Les ensembles implémentés via un tableau de booléens

Exercices obligatoires

A Implémentation de l'interface `Ensemble`

A1 Implémentez l'interface `Ensemble` en utilisant un tableau de booléens.
Appelons cette classe *EnsembleTableBooleens* **.**
Pour obtenir l'indice de l'objet à insérer, utilisez la méthode `hashcode()` **.**
Testez cette classe via la classe *TestEnsembleTableBooleens*.

A2 Sur moodle, répondez au questionnaire à choix multiples *EnsembleTableBooleens*.

B Applications utilisant un ensemble

B1 La société X voudrait une application qui permet de savoir si un employé de son département informatique est connecté ou pas sur le réseau.
La société compte 137 employés dans son département informatique.
Les logins de ces employés sont : info001 → info137

La classe *ConnexionInfo* que vous allez compléter possède les méthodes suivantes :

```
void connecter(LoginInfo login)
void deconnecter(LoginInfo login)
boolean estConnecte(LoginInfo login)
```

Utilisez un ensemble pour retenir les logins connectés. Ces 3 méthodes auront une complexité en O(1).
N'oubliez pas d'écrire la méthode `hashCode()` de la classe *LoginInfo.*
Testez cette classe avec la classe *TestConnexionInfo*.

B2 Cette même société voudrait étendre cette application à tous les employés.
La société compte 256 employés qui sont répartis dans 3 départements.
Voici les logins :
Département informatique : info001 → info137
Département marketing : mark001 → mark073
Département administratif : admi001 → admi046
Complétez la classe *Connexion.* Il vous faut aussi écrire la méthode `hashCode()` de la classe *Login !*
Testez cette classe avec la classe *TestConnexion*.

C Limites de cette implémentation des ensembles

Pour traiter les problèmes qui suivent, l'ensemble est sans doute la structure de données la plus efficace, MAIS l'implémentation de l'ensemble avec une table de booléens telle que vous l'avez écrite en début de séance ne pourra convenir !
Voyez-vous pourquoi?

Imaginez une solution à ces différents problèmes.
Surtout ne les programmez pas.
Lors de la prochaine présentation sur les ensembles (EnsembleTableViaTableHashing.ppt), nous discuterons de vos solutions et nous vous en proposerons une à implémenter!!!

C1 Tous les étudiants de l'Institut Paul Lambin ont reçu un login permettant de se connecter sur le réseau.
Le login est composé de 6 caractères. Le premier caractère est la première lettre du prénom. Il est suivi des 5 premiers caractères du nom.
L'Institut voudrait une application qui permet de savoir si un étudiant est connecté ou pas sur le réseau.

le probleme ici est que la capacité que l'on doit prévoir est de 26^6
pour avoir toute les possibilité de prénom et de nom, cela n'est pas du tout optimal
on aura un ensemble de plus d'1M d'elements.

C2 Le personnel de la société X a le droit de placer sa voiture dans le parking de la société.
Pour éviter les indésirables, l'entrée du parking est munie d'une barrière.
La barrière est équipée d'un détecteur de plaques.
On vous demande d'écrire un programme qui propose les fonctionnalités suivantes :

```
boolean retirerPlaque(String numeroPlaque)
boolean ajouterPlaque(String numeroPlaque)
boolean plaqueAcceptee(String numeroPlaque)
```

Le probleme ici est le meme que pour le C1, on doit prendre toute les combinaison de plaque possible.

D Sudoku

D1 Complétez la classe Grille9X9.
Les méthodes demandées peuvent s'écrire facilement en utilisant un ensemble.
Vous pouvez utiliser la classe Ensemble1A9 fournie sur moodle.
Testez votre classe avec la classe *TestSudoku*.

Le constructeur de cette classe lève une *IllegalArgumentException* si le tableau d'entiers passé en paramètre ne correspond pas à une grille 9x9 qui contient uniquement les chiffres 1 à 9.

Sudoku (classique)
Chaque ligne, chaque colonne et chaque bloc doit contenir les chiffres de 1 à 9, une seule fois.

Exemple :

4	2	1	5	6	3	7	8	9
6	3	8	7	9	4	1	2	5
5	9	7	8	2	1	4	6	3
7	1	6	4	3	2	5	9	8
9	8	2	1	7	5	3	4	6
3	5	4	6	8	9	2	1	7
2	6	5	3	4	8	9	7	1
8	4	3	9	1	7	6	5	2
1	7	9	2	5	6	8	3	4

Exercices supplémentaires

D2 Il existe des variantes au sudoku classique.
Complétez les méthodes `estUnSudokuDiagonal()` et `estUnHyperSudoku()`.

Diagonal
Chaque ligne, chaque colonne, chaque bloc et chaque diagonale doit contenir les chiffres de 1 à 9, une seule fois.

Exemple :

6	3	2	1	5	8	7	9	4
4	7	8	3	2	9	5	1	6
5	9	1	7	6	4	3	8	2
1	8	7	5	9	2	6	4	3
3	2	6	4	8	7	9	5	1
9	4	5	6	1	3	2	7	8
2	6	9	8	7	1	4	3	5
8	5	4	9	3	6	1	2	7
7	1	3	2	4	5	8	6	9

Hyper Sudoku
Chaque ligne, chaque colonne, chaque bloc et chaque zone grise doit contenir les chiffres de 1 à 9, une seule fois.

Exemple :

8	9	1	6	5	7	4	2	3
5	6	2	4	1	3	7	9	8
4	7	3	8	9	2	5	1	6
3	1	5	9	7	6	8	4	2
6	8	7	2	4	5	1	3	9
2	4	9	3	8	1	6	7	5
9	2	8	1	6	4	3	5	7
1	5	6	7	3	9	2	8	4
7	3	4	5	2	8	9	6	1