Université du Maine
P. JACOBONI, E. DELOZANNE

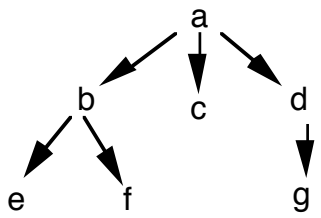
Introduction à La Programmation Orienté Objet.
Application à Ruby
TD n° 5 :

Exercice 0 : Elément de correction du TP 2 Usine

Exercice 1 : Une expression postfixée étant contenue dans une chaîne, écrire le code Ruby permettant de l'évaluer.. ex : '128 12+10-2*' => 260

Exercice 2 : (exam de février 1976)

1) On se propose de mettre en œuvre en Ruby les arbres généraux dans une représentation par liste des fils.
Rappel : un arbre général n'est jamais vide.
Un arbre est caractérisé par l'étiquette de sa racine et par la liste de ses fils.
On prévoira :



construction :
- de construire un arbre à partir de la liste de ses fils et de l'étiquette de sa racine (construction à côté)

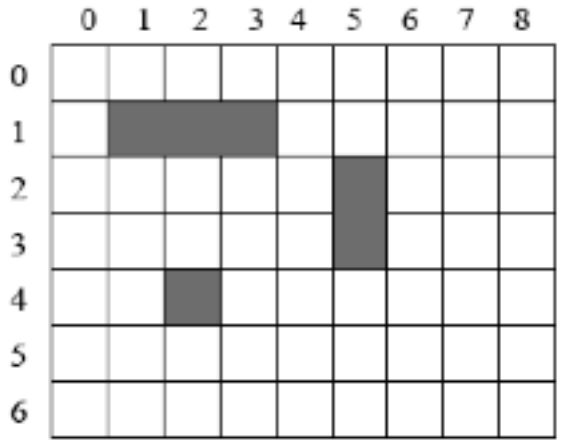
accès :
- de retourner le premier fils d'un arbre
- de retourner le fils suivant d'un fils d'un arbre
- de retourner l'étiquette d'un arbre

test :
- tester si l'arbre est reduit à une feuille (liste des fils vide)
- tester si le fils d'un arbre est son dernier fils

2) On définira des opérations permettant
- de construire l'arbre donné en exemple sur la figure
- d'afficher les étiquettes d'un arbre par ordre préfixe (racine, premier,… dernier) :
  (a (b (e) (f)) (c) (d (g)))
- d'afficher les étiquettes par niveau :
  a b c d e f g
- de saisir les étiquettes des nœuds

Exercice 3 : Une Modélisation de la Bataille Navale

On veut réaliser un jeu de bataille navale. Un jeu de bataille navale se compose d'une grille et d'un ensemble de bateaux, chaque bateau se compose d'un ensemble de taille fixe d'éléments. Un **escorteur** comprend 3 éléments, un **remorqueur** 2 et un **sous-marin** un seul élément.
Chaque élément est caractérisé par sa position et par son état : sain ou touché.
Une grille contient un ensemble de bateaux (stockés dans une Liste). Un bateau est caractérisé par l'ensemble de ses éléments (stockés dans un tableau).



Les **sous-marins** ont la possibilité de plonger. Lorsqu'ils plongent ils ne peuvent pas être touchés.
Les **escorteurs** disposent de canons tandis que les **remorqueurs** sont caractérisés par leur vitesse d'intervention (un coefficient multiplicateur / aux autres bateaux et leur puissance de remorquage

Voici comment on instancie une flotte de bateaux (qui correspond à la figure) :

```
# un escorteur horizontal dont le premier élément est en 1,1 et qui dispose
# de 20 canons.
# (les coordonnées ont leur origine en 0,0).
bateau1 = Escorteur.new(1,1, true, 20);
# un Remorqueur vertical dont le premier élément est en 2,5 deux fois plus
# rapide et de puissance 10000 Carambars (unités de puissance des
# remorqueurs)
bateau2 = Remorqueur.new(2,5,false, 2, 10000);
# un sous-marin en 4,2 et en surface (false = en plongée).
bateau3 = SousMarin.new(4,2,true);
# le plateau du jeu
grille1 = Grille.new(7,9);
grille1.ajouterBateau(bateau1);
grille1.ajouterBateau(bateau2);
grille1.ajouterBateau(bateau3);
```

1. Donnez la liste des classes nécessaires et leurs caractéristiques principales (vi et méthodes)
2. Donnez le code des méthodes `ajouterBateau` et `enleverBateau` dans `Grille`.
3. Donnez le code de la méthode `estTouche(px, py)` définie dans Bateau (et éventuellement dans ses sous-classes) qui retourne 0 lorsque le coup est dans l'eau, 1 si le coup tombe sur un élément touché, 2 si le coup tombe sur un élément touché pour la première fois, et 3 si le bateau est coulé (tous les éléments sont touchés).
4. Donnez en Ruby le code complet des classes Bateau définies (variables d'instances, création / initialisation, méthodes)
5. Donnez le code de la méthode `coup(px, py)` définie dans `Grille`, qui retourne -1 si le coup est en dehors de la grille, 0 lorsque le coup est dans l'eau, 1 si le coup tombe sur un élément déjà touché, 2 si le coup tombe sur un élément touché pour la première fois et 3 si le bateau est coulé. Si le bateau est coulé, il est alors supprimé de la grille.
6. *On suppose que les bateaux peuvent avancer, sauf lorsque les sous-marins sont en plongée. Ecrire la méthode `avancer(dx, dy)` dans `Bateau` (et éventuellement dans ses sous-classes) qui fait avancer le bateau dans la direction définie par `dx` et `dy` (la valeur de `dx` et `dy` est -1 ou 1, ce qui implique que les bateaux peuvent avancer en diagonale). Cette méthode renvoie un booléen qui indique si le déplacement est possible.*