Licence Informatique **UE INF203**
Université Bordeaux 1 **Programmation par objets**
2006-2007 **Travaux Dirigés**
feuille 5

Télécharger les fichiers qui accompagnent le td à l'aide de la commande
`wget http://dept-info.labri.fr/ENSEIGNEMENT/poo/src/fichiers-5.tar.gz` puis décompacter cette archive au moyen de la commande `tar -xvzf fichiers-5.tar.gz`. Se placer dans le répertoire `td05`.

Techniques de résolution des Sudoku

Une grille de sudoku est un matrice 9x9 partionnée en 9 carrés 3x3. Cette grille contient des chiffres de 1 à 9 et a les propriétés suivantes :

chaque ligne de 9 cases contient tous les chiffres de 1 à 9.
chaque colonne de 9 cases contient tous les chiffres de 1 à 9.
chaque carré de la partition contient tous les chiffres de 1 à 9.

Il ne peut donc y avoir deux fois le même chiffre ni dans une ligne ni dans une colonne, ni dans un carré de la partition. Le jeu consiste à remplir une grille qui a été fournie incomplète.

On vous propose d'écrire un certain nombre de méthodes qui permettent de résoudre une partie des problèmes de Sudoku qui paraissent dans la presse.

Vous utiliserez la classe `Liste` vue en cours, qui contient le constructeur de la liste vide et les méthodes d'instances publiques suivantes :

`int longueur ()`	retourne le nombre d'entiers de la liste
`boolean appartient(int a)`	indique si l'entier `a` appartient a la liste
`void ajoutDebut(int a)`	ajoute l'entier `a` en debut de la liste
`String toString ()`	transforme une liste en chaine de caractères
`void supprimer (int a)`	supprime l'entier `a` de la liste
`int valeur()`	retourne la valeur du premier élément de la liste
`boolean equals(Object o`	retourne true si les liste sont égales(mêmes valeurs dans le même ordre

La classe `Sudoku` contient les champs suivants
– une grille `gr` qui contient les valeurs connue de la grille à résoudre (une valeur égale à zéro signifie que l'on n'a pas encore rempli la case correspondante).
– Un tableau à deux dimensions `grL` contenant des listes d'entiers ; ainsi `grl[i][j]` contiendra la Liste des entiers que l'on peut éventuellement inscrire dans la case `gr[i][j]`, c'est à dire tous les entiers qui ne sont pas sur la ligne `i`, la colonne `j` ou sur le carré 3x3 qui contient la case `gr[i][j]`.

On dispose d'une ébauche de la classe `Sudoku` et d'une classe `SudokuTest`. Cette dernière classe contient un certain nombre de grilles de Sudoku destinées à réaliser vos tests.

1 constructeur et initialisations

1. Écrire une méthode `String toString()` qui retourne une chaîne représentant la grille de sudoku sous la forme d'une matrice carrée 9x9.
   Les cases dont la valeur n'est pas fixée seront représentées par des `*`.
2. Écrire une méthode `copieGrille(int[][] grille)` qui initialisera le champ `gr` avec une copie de la grille fournie en paramètre.
3. Écrire une méthode `void initgrL()` initialisant le tableau à deux dimensions `grL`.
   Avant d'écrire cette méthode,
   – on étudiera le code de la méthode `boolean possible(int a, int i, int j)` qui vous est fournie
   – puis on écrira une méthode `Liste initListe(int i, int j)` qui contient la liste des entiers que l'on peut inscrire dans la case `gr[i][j]`. On utilisera pour cela la méthode `possible`.
   Par convention, le résultat sera la liste vide s'il y a une valeur différente de 0 dans la case `gr[i][j]`.

4. Utiliser la méthode `public String listeDesPossibles(int i, int j)` pour afficher la liste `grL[0][2]` et vérifier votre méthode `initgrL`.

