TD3 (application): jeu de Sudoku

Un jeu de Sudoku

Pour ceux qui s'ennuient, vous pouvez appliquer la récursivité pour remplir une grille de Sudoku. Ici, il s'agit d'un algorithme de "rétro-parcours" c'est à dire que la récursivité va permettre de faire un choix pour une case, de continuer récursivement à remplir la grille, et si finalement ce choix n'était pas bon, la sortie de la récursivité permettra de passer à un autre choix.

Le célèbre jeu de Sudoku consiste à remplir une grille avec des chiffres compris entre 0 et 9 de façon à ce qu'il n'y ait pas deux fois le même chiffre sur la même ligne, la même colonne, et dans le même carré.

La classe Sudoku représente une grille de Sudoku par une matrice d'entiers où la valeur 0 représente une place disponible dans la grille. Pour être plus général, la taille des carrés est quelconque, mais elle est de taille 3 par défaut ce qui correspond à une grille $(9, 9)$ classique (3 carrés de taille 3 en ligne et colonne).

1. Compléter la classe Sudoku.java à chaque fois que vous trouvez le commentaire *// A COMPLETER*. Il peut s'agir de compléter une méthode ou de compléter la javadoc.
2. Expliquer pourquoi certaines méthodes sont "private" et les autres "public".
3. En vous servant uniquement des méthodes fournies dans la classe Sudoku, compléter la méthode *private boolean remplir(int i, int j)* qui place une valeur dans la case (i, j) en continuant à remplir la grille, renvoie true si c'est possible, et renvoie false si aucune valeur ne peut être placée en (i, j). La méthode "remplir" est récursive :
   - le cas d'arrêt est d'avoir fini la grille c'est à dire que l'on est arrivé sur la case $(nombre\_ligne+1, 0)$
   - le cas général procède de la façon suivante. On choisit la première valeur qui est possible pour la case (i, j) c'est à dire qui n'est pas dans la ligne i, ni dans la colonne j, ni dans le carré contenant (i, j). On place cette valeur dans la case (i, j) puis on essaie de remplir le reste de la grille récursivement. Si ce n'est pas possible, on efface en remettant 0 et on fait un autre choix.

   Nota: quand on est sur la case (i, j) la prochaine case est la case

   $(ligneSuivante(i, j), colonneSuivante(i, j))$.

   $ligneSuivante(i, j)$ renvoie $i+1$ si j est la dernière colonne, sinon cela renvoie i. $colonneSuivante(i, j)$ renvoie 0 si j est la dernière colonne (pour passer à la ligne) et renvoie $j + 1$ sinon.

   Une classe de test vous est fournie avec des grilles de différentes difficultés.