TD/TP 3 : Champs et méthodes statique

a- Écrire une classe **Ballon** contenant une variable : ***couleur*** de type String. La classe aura un constructeur avec un argument et un sans argument. La classe possèdera deux méthodes **afficheBallon**.

b- On souhaite maintenant affecter un numéro à chaque ballon. Compléter la classe précédente de manière à pouvoir attribuer un numéro unique à chaque nouveau ballon créé (1 au premier, 2 au suivant, . . .). On ne cherchera pas à réutiliser des numéros déjà affectés. On ajoutera à la classe une méthode **getIdent** renvoyant le numéro attribué au ballon et une méthode de **getIdentMax** fournissant le numéro du dernier ballon créé.

c- Écrire un petit programme d'essai en affichant bien le numéro de chaque ballon et le nombre de ballons créés.

d- Malheureusement, il est possible que certains ballons disparaissent (par exemple lorsque le ramasse-miettes détecte qu'un ballon n'est plus référencé). Compléter la classe précédente en ajoutant une variable permettant de savoir à peu près combien de ballons sont présents dans le programme. Ajoutez la ou les méthodes nécessaires à ce comptage.

Remarque : Pour tester votre code, il faudra faire en sorte que certains objets soient candidat au ramasse-miettes. Il n'est pas possible de contrôler le lancement du **Garbage Collector**, mais on peut indiquer à la **JVM** que l'on souhaite appeler le ramasse-miettes ce qui se fait par l'instruction `System.gc();`

TD/TP 4 : Les tableaux

1. Utilitaire

a. Écrire une classe **Utilitaire** :

Utilitaire
+ genere (int n) : int [] + somme (int[] t1, int[] t2) : int [] + affiche (int[] t) : void

Cette classe contient les méthodes suivantes :

- ✓ une méthode ***genere*** et qui fournit en retour un tableau des n premiers nombres impairs, la valeur de n est donnée en argument.
- ✓ une méthode ***somme*** qui reçoit en argument 2 tableaux d'entiers de même taille et qui fournit en retour un tableau représentant la somme des deux tableaux.
- ✓ une méthode **affiche** qui affiche un tableau d'entiers reçu en argument.

b. Écrire une classe **TestUtilitaire** pour tester les fonctionnalités de la classe **Utilitaire**

2. Matrice

a. Ecrire une classe définissant le concept de matrice carré avec un constructeur à un seul argument.

Matrice
- m : double[][]
+ Matrice (double[] m[]) + multiplie (double x) : void + somme (Matrice Mat) : Matrice + afficher () : void

Cette classe contient les méthodes suivantes :

- ✓ ***multiplie*** qui prend en argument un réel et qui multiplie tous les éléments de la matrice par ce réel
- ✓ ***somme*** qui prend en argument une matrice de même dimension et de retourner la somme des 2 matrices.
- ✓ une méthode ***affiche*** sans argument qui affiche tous les éléments de la matrice à l'écran.

b. Écrire une classe **TestMatrice** pour tester les fonctionnalités de la classe **Matrice**.

TD/TP 5 : Les chaines de caractères

Date

1. Écrire une classe **Date** susceptible de représenter la date sous forme de 3 entiers ***jour***, ***mois*** et ***annee*** et qui possède :

- ✓ un constructeur sans paramètre (par défaut 1 janvier 1900),
- ✓ un constructeur avec trois paramètres (jour, mois, année)
- ✓ et un constructeur avec un seul paramètre de type String

(Par exemple : 27 fevrier 2008 ou 27.fevrier.2008 ou 27-fevrier-2008 ou 27/2/2008 …)

- ✓ La redéfinition de la méthode toString pour afficher une date sous forme de chaine de caractères.

(Par exemple : 29.4.2002 sera afficher comme « 24-avr-2002 »)

2. Écrire une classe de test qui teste la classe Date