Cours de programmation orientée-objet

Enoncé du travail

Année 2010–2011

Ce travail doit être rendu sous la forme d'un programme source Java, transmis par courrier électronique à `degbomont@montefiore.ulg.ac.be` *pour le* **lundi 2 mai 2011 à minuit** *au plus tard. Les travaux rendus en retard et les travaux plagiés ne seront pas corrigés.*

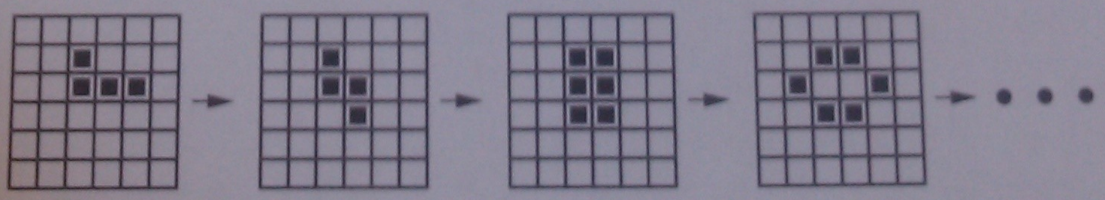
Le "jeu" décrit ci-dessous est inspiré du jeu de la vie inventé par John Conway au début des années 70. Notre jeu est un automate cellulaire très simple : des cellules sont disposées sur une grille carrée de taille arbitraire. A chaque étape, on détermine si une cellule vit ou meurt en appliquant les règles suivantes :

- Si une cellule est entourée de 2 ou 3 autres cellules, elle vit.
- Si elle est entourée par 0 ou 1 autre cellule, elle meurt de solitude.
- Si elle est entourée par 4 cellules ou plus, elle meurt étouffée.
- Si une case de l'échiquier est vide mais entourée d'exactement 3 cellules, il apparaît une nouvelle cellule sur cette case.
- Lorsqu'une cellule meurt, elle disparait de l'échiquier.

Le voisinage d'une case de l'échiquier est constitué par les cases qui lui sont adjacentes verticalement, horizontalement et en diagonale (c'est-à-dire 8 cases si la case n'est pas située au bord de la grille).

A chaque tour, on décide d'abord pour l'ensemble de la population des cellules lesquelles vont survivre, mourir ou naître avant de modifier cette population.

Exemple : La population suivante se stabilise après quelques générations :



On demande d'écrire un programme Java capable d'afficher successivement les populations générées à partir d'une population initiale et d'écrire la situation de la grille dans un fichier après un nombre fixé de générations.

Cette population initiale sera déterminée soit en plaçant au hasard des cellules sur un carré de cases situé au centre de la grille, soit en utilisant un fichier décrivant une situation particulière.

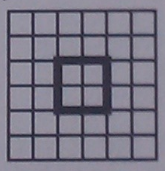
L'affichage, ainsi que la représentation dans un fichier d'une situation de grille, se fait sous forme de texte, les cellules étant représentées par le caractère "*" et les cases vides par un caractère ".".

Exemple : Une situation d'une grille 6x6

```
*.....
.*.*..
.**...
..*...
.*.*..
*.....
```

Dans le cas d'une population initiale aléatoire, le programme doit recevoir en arguments, et dans cet ordre : un nombre entier n, qui est la taille de la grille, un nombre entier m, qui est la taille du carré destiné à recevoir les cellules initiales, ainsi qu'un nombre entier s, qui est le nombre de générations qui doivent être déterminées. On imposera que m et n soient de même parité, et que la contrainte $0 < m \leq n$ soit respectée. Le nombre de générations est toujours positif.

Exemple : $n = 6$ et $m = 2$.



Enfin, dans le cas d'une population initiale imposée, le programme doit recevoir le nom d'un fichier texte contenant une situation de grille en premier argument. Le nombre de générations successives devant être déterminées et affichées, est passé en second argument sous la forme d'un nombre entier.

Dans les deux cas, les générations successives doivent être affichées sur la console. Une fois le nombre de générations demandé atteint, la situation finale de la grille doit toujours être sauvée dans un fichier texte nommé `fin.txt`.

Organisation souhaitée du programme : On veillera à écrire un programme respectant les principes de la programmation orientée-objet. Chaque case de la grille sera représentée par un objet propre, communiquant de manière appropriée avec ses voisins. Il en sera de même pour les éléments nécessaires au jeu (grille, cellules, fichier texte, ...).