Exercice 0

-Quelles sont les méthodes que vous connaissez dans la classe Thread ?
-Quels sont les différents états possibles pour une thread ?
-comment bloquer l'usage d'un objet ?

Exercice 1

```
class A extends Thread {
    public void run()
      {for (int i=0; i<8; i++)
        {System.out.print("A"+ i +" ");
          try {sleep(100);}
          catch(InterruptedException e){};}}}
class B extends Thread {
    public void run()
      {for (int i=0; i<8; i++)
        {System.out.print("B"+ i +" ");
          try {sleep(200);}
          catch(InterruptedException e){};}}}
class Test {
    public static void main (String argv[])
      {new A().start();        new B().start();}}
```

Quel résultat peut être attendu de l'exécution de ce code ?

Exercice 2

Compte à rebours pour le lancement d'une fusée
On veut programmer la simulation du lancement d'une fusée, entre le moment où l'on déclenche le compte à rebours et le moment où on allume les réacteurs.
Deux équipes participent à cette simulation: l'équipe météo et l'équipe sécurité. Quand le compte à rebours est déclenché, les deux équipes entrent dans une phase de réflexion qui peut les amener à retarder la fin du compte à rebours pour reporter le lancement de la fusée. Les deux équipes travaillent indépendamment en parallèle l'une de l'autre.
On utilisera une classe `CompteARebours`, caractérisée par une `dateDébut` qui correspond au lancement du compte à rebours et une `dateFin` qui correspond à la date de fin du compte à rebours (cette date de fin pourra être décalée quand une équipe prend du retard). Le décompte du temps restant se fait par rapport à cette date de fin.
On utilisera une classe `Equipe`. Les équipes ont accès à l'unique compte à rebours qui est le même pour toutes les équipes et est stocké dans `cReb`. Lors d'une reflexion, l'équipe prend un certain temps en secondes (int) pour celle-ci, ce temps est noté `tReflexion`; lors d'une réflexion un temps de retard du compte à rebours peut être nécessaire; ce temps est donné en secondes et est noté `tRetard`.
On utilisera une classe `simulation` qui permettra de tester notre simulateur.

1) La classe `CompteARebours` est composée de
-un constructeur à un paramètre de type long qui lance le `compteARebours` pour le délai en millisecondes passé en paramètres
-une méthode `getDateFin` qui retourne la date de fin (en `long`)
-une méthode `setDateFinLong` qui permet de modifier la date de fin par une nouvelle valeur donnée en long
-une méthode `majDateFinLong` qui permet d'ajouter un délai en milisecondes à la date de fin (cette méthode utilise les deux méthodes précédentes)

-une méthode `getTempsRestant` qui retourne en milisecondes le temps restant avant la fin du compte à rebours

Programmer la classe `compteARebours` (champs et méthodes). On fera attention au fait qu'un compte à rebours peut être utilisé par plusieurs thread, donc on évitera les problèmes d'accès concurrents.

Rappel: L'importation de la classe `Date` se fait avec "`import java.util.Date`".

La classe `Date` stocke et gère l'heure en miliseconde, avec un type `long`; l'heure 0 est le 1/1/1970 à 0h.

Elle `possède les méthodes suivantes:`

-constructeurs: `Date()` qui construit l'objet en l'instantiant à la date courante

`Date(long X)` qui construit l'objet en l'instantiant à la date X

-autres méthodes:

`long getTime()` qui retourne l'heure

`setTime(long X)` qui met la date à x

2) La classe `Equipe` est composée de:

-un constructeur à un paramètre