsp->query_string();#chaine de requête pour
alignement
$hsp->hit_string();#chaine des hits pour alignement
$hsp->homology_string();#chaine pour alignement
$hsp->length('total');#longueur du HSP (avec gaps)
$hsp->length('hit');#longueur du hit participant à
l'alignement sans les gaps
$hsp->length('query');#longueur de la requête
participant à l'alignement sans les gaps
$hsp->hsp_length();même chose que length('total')
$hsp->frame();
```

Méthodes des objets HSP

```
$hsp->num_conserved();#nombre de résidus conservés
$hsp->num_identical();#nombre de résidus identiques
$hsp->rank();#rang du HSP
$hsp->seq_inds('query','identical');#positions
identiques dans la requête
$hsp->seq_inds('query','conserved-not-
identical');#positions conservés mais pas identiques
identiques dans la requête
$hsp->seq_inds('query','conserved');#positions
conservées identiques dans la requête
$hsp->seq_inds('hit','identical');#positions identiques
   dans le hit
$hsp->seq_inds('hit','conserved-not-identical');
#positions conservés mais pas identiques identiques
dans le hit
$hsp->seq_inds('hit','conserved'); #positions
conservées identiques dans le hit
$hsp->score;#score
$hsp->range('query');#début et fin de la requête
$hsp->range('hit'); #début et fin du hit
```

Méthodes des objets HSP

```
$hsp->percent_identity();#pourcentage d'identité
$hsp->strand('hit');#brin du hit
$hsp->strand('query');#brin de la requête
$hsp->start('hit');#début du hit
$hsp->start('query');#début de la requête
$hsp->end('hit');#fin du hit
$hsp->end('query');#fin de la requête
$hsp->matches('hit'); #nombre de positions
identiques et conservées pour le hit
$hsp->matches('query'); #nombre de positions
identiques et conservées pour la requête
$hsp->get_aln;#alignement : objet Bio::SimpleAlig
```