2 résolution : étape 1

1. Écrire une méthode `void supprimer (int a, int i, int j)` qui supprime l'entier `a` des listes `grL[][]` pour toutes les cases qui se trouvent sur la même ligne, la même colonne ou le même carré 3x3 que la case `gr[i][j]`. On s'inspirera de la méthode `possible` pour la façon de parcourir toutes les cases qui se trouvent sur cette ligne, colonne ou carré.
2. Écrire une méthode `void placer (int a, int i, int j)` qui inscrit l'entier `a` dans la case `gr[i][j]`, supprime `a` des listes de possibles sur les cases de la même ligne, même colonne ou même carré 3x3 et qui affecte la Liste vide `grL[i][j]`.
3. Écrire une méthode `boolean ajoutUniquePoss()` qui parcourt toutes les listes `grL[i][j]`, et dans le cas où l'une d'entre elles contient un seul entier `a` inscrit cet entier dans la case correspondante. Cette méthode devra retourner `true` si elle a rencontré une telle possibilité et `false` dans le cas contraire.
4. Utiliser la méthode par `resoudre` pour voir laquelle des 7 grilles données peut être complètement remplie par cette technique.

3 résolution : étape 2

On se propose de mettre en œuvre une autre technique pour résoudre les cas récalcitrants. On va chercher si pour une ligne `i`, et un entier `a` il existe une seule des cases de la ligne `i` qui peut contenir `a`.

1. Écrire une méthode `int uniqueSurLigne(int a, int i)` qui retourne :
   - -1 si l'entier `a` figure sur strictement plus d'une des listes de des cases de la ligne `i` ou sur aucune des listes des cases de cette ligne.
   - Le nombre `j` si cet entier ne figure que sur la liste `grL[i][j]` de cette ligne.
2. Écrire une méthode `boolean ajoutSurLigne(int i)` qui cherche pour tout entier `a` compris entre 1 et 9, s'il ne figure que dans une des listes de la ligne `i` et dans ce cas le place sur `gr` dans la case correspondante. Cette méthode retourne `true` si un tel entier existe et `false` dans le cas contraire.
3. Écrire une méthode `boolean ajoutSurLignes()` qui effectue le même travail sur toutes les lignes.
4. Utiliser la méthode par `resoudre` qui applique tant que cela est possible les deux techniques, `ajoutUniquePoss` et `ajoutSurLignes` pour compléter une grille de Sudoku. Vérifier, sur les exemples donnés, quelles sont les grilles que l'on peut remplir complétement par ces deux techniques.
5. Ecrire des méthodes qui pemettent d'appliquer pour les colonnes la technique développée sur les lignes dans les questions précédentes. On écrira ainsi les méthodes suivantes :

`int uniqueSurColonne(int a, int j)`	retourne l'unique case de la colonne j pouvant contenir `a`
`boolean ajoutSurColonne(int j)`	idem `ajoutSurLigne` mais usr les colonnes
`boolean ajoutSurColonnes()`	effectue l opération précédente sur toutes les colonnes

6. Utiliser la méthode par `resoudre` qui applique tant que cela est possible les trois techniques,`ajoutUniquePoss`, `ajoutSurLignes` et `ajoutSurColonnes` pour compléter une grille de Sudoku.

Si nécessaire, factoriser les méthodes précédemment définies pour éviter la duplication de code.

4 Compléments - résolution : étape 3

Si dans un carré 3x3, il existe une ligne où il ne manque que deux cases, et que les listes de possibles pour ces deux cases sont de longueur 2 et sont égales, alors les deux valeurs des listes de possibles sont nécessairement sur cette ligne. On peut donc supprimer les deux valeurs des listes de possibles des lignes restantes du carré. Le travail réalisé sur les lignes peut être effectué sur les colonnes. Appliquer le raisonnement précédent pour définir une troisième étape de résolution, puis modifier la méthode `résoudre` en consequence.

Remarque : on peut appliquer le même principe avec trois valeurs et des listes de longueur 3.