|  |  |  |  |
| --- | --- | --- | --- |
| **NOM** |  | **Série** |  |
| **Prénom** |  | **Date** |  |
| **Nom d’utilisateur / login du compte examen** *(Exemple : hs124)* | |  | |

|  |  |
| --- | --- |
| Institut Paul Lambin | |
| Examen d’Agorithmique sur machine | |
| Titulaire(s) : | Annick Dupont, José Vander Meulen, Anthony Legrand, Grégory Seront, Isabelle Cambron |
| Année(s) d’études : | Bloc 1 |
| Durée : | 1h30 ; pas de sortie durant les 60 premières minutes |
| Nombre de pages : | 4 |
| Modalités : | pas d’accès à internet, copions : 4 feuilles manuscrites A4 de votre écriture |

###### Consignes générales

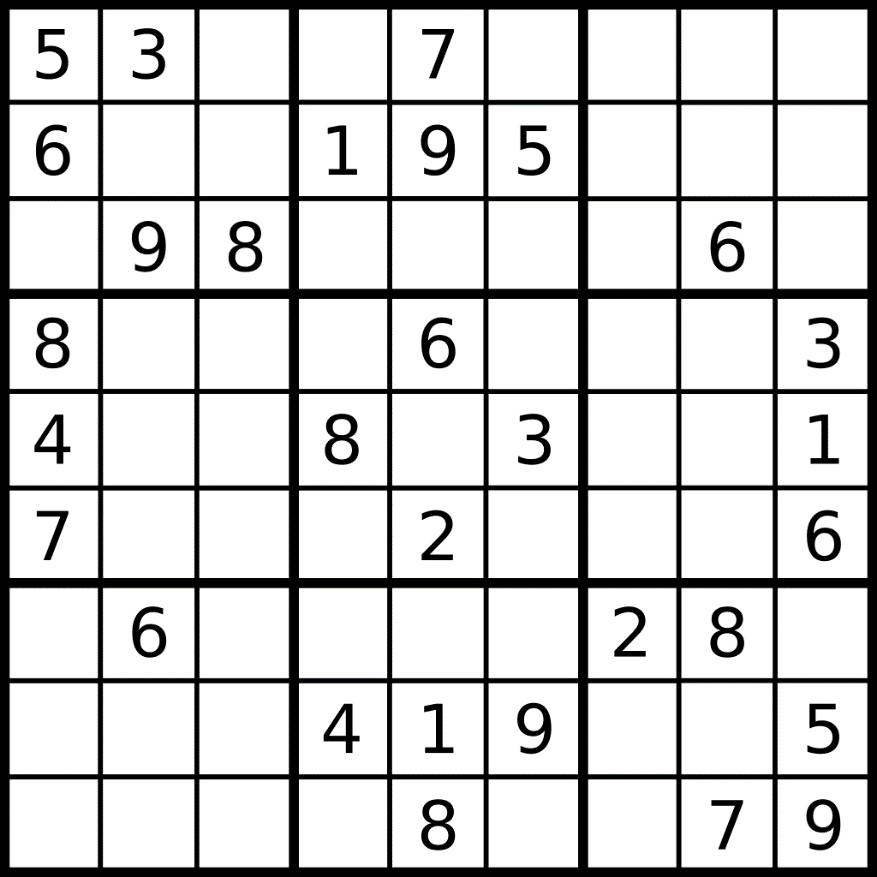
* Au démarrage d’Eclipse, sélectionnez le répertoire **U:\** comme workspace. Créez un projet Java nommé **NOM\_PRENOM** au sein d'Eclipse, dont le chemin est : **U:\NOM\_PRENOM** NB : pas de caractères spéciaux : accents… **NOM** et **PRENOM** sont à remplacer par vos nom et prénom !
* Via l’explorateur de fichiers, vérifiez que votre projet se trouve bien sous **U:\NOM\_PRENOM** (par exemple : **U:\TORVALDS\_LINUS**). Tout fichier présent à un autre emplacement ne sera pas évalué !

**Examen d’algorithmique – juin 2021**

# Sudoku généralisé

Le jeu de Sudoku classique se joue sur une grille de 9x9 cases. Chaque case pouvant comporter un nombre entre 1 et 9 ou être vide.

La grille est constituée de 9 **lignes**, 9 **colonnes** et 9 **régions** comme nous le voyons dans la figure ci-dessous :



**Région**

Figure 1 Grille de Sudoku 9x9 (wikipedia)

Une région est une zone de 3x3 cases.

Pour qu’un remplissage de la grille soit légale, il faut qu’au sein d’une même ligne, d’une même colonne ou d’une même région un nombre n’apparaisse qu’une seule fois. Un nombre peut apparaître plusieurs fois, mais pas sur la même ligne, ni sur la même colonne, ni dans la même région.

