TD n°2 – Types, Création de classes, Tableaux

Rappels :

- Vous devez organiser correctement le code réalisé en TD.
- A chaque nouvel énoncé de TD, vous devez créer un nouveau projet (*Java Project*) sous Eclipse portant le numéro du TD concerné (ici TD2).
- Vous créez ensuite un package pour chaque exercice à l'intérieur du projet.

I. Types primitifs

Pour chaque ligne, indiquez si l'initialisation est autorisée. Si oui indiquez la valeur de la variable.

```
double a = 3 + 2;
int b = 3 + 2.5;
double d = (3 / 2);
double e = (3.0 / 2);
int f = "124";
String g = 5.78;
String h = ""+5.78;
String i = "Chat" + "Pitre";
char j = 'A';
boolean k = (a <= 5.78);
boolean l = k && !(d < e);
```

II. Tableaux à 1 dimension

Le but de cet exercice est de réaliser une classe `EnsembleEntierBorne` permettant de manipuler des ensembles d'entiers de taille bornée.

1) **Attributs :** Chaque instance de `EnsembleEntierBorne` est définie par 2 attributs :
   a. une constante entière `MAXIMUM` qui ne peut être initialisée qu'une seule fois, et
   b. un tableau de booléens où chaque case indique si oui ou non l'entier (correspondant à l'indice) est contenu dans l'ensemble.
2) **Constructeurs :** l'unique constructeur prend en paramètre un entier. L'ensemble ne pourra contenir que des entiers compris entre 0 (inclus) et cet entier (inclus). *Le tableau a donc une case de plus que cette valeur.*
3) **Méthodes :**
   a. `add(…)` Cette procédure prend un entier en paramètre et le rajoute à l'ensemble (si l'entier est déjà dans l'ensemble ou ne peut être contenu dans l'ensemble, la procédure est sans effet).
   b. `remove(…)` Comme pour la procédure `add`, mais cette fois-ci il s'agit d'enlever un élément.
   c. `doesContain(…)` Cette fonction prend un entier en paramètre et renvoie vrai si et seulement si l'entier est dans l'ensemble.
   d. `toString()` Cette fonction retourne une chaine de caractères représentant l'ensemble sous forme `{2,4,6,7,99}` (on peut éventuellement laisser une virgule en trop dans une première version, sinon utiliser la fonction `substring` de la classe `String` afin de supprimer la dernière virgule inutile).
   e. `intersect(…)` Cette fonction prend un `EnsembleEntierBorne` en paramètre et renvoie l'ensemble correspondant à l'intersection entre l'ensemble courant et celui entré en paramètre.

4) **Main :** Dans une classe `TestEnsemble`, instancier des objets `EnsembleEntierBorne` et tester les méthodes développées.
5) En utilisant la description des ensembles d'entiers bornés de l'exercice précédent (constructeur + méthodes), ajouter dans une fonction associée à la classe `TestEnsemble` (**qui ne dépend pas des instances de la classe**) le code Java correspondant à l'algorithme suivant :

```
variables
        maxim : entier
        premiers : ensemble d'entiers entre 0 et maxim
        i, j : 0..maxim
début
        pour i de 2 à maxim faire
                inclure i dans premiers
        finpour
        pour i de 2 à maxim faire
                si i appartient à premiers alors
                        pour j de 2i à maxim par pas de i faire
                                enlever j de premiers
                        finpour
                finsi
        finpour
        afficher premiers
fin
```

Que fait cet algorithme ? Regarder ce qu'il se passe sur différents exemples.
Modifier le `main` pour appeler cette fonction et la tester sur différents exemples.

III. Utilitaires sur les tableaux

Services de la classe Arrays

https://docs.oracle.com/javase/8/docs/api/java/util/Arrays.html

La classe `Arrays` du package `java.util` offre des services pour les tableaux (tri, recherche, remplissage, égalité, ...) La liste des méthodes définies dans la classe `Arrays` sont `static` et s'appellent donc en utilisant le nom de la classe suivi de celui de la méthode : `Arrays.sort(Object[] a).`

Soient les tableaux instanciés suivants :

```
int[] T1 = new int[10] ;
String[] T2 = new String[7] ;
int[] T3 = {0, 6, 2, 4, 3} ;
String[] T4={«bleu»,«rouge»,«blanc»,«vert»,«mauve»,«indigo»};
```

Consulter les nombreuses méthodes disponibles dans la classe `Arrays` et trouver les méthodes permettant de faire les actions suivantes :

- Remplir T1 avec l'entier 5 dans toutes les cases.
- Remplir T2 avec la chaîne de caractères « bleu » dans les cases d'indices 1 et 2.
- Afficher les tableaux
- Trier T3 dans l'ordre croissant

- Trier T4 dans l'ordre alphabétique
- Vérifier si T1 et T3 sont égaux
- Copier les 5 premiers éléments de T4 dans un nouveau tableau T5
- Dupliquer T4 dans un tableau T6 et vérifier s'ils sont égaux

Créer une classe avec une méthode main afin de tester les méthodes demandées ci-dessus.

IV. S'il vous reste du temps : tableaux à 2 dimensions

Un *carré magique* est une matrice carrée n x n contenant tous les entiers entre 1 et n^2, et telle que la somme des entiers de chaque ligne, de chaque colonne, et des deux diagonales sont identiques.

Dans cet exercice, nous allons créer des carrés magiques de taille impaire (n impair).

Exemples :

Carrés magiques d'ordre 3 et 5

4	9	2
3	5	7
8	1	6

11	24	7	20	3
4	12	25	8	16
17	5	13	21	9
10	18	1	14	22
23	6	19	2	15

L'algorithme pour générer un carré magique de taille n impaire est le suivant :

- On place le 1 dans la case située une ligne en dessous de la case centrale (qui existe toujours vu que n est impair).
- Lorsque l'on a placé l'entier x dans la case (i, j), on place l'entier $x + 1$ dans la case $(i + 1, j + 1)$, mais :
  - Si un indice devient égal à n, il revient à 0
  - Tant que l'on tombe sur une case déjà occupée, par exemple (l, k), on essaie de placer le nombre en $(l + 1, k - 1)$ (attention encore aux bornes de l et k !)

Notez que cet algorithme suppose que le carré est représenté par un tableau Java à 2 dimensions et donc que les indices commencent à 0.

Ecrire le code qui permet d'implémenter cet algorithme et créer des carrés magiques de taille n.

Comment est organisé le code ? De quoi dépend la méthode qui créer les carrés ? Justifier vos choix d'implémentation.