Université du Maine
P. JACOBONI, E. DELOZANNE

Introduction à La Programmation Orienté Objet.
Application à Ruby

TD n° 1 : Introduction à Ruby

Sujet 1 : Le compte en Banque (Episode 1)

Implémenter en Ruby la classe Compte qui modélise un compte en banque. Les objets de cette classe

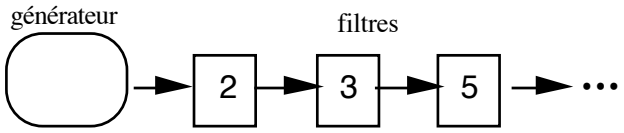
1) sont caractérisés par leur numéro, leur titulaire et leur solde
2) sont capables de donner leur solde, de deposer une somme et de retirer une somme

Sujet 2 : Le crible d'Érathostène

Donner une réalisation en Ruby du crible d'Erathostène, un algorithme qui permet de trouver l'ensemble des nombres premiers inférieurs ou égaux à n, n étant donné.

Principe :

Un générateur envoie des nombres de 2 à n de façon incrémentale. Une série de filtres sont placés de façon à récupérer les valeurs envoyées par les filtres précédents. Lorsqu'un nombre x est récupéré par un filtre i , si x est divisible par i, alors x n'est pas premier. Sinon x passe sur le filtre suivant. Si x n'a rencontré aucun diviseur à la sortie de la chaîne des filtres, on construit au bout de la chaîne un filtre de valeur x.



a) Donner une description en français des classes nécessaires à la réalisation de ce programme.
b) Écrire la définition des classes et des fonctions membres correspondantes
c) Écrire un programme de test demandant un nombre à l'utilisateur et affichant tous les nombres premiers inférieurs à ce nombre

Rappels/Compléments de syntaxe

<table>
<tr><th>Opération</th><th>Exemples</th></tr>
<tr><td>méthodes pour afficher</td><td>Le module <code>Kernel</code> est inclus dans la classe <code>Object</code>, ses méthodes sont accessibles dans tous les ojets Ruby. On utilise la méthode <code>print</code> du module <code>Kernel</code> et les deux variables prédéfinies <code>$,</code> et <code>$\</code>
La variable $, contient le séparateur des différentes valeurs, alors que la variable <b>$\ contient le caractère séparateur de ligne.
<table>
<tr><td><pre>print "cat", [1,2,3], 99, "\n"
On change le séparateur de champ
$, = ", " #
On change le séparateur de ligne
$\ = "\n"
print "cat", [1,2,3], 99</pre></td><td><pre>cat12399

cat, 1, 2, 3, 99</pre></td></tr>
</table></td></tr>
<tr><td>reste de la division euclidienne : %</td><td><code>7 % 2</code> retourne 1
<code>6 % 2</code> retourne 0</td></tr>
<tr><td>tester si deux objets sont égaux : ==</td><td><code>2 == 3</code> retourne false
<code>2 == 2.0</code> retourne true.</td></tr>
<tr><td>tester si un objet est égal à nil</td><td><code>2 == nil</code> retourne false
<code>false == nil</code> retourne false
<code>nil == nil</code> retourne true</td></tr>
<tr><td>Alternative</td><td>une expression <code>if</code> en Ruby est proche des instructions if des autres langages
<pre>if artiste == "Fersen" then
  prenom = "Thomas"
elsif artiste == "Bénabar" then
  prenom = ""
else
  prenom = "inconnu"
end</pre>
<code>if</code> est une expression. Elle retourne une valeur que l'on peut utiliser.
<pre>prenom = if artiste == "Fersen" then
           "Thomas"
         elsif artiste == "Bénabar" then
           ""
         else
           "inconnu"
         end</pre>
Ruby possède aussi une forme négative du <code>if : unless</code>
<pre>unless duree > 180 then
  cout = .25
else
  cout = .35
end</pre>
Ruby supporte l'expression conditionnelle du C.
<pre>cout = duree > 180 ? .35 : .25</pre></td></tr>
<tr><td>Répétitions et Itérateurs</td><td>Les possibilités de Ruby pour les répétitions sont « réparties » dans différentes classes à travers les « iterateurs ». Il existe d'autres formes de répétitons (cf. doc) Exemples.
<table>
<tr><td><pre>3.times do
  print "Ho! "
end</pre></td><td><pre>Ho! Ho! Ho!</pre></td></tr>
<tr><td><pre>0.upto(9) do |x|
  print x, " "
end</pre></td><td><pre>0 1 2 3 4 5 6 7 8 9</pre></td></tr>
<tr><td><pre>0.step(12, 3) {|x| print x, " " }</pre></td><td><pre>0 3 6 9 12</pre></td></tr>
</table></td></tr>
</table>