Université du Maine L3-Informatique
P. JACOBONI, E. DELOZANNE

# Introduction à La Programmation Orienté Objet.
# Application à Ruby

## TD n° 2 : Introduction à la syntaxe Ruby

---

**Sujet 1 : Le compte en Banque (Episode 2)**

La Classe Compte du TD n°1 d'introduction à Ruby un peu moins rudimentaire avec un seuil de retrait et des fonctions d'accès générées par le "Coding Assistant" aux variables d'instances numero et titulaire. La méthode attr permet de générer les méthodes d'accès en lecture / écriture de la variable d'instance considérée

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, de donner le seuil de retrait autorisé, et de retirer une somme si le seuil de retrait n'est pas atteint. Par défaut le seuil est fixé à 0

Pour créer un objet de cette classe on envoie le message ouvrir à la classe Compte avec pour arguments le numero, le titulaire et le solde du compte que l'on veut créer. Faire le nécéssaire pour pouvoir afficher un compte c avec l'instruction « `print c` »

**Sujet 2 : Ruby games**

Les grands classiques…Cela permet d'appréhender syntaxe et lexique sans se casser la tête à inventer des algorithmes. Commençons par la plus célèbre des fonctions de tout enseignement de programmation qui se respecte :

1. Calculer la factorielle d'un entier ( 3 versions)
2. Élever un nombre à une puissance donnée (exposant entier)
3. Tester si une chaîne est un palindrome
4. Transformer une chaîne en écrivant les voyelles en majuscules et les autres caractères en minuscules
5. Convertir une chaîne en entier
6. Renverser une chaîne de caractères (par caractère/ par mot)
   a. Par caractère : "Ruby est très joli" → "iloj sèrt tse ybuR"
   b. Par Mot : "Ruby est très joli" → "joli très est Ruby"
7. Ecrire une méthode enleveStr( S2) de la classe Chaîne permettant d'enlever toutes les occurrences de la chaîne s2 dans la chaine receveuse.
8. Ecrire une méthode `majusMot` qui met en majuscule la première lettre de chaque mot et en minuscule le reste du mot. (Ceci quelque soit la chaîne d'origine)

## Sujet 3 : Simulation d'afficheurs lumineux – Partie 1

Le but de cet exercice est de simuler en Ruby les afficheurs lumineux qu'on voit un peu partout et qui font circuler un texte en boucle.

### A) Le décaleur

Intéressons-nous d'abord au *décaleur*. C'est un objet qui stocke une suite de L caractères avec L > 0 (et pas plus). L est constant et est appelé *largeur* du décaleur. Initialement, un décaleur contient L espaces.

Les quatre fonctionnalités d'un décaleur sont :

- Fournir un accès à la largeur du décaleur,
- Remettre à Zéro : force tous les caractères à *espace*,
- Décaler d'une position vers la gauche de sa suite de caractères : le caractère le plus à gauche est supprimé de la suite et est renvoyé par la méthode. Le nouveau caractère le plus à droite est fixé à la valeur du paramètre de la méthode.
- On doit pouvoir afficher un décaleur.

**Définir** entièrement en Ruby la classe `Decaleur`. Préciser en particulier ses variables d'instances et ce qu'elles représenteront. Écrire en Ruby le code de toutes les méthodes de la classe, sans oublier en particulier les méthodes de création/initialisation.

Exemple d'utilisation de la classe `Decaleur`

| Test | Affichage |
|---|---|
| `d=Decaleur.creer(5)` | |
| `puts d` | `<<----->>` |
| `d.decale('a')` | |
| `puts d` | `<<----a>>` |
| `d.decale('b')` | |
| `puts d` | `<<---ab>>` |
| `d.decale('c')` | |
| `puts d` | `<<--abc>>` |
| `d.decale('d')` | |
| `puts d` | `<<-abcd>>` |
| `d.decale('e')` | |
| `puts d` | `<<abcde>>` |
| `d.decale('f')` | |
| `puts d` | `<<bcdef>>` |
| `puts d.getLargeur` | `5` |