Programmation orientée objet (C++) – ING2-GSI

TP4 : Héritage, polymorphisme et liaison dynamique

Objectifs

- Utiliser l'héritage pour généraliser/factoriser des propriétés
- Comprendre la liaison dynamique
- Montrer l'intérêt des classes abstraites pour l'extensibilité
- Résoudre un problème complet sur les classes abstraites.

On considère que le monde est constitué d'une matrice de cases. Différents 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 pollueurs sont de deux sortes : robots sauteurs et robots qui vont tout droit. Chaque type de robot pollueur suit son parcours (voir plus loin) et dépose un papier gras sur chaque case rencontrée.
- Les robots nettoyeurs parcourent méthodiquement le monde ligne par ligne, colonne par colonne. Ils ne supportent pas la vue d'un papier gras et dès qu'ils en voient un, l'enlèvent de la case où ils se trouvent. Parmi les robots nettoyeurs, certains sont distraits et n'enlèvent qu'un papier sur deux.

Exercice 5.1

Écrire la classe *Monde* qui contient les attributs, le constructeur et les méthodes suivantes :

- Le nombre de lignes nbL, le nombre de colonnes nbC, et une matrice booléenne mat de nbL lignes et nbC colonnes (vrai signifiant la présence d'un papier gras).
- Un constructeur initialisant les attributs (en particulier la matrice).
- afficherMatrice() : retourne une chaîne de caractères décrivant le monde. On représentera un papier gras par le caractère "o", rien par le caractère "." (point). Le caractère de passage à la ligne est " \n ".
- metPapierGras(i; j) : met un papier gras dans la case (i; j).
- prendPapierGras(i; j) : enlève le papier gras de la case (i; j).
- estSale(i; j) : teste si la case (i, j) a un papier gras.
- nbPapiersGras() : qui rend le nombre de papier gras dans le monde.

Exercice 5.2

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.
- le(s) constructeur(s) nécessaire(s).
- allerEn(int i, int j) : se déplace en (i, j).
- parcourir() : méthode abstraite qui sera définie dans les sous-classes.

Exercice 5.3

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.
- ainsi que le(s) constructeur(s) nécessaire(s).

Exercice 5.4

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.
- le(s) constructeur(s) nécessaire(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 5.5

Écrire la classe *PollueurSauteur*, qui contient les champs et méthodes suivants :

- deltax représentant la taille du saut effectué à chaque déplacement du sauteur.
- le(s) constructeur(s) nécessaire(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 saute de façon analogue au cavalier des Echecs et va en (1, *colDepart + deltax*), puis en (2, *colDepart*), puis en (3, *colDepart + 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 5.6

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

- nettoyer() : enlève le papier gras de la case où se trouve ce robot.
- le(s) constructeur(s) nécessaire(s).

- 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 ligne par ligne, colonne par colonne. Si la case est sale, il enlève le papier gras qui s'y trouve.

Exercice 5.7

Écrire la classe NettoyeurDistrait qui contient les méthodes suivantes :

- le(s) constructeur(s) nécessaire(s).
- 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.

Exercice 5.8

Ecrire un programme principal pour tester vos classes. Ce programme crée un monde de 10 * 10, ainsi que des robots pollueurs et nettoyeurs qui vont successivement mettre des papiers gras ou les enlever. N'oubliez pas d'afficher l'état de la matrice à chaque passage d'un robot.