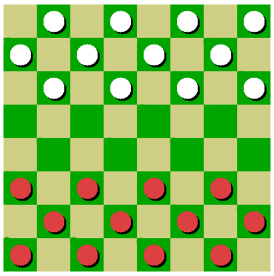
Université Denis Diderot — DEUG MIAS/MASS — IF 122

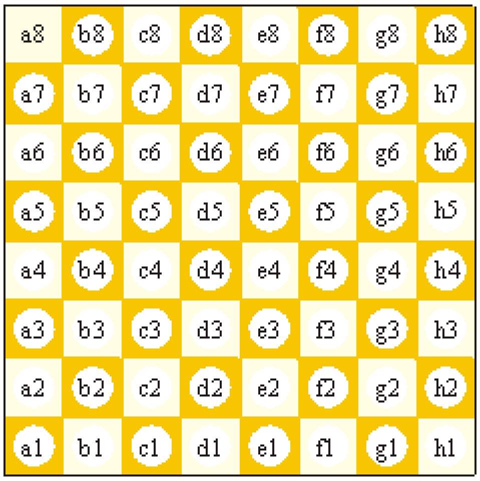
Pariel du 22/04/2005 — Durée : 2 heures — Aucun document n'est autorisé
Le sujet comporte deux parties indépendantes.

1 Le jeu de dame

Le jeu de dame irlandaise se joue sur un échiquier 8x8 posé entre 2 joueurs qui se font face. Au départ chaque joueur dispose devant lui, mais seulement sur les cases noires des 3 premières lignes, les pions de sa couleur (l'un a choisi les blanc, l'autre a choisi les noirs). Ils ont chacun 12 pions au départ.



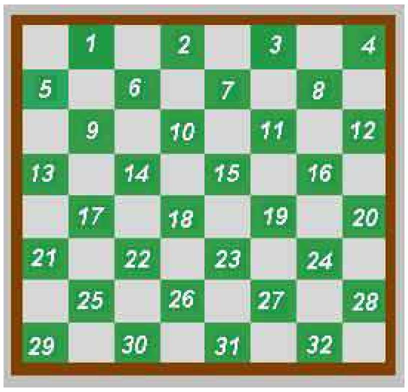
Pour repérer une position, on adopte les mêmes conventions qu'aux échecs, comme indiqué sur la figure suivante :



Sauf qu'une particularité des dames est que toutes les cases ne sont pas utiles. En fait toutes les cases blanches resteront vides, il n'est donc pas utile de leur réserver une place en mémoire.

Nous allons proposer deux modélisation qui prennent en compte cette économie, par contre pour le joueur tout doit rester transparent. Les exercices 1 et 2 sont indépendants des exercices 3 et 4.

Dans la première modélisation on utilise un seul tableau d'entiers a une dimension, avec dans l'idée de respecter la correspondance décrite dans la figure suivante :



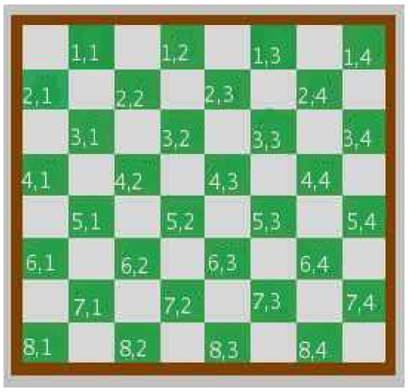

```
public class mod_1{
    public int[] tab;
}
```

Chaque élément du tableau correspond à une case et vaut 0 si cette case est vide, 1 pour un pion noir, et 2 pour un pion blanc.

Exercice 1 Écrivez le constructeur de la classe `mod_1` qui donne la position initiale.

Exercice 2 Écrivez une méthode qui prenne en argument un caractère et un entier, (les coordonnées d'une case vue par un joueur) et qui retourne la position correspondante dans notre modélisation.

Dans la deuxième modélisation on utilise un tableau d'entiers a deux dimension, et qui cette fois respecte la correspondance décrite dans la figure suivante :



```
public class mod_2{
    public int[][] tab;
}
```

De nouveau on a 0 pour une case vide, 1 pour un pion noir, et 2 pour un pion blanc.

Exercice 3 Écrivez le constructeur de la classe `mod_2` qui donne la position initiale.

Exercice 4 Écrivez une méthode qui prenne en argument un caractère et un entier, (les coordonnées d'une case vue par un joueur) et qui retourne la position correspondante dans cette modélisation.

Exercice 5 Écrivez une méthode dans la classe `mod_1` qui réalise l'affichage d'un plateau sous la forme :

```
-----------------
| |B| |B| |B| |B|
-----------------
|B| |B| |B| |B| |
-----------------
| |B| |B| |B| |B|
-----------------
| | | | | | | | |
-----------------
| | | | | | | | |
-----------------
| |N| |N| |N| |N|
-----------------
|N| |N| |N| |N| |
-----------------
| |N| |N| |N| |N|
-----------------
```

Exercice 6 Faites de même dans la classe `mod_2`.

2 Opérations sur les ensembles

L'objectif de ces exercices est de manipuler les opérations de base sur les ensembles à l'aide de la structure liste (simplement chaînée). Une liste représente un ensemble dont les valeurs des éléments sont différentes deux à deux. La structure `Ensemble` est donnée comme la suivante :

```
import fr.jussieu.script.Deug;

public class Ensemble{
    public int valeur;
    public Ensemble suivant;
 }
```

L'ensemble vide vaut `null`.

Dans les exercices de 7 à 15, on suppose que les valeurs des éléments d'un ensemble sont différentes deux à deux.

Exercice 7 Écrire une méthode qui vérifie si un ensemble est vide ou pas.

```
public static boolean estVide(Ensemble l)
```

Exercice 8 Écrire une méthode pour afficher toutes les valeurs d'un ensemble.

```
public static void affiche(Ensemble l)
```

Exercice 9 Écrire une méthode qui retourne le nombre d'éléments d'un ensemble.

```
public static int nombreElement(Ensemble l)
```

Exercice 10 Écrire une méthode pour tester si l'ensemble l contient la valeur a.

```
public static boolean estElement(Ensemble l, int a)
```

Exercice 11 Écrire une méthode qui ajoute une valeur à la fin d'un ensemble.

```
public static Ensemble ajoute(Ensemble l, int v)
```

Exercice 12 Écrire une méthode qui retourne l'intersection de deux ensembles.

```
public static Ensemble intersection(Ensemble l, Ensemble s)
```

Exercice 13 Écrire une méthode qui retourne l'union de deux ensembles (rappelons que les valeurs des éléments de l'union sont différentes deux à deux).

```
public static Ensemble union(Ensemble l, Ensemble s)
```

Exercice 14 Écrire une méthode qui retourne la soustraction de l par rapport à s (c'est-à-dire les valeurs dans l mais pas dans s)

```
public static Ensemble soustraction(Ensemble l, Ensemble s)
```

Exercice 15 Écrire un programme pour créer deux ensembles l et s, puis calculer l'union, l'intersection de ces deux ensembles.