Application : Les robots pollueurs

On considère que le monde est constitué d'une matrice de cases. Diffiérents types de robots agissent dans ce monde : des robots pollueurs et des robots nettoyeurs.

- Les robots pollueurs se baladent dans le monde et déposent des papiers gras sur les cases où ils se trouvent. Les robots nettoyeurs, quant à eux, ne supportent pas la vue d'un papier gras et l'enlèvent dès qu'ils en voient un sur une case. Les robots pollueurs sont de deux sortes : robots sauteurs et robots qui vont tout droit. Chaque robot pollueur suit son parcours et dépose un papier gras sur chaque case rencontrée. Le parcours précis des robots pollueurs sera donné dans la suite.
- Les robots nettoyeurs parcourent méthodiquement le monde en "boustrophédon", c'est-à-dire qu'ils parcourent la première ligne case par case, puis arrivé en bout de ligne font demi-tour, parcourent la deuxième ligne en sens inverse, et ainsi de suite jusqu'à la dernière case de la dernière ligne. Ils enlèvent, s'il s'en trouve un, le papier gras de la case où ils se trouvent. Parmi les robots nettoyeurs, certains sont distraits et n'enlèvent qu'un papier sur deux.

Exercice 1

Écrire la classe `Monde` qui contient les variables d'instance, les constructeurs et les méthodes suivantes (définir si elles sont d'instance ou de classe) :

- Le nombre de lignes `nbL`, le nombre de colonnes `nbC`, et une matrice booléenne `mat` de `/nbL` lignes et `nbC` colonnes (`true` signifiant la présence d'un papier gras).
- Un constructeur avec paramètres et un sans paramètres `Monde()` : ce constructeur par défaut crée un monde 10*10 sans papiers gras.
- `public String toString()` : retourne une chaîne de caractères décrivant le monde. On représentera un papier gras par le caractère "o", rien par le caractère "."(point).
- `metPapierGras(int i;int j)` : met un papier gras dans la case (i;j).
- `prendPapierGras(int i;int j)` : enlève le papier gras de la case (i;j).
- `estSale(int i; int j)` : teste si la case (i, j) a un papier gras.
- `nbPapiersGras()` : qui rend le nombre de papier gras dans le monde.

Exercice 2

Écrire dans une classe `TestRobot` la méthode `main` et y créer un `Monde` de 10 lignes et 10 colonnes. Y tester les méthodes `metPapierGras`, `prendPapierGras` et `estSale`.

Exercice 3

Écrire la classe `Robot` qui est abstraite car elle n'aura aucune instance, et qui contient les champs et méthodes suivantes :

- `posx`, `posy` : position du robot sur le monde.
- `m` : variable de type Monde. En effet, il faut que le robot connaisse le monde pour pouvoir s'y déplacer et agir.
- deux constructeurs : `Robot(int x,int y,Monde m)` qui crée un robot à la position `(x,y)` et `Robot(Monde m)` qui crée un robot avec une position aléatoire. Le constructeur `Robot(Monde m)` doit appeler l'autre constructeur.
- `vaEn(int i, int j)` : se déplace en `(i, j)`.
- `parcourir()` : méthode abstraite qui sera définie dans les sous-classes.

Exercice 4

Écrire la classe `RobotPollueur` (également abstraite car elle n'aura pas d'instance) qui contient la méthode :

- `polluer()` : met un papier gras là où ce robot se trouve dans le monde.
- constructeurs.

Exercice 5

Écrire la classe `PollueurToutDroit` qui contient les champs et méthodes suivantes :

- `colDepart`, numéro de la colonne où il va se rendre pour commencer sa mission.
- constructeur(s).
- `parcourir()` : cette méthode définit la méthode parcourir abstraite de la classe Robot. Elle décrit un parcours de ce robot dans le monde. Le robot se positionne d'abord dans sa colonne de départ en case (0, ColDepart), puis il va tout droit vers le sud en visitant chaque case de la colonne et dépose un papier gras sur chacune d'elle. Il s'arrête dans la dernière case de la colonne.

Exercice 6

Écrire la classe `PollueurSauteur`, avec les champs et méthodes suivants :

- `deltax` représentant la taille du saut e?ectué à chaque déplacement du sauteur.
- constructeurs.
- `parcourir()` : cette méthode définit la méthode parcourir abstraite de la classe Robot. Elle décrit un parcours de ce robot dans le monde. Le robot se positionne d'abord dans sa colonne de départ en case (0, 0), puis il saute de façon analogue au cavalier des Echecs et va en (1, deltax), puis en (2, 2* deltax), puis en (3, 3*deltax), etc. Si la colonne sort de l'échiquier, on revient au début. Par exemple, la colonne nbC, qui n'existe pas, sera transformée en nbC modulo nbC, i.e, colonne 0. Chaque case rencontrée est souillée d'un papier gras. Le robot s'arrête lorsqu'il atteint la dernière ligne.

Exercice 7

Écrire la classe `RobotNettoyeur`, qui contient les méthodes suivantes :

- `nettoyer()` : enlève le papier gras de la case où se trouve ce robot.
- constructeurs,
- `parcourir()` : cette méthode définit la méthode parcourir abstraite de la classe Robot. Elle décrit un parcours complet de ce robot dans le monde. Il part de la case (0, 0) et parcourt le monde en "boustrophédon". Si la case est sale, il enlève le papier gras qui s'y trouve.

Exercice 8

Écrire la classe `NettoyeurDistrait` qui contient les méthodes suivantes :

- constructeurs.
- `parcourir()` : cette méthode redéfinit la méthode
- `parcourir()`. Elle décrit un parcours complet de ce robot dans le monde. C'est le même parcours que celui des robots nettoyeurs mais comme il est distrait, il n'enlève qu'un papier sur deux.

Exercie 9

Écrire dans une classse TestRobot la méthode `main` pour tester les classes définies dans les exercices précédents.