Dans ce premier exemple, nous voyons ici une grille illégale car la région centrale comporte deux fois le nombre 2 :

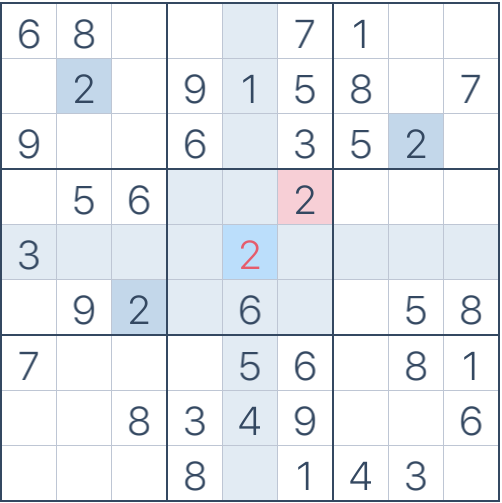


Figure 2 Grille avec région illégale (sudoku.com)

Dans ce deuxième exemple, nous voyons ici une grille illégale car la cinquième ligne comporte deux fois le nombre 7 :

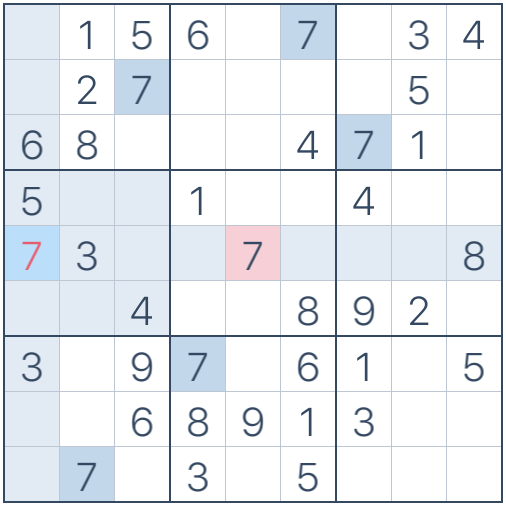


Figure 3 Grille avec ligne illégale (sudoku.com)

Nous appelons Sudoku généralisé un Sudoku dont la taille peut être quelconque. Toutefois pour que l’on puisse avoir des ligne et région de même taille il faudra que la taille du Sudoku (c’est-à-dire le nombre de cases d’un côté du carré) soit un nombre entier élevé au carré.

On peut donc avoir des Sudoku 1x1, 4x4, 9x9, 16x16, 25x25 etc. La taille des côtés des régions est donc égale la racine carrée du côté de la grille. Dans les Sudoku 9x9, la taille des régions est 3x3.

Nous voyons ici un Sudoku 4x4 :

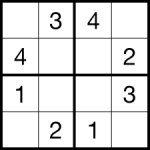


Figure 4 Sudoku 4x4

# Classe SudokuGeneral

Nous vous donnons la classe SudokuGeneral qui représente une grille de Sudoku. La grille est représentée par un tableau à deux dimensions d’entiers. Une case vide est représentée par un 0. Nous supposerons que la grille ne contient que des entiers compris entre 0 et la taille du carré.

Deux dimensions sont définies : *tailleCote* qui représente la taille d’un côté de la grille et *tailleRégion* qui représente la taille d’un côté d’une région. Exemple : dans une grille 9x9, *tailleCote* = 9 et *tailleRegion* = 3.

## A implémenter

### Vérification de la légalité d’un coup

Nous vous donnons le code de la méthode ***boolean isCoupLegal(int i, int j, int nombre)*** qui vérifie que si l'on met le nombre 'nombre' à la position i, j, la grille de sudoku reste légale.

Cette méthode fait appel à trois autres méthodes que vous devrez implémenter :

*private boolean* ***isNombreDansRegion****(int nombre, int coinI, int coinJ)*

*private boolean* ***isNombreDansColonne****(int j, int nombre)*

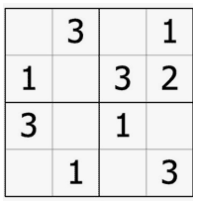
*private boolean* ***isNombreDansLigne****(int i, int nombre)*

Ces méthodes vérifient respectivement la présence du nombre passé en paramètre dans une région, une colonne et une ligne. La région est définie par la position *coinI*, *coinJ* de sa case située dans son coin supérieur gauche.

### Nombre le plus présent

La méthode *nombreLePlusPresent* renvoie le nombre différent de 0 le plus présent au sein de la grille. S'il n'y a que des 0 dans la grille, on renvoie un zéro. En cas d'ex aequo, on renvoie n'importe lequel des vainqueurs.

Par exemple dans la grille suivante, on peut renvoyer soit 3 soit 1.



Piste pour une solution efficace :

Commencez par remplir une table de fréquences et procédez ensuite à une recherche de max dans cette table.

La table de fréquences pour l’exemple est :

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 7 | 4 | 1 | 4 | 0 |

On trouve 7x0, 4x1, 1x2, 4x3 et 0x4

### Jeu de tests

Pour vous aider, nous vous fournissons un jeu de test dans la classe *TestSudokuGeneral*. Bien entendu le fait de réussir ces tests n’implique pas que votre classe soit totalement correcte.

Voici les deux grilles de test utilisées dans la classe :



## Brouillon