Introduction :

Le but de ce TP est de réaliser une implémentation d'un calendrier grégorien au format JJ//MM//AAAA à partir du 1/1/1900, le programme doit faire défiler le calendrier pendant un temps suffisamment grand pour pouvoir comparer les performances des deux structures proposées.

L'affichage étant très coûteux en temps, on ne prendra pas en compte l'affichage dans la mesure du temps d'exécution.

Exercice 1

Nous allons pour cet exercice, écrire les méthodes pour implémenter la structure à 3 entiers.
Nous déclarons donc 3 variables entières qui représentent le jour le mois et l'année :

```
int jour ;
int mois ;
int annee ;
```

Le but est d'abord de faire le test pour passer de 1900 à 2000 (voir le programme CalendrierEx1.java) puis de 1900 à 1902.
L'affichage n'est pas comptabilisé dans le temps d'exécution et sert uniquement à des fins de test.

```
da240318@c-di-720-12:~/TP/dossier_exam java CalendrierEx1
1006710 nanosecondes
annee : 2000
jour : 1
mois : 1
```

```
da240318@c-di-720-12:~/TP/dossier_exam java CalendrierEx1
53735 nanosecondes
annee : 1902
jour : 1
mois : 1
```

Le programme semble fonctionner et passer de 1900 à 2000 puisque l'année 2000 est affichée, on remarque que le programme prend 1006710 nanosecondes = 1006 ms pour passer de 1900 à 2000.

Nous avons également codé les méthodes -
`estBissextile` (qui retourne si l'année est bissextile)
`dureeAnnee` (qui retourne le nombre de jours de l'année)
`dureeMois` (qui retourne le nombre de jours du mois)

Exercice 2

Pour cet exercice, le but est de réaliser un calendrier, mais cette fois ci, la date ne sera stockée que dans un seul entier qui devra être décomposé en jour/mois/année

L'algorithme pour passer d'un nombre de jours à une date nous est donné, nous allons coder celui-ci dans une méthode `jour2Date` qui prend en entrée un nombre de jours et s'occupe de le convertir en une date au format JJ//MM//AAAA ; cette date est ensuite retournée dans un tableau `tabDate` de 3 cases, une pour le jour, une pour le mois et une pour l'année

Dans notre code main (CalendrierEx2.java), on instancie une variable entière `nbJour` que l'on initialise à 0.

Pour notre test, nous allons l'incrémenter de 1 jusqu'à ce que celle-ci atteigne 36525 c'est à dire atteindre le 1er janvier 2000. Nous allons ensuite tester en passant de 1900 à 1902.

Voici une portion de code

```
tabDate = new int[3];
nbJour = 1;

long debut = System.nanoTime();


// On teste le passage de 1900 à 2000

while (nbJour < 36525) //36525 jours pour passer du  1/1/1900 au
1/1/2000
{
     nbJour++;
}

tabDate = Outils.jour2Date(nbJour);

System.out.println("le " + nbJour + " ème jour correspond au " +
tabDate[0] +    '/' + tabDate[1] + '/' + tabDate[2]); //retourne
la date à partir du jour

long fin = System.nanoTime();
System.out.println((fin-debut) + " nanosecondes");

}
```

Ainsi qu'une image de l'exécution du programme.

```
da240318@c-di-720-12:~/TP/dossier_exam java CalendrierEx2
869599 nanosecondes
le 36525 ème jour correspond au 1/1/2000
```

```
da240318@c-di-720-12:~/TP/dossier_exam java CalendrierEx2
507839 nanosecondes
le 1000 ème jour correspond au 27/9/1902
```

On remarque que le programme de l'exercice 2 prend moins de temps que celui de l'exercice 1, pour le passage de 1900 à 2000 869 ms < 1006ms.

Cependant, le programme 1 prend moins de temps pour le passage de 1900 à 1902 que le programme 2. (53 ms < 507 ms)

Le programme 2 est donc plus avantageux à utiliser pour de grandes durées.

Le choix du programme dépend donc de l'exigence de l'utilisateur.