TD 2
Itération, tableaux

1 Itérations sur un tableau

Exercice 1. Recherche d'information

1. Rédiger une fonction `enClair` qui à partir d'un tableau d'entiers `t` délivre une chaîne sous la forme {t[0], t[1],..., t[n]} (par exemple {2, 6, 3, 5}). Vous respecterez la spécification de la fonction suivante :

```
/**
* Fonction de construction d'une chaîne de caractères à partir des éléments d'un tableau
↪  d'entiers
*
* @param t tableau d'entiers
* @return chaîne de caractères représentant les éléments du tableau reçu en paramètre sous
↪  la forme {t[0], t[1], ...,t[n]}
*/
static String enClair (int [] t)
```

2. Rédiger une fonction `indiceDe` qui à partir d'un tableau d'entiers `t` délivre l'indice d'un élément du tableau où la valeur `val` est présente, -1 si `val` est absente de `t`. La spécification de la fonction est la suivante :

```
/**
 * Recherche la première occurrence d'une valeur
 * @param tableau d'entiers dans lequel la recherche est faite
 * @param val la valeur entière recherchée
 * @return l'indice ival de la première occurrence de val ; t[ival] == val si val présente
↪  dans t, -1 sinon
 */
static int indiceDe(int[] t, int val)
```

Exercice 2. Affichage de tableau

On choisit de représenter un ensemble E d'entiers compris entre 0 et n par un tableau `b[0..n]` de booléens : $i \in E$ si et seulement si `b[i]` vaut `true`.

— Rédiger une fonction `enClair` qui à partir d'un tableau de booléens `b` représentant un ensemble E d'entiers tous compris entre 0 et `b.length-1` délivre une chaîne de caractères représentant E sous une forme lisible. Les spécifications de cette fonctions sont les suivantes :

```
/**
 * Construction d'une chaîne de caractères représentant un ensemble d'entiers
 * @param b le tableau de booléen représentant les éléments d'un ensemble E ; i est dans E
↪  si et seulement si b[i] vaut true.
```

```
 * @return une chaîne de caractères représentant E de manière lisible, i.e. l'ensemble des
↪  indices i pour lesquels b[i] vaut true
 */
static String enClair (boolean [] b)
```

Exercice 3. Modification de tableau et appel de fonction
On dispose d'une fonction `indiceDuMax` qui à partir d'un tableau d'entiers ayant au moins un élément, délivre l'indice d'un élément du tableau de valeur maximale. La spécification de cette fonction est la suivante :

```
/**
 * Recherche de l'indice de la valeur maximale contenue dans le tableau d'entiers
 * @param t un tableau d'entiers non vide
 * @return l'indice iMax tel qu'aucun élément de t ne soit strictement supérieur à t[iMax]
 */
static int indiceDuMax(int[] t)
```

— Rédiger une fonction `lePlusGrandALaFin` qui à partir d'un tableau d'entiers `t` délivre un autre tableau d'entiers obtenu à partir de `t` en échangeant la valeur maximale et la valeur d'indice maximal. **On veillera à ce que `t` reste inchangé.** Les spécifications de la fonction sont les suivantes :

```
/**
 * Fonction qui construit un tableau d'entiers identique à celui reçu en paramètre sauf en
↪  au plus 2 places :
 * le dernier élément et une occurrence de la valeur maximale ayant été échangées
 * @param le tableau à analyser
 * @return le tableau identique à t mais avec la valeur maximale à la fin
 */
static int [] lePlusGrandALaFin (int []t)
```

2 Tableaux complexes et calculs

Exercice 4. Tableaux carrés
Un tableau carré d'entiers est un tableau de tableaux `m` tels que tous les `m[i].length` ont la même valeur qui, en outre, est égale à `m.length`.

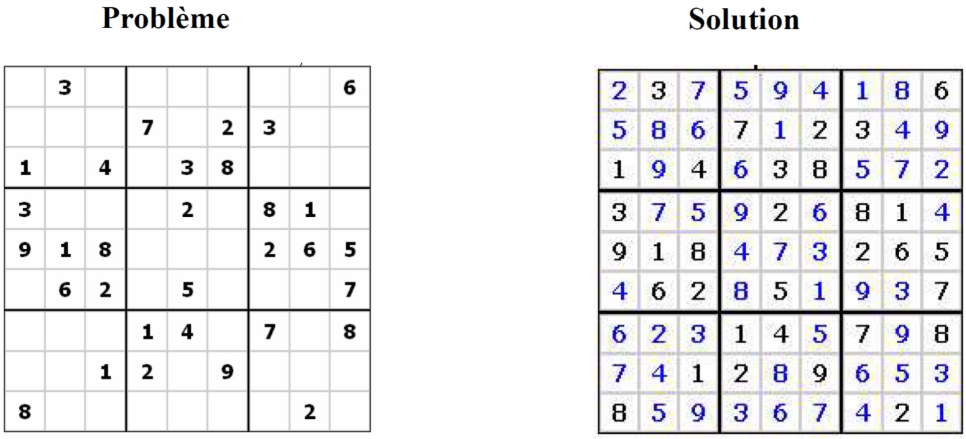
Pour les questions suivantes vous devrez *Concevoir, i.e.* définir la signature de chaque fonction et rédiger la spécification (JavaDoc) associée, avant de *rédiger* le corps de la fonction.

1. Concevoir et rédiger une fonction `estDiagonale` qui indique si une matrice `m`, représentée par un tableau carré, est diagonale. On rappelle que dans une matrice diagonale `M`, seuls les éléments $M_{i,i}$ peuvent être différents de 0.
2. Concevoir et rédiger une fonction transposée qui à partir d'un tableau carré `m` délivre un tableau carré représentant la matrice transposée de celle représentée par `m`. On rappelle que la transposée d'une matrice `M` est une matrice `T` telle que $M_{i,j} = T_{j,i}$.

3. Concevoir et rédiger une fonction `transpose` qui à partir d'un tableau carré `m` représentant une matrice M, le modifie de telle sorte qu'il représente la transposée de M.

3 Préparation du TP2 - Sudoku

Une grille de Sudoku (*cf.* figure 3) se définit comme suit (d'après Wikipédia) : « La grille de jeu est un carré de neuf cases de côté, subdivisé en autant de carrés identiques, appelés régions (voir figure). La règle du jeu est simple : chaque ligne, colonne et région ne doit contenir qu'une seule fois tous les chiffres de un à neuf. Formulé autrement, chacun de ces ensembles doit contenir tous les chiffres de un à neuf. »



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

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

FIGURE 1 – Exemple de grille de Sudoku

Pour modéliser un plateau de Sudoku, on utilisera un tableau d'entiers à 2 dimensions, dont les éléments peuvent valoir 0 (la cellule n'a pas encore de valeur fixée), 1, . . . , 9. Celle-ci sera donc déclarée et instanciée comme suit :

```
1 static final int n = 3 ; // taille des régions
2 int [][] m = new int [n*n][n*n] ;
```

Exercice 5. Bornes de parcours

— À partir des indices `i` et `j` désignant une case d'un plateau, calculer les indices `iCoin` et `jCoin` du coin supérieur gauche de la région contenant la case <i,j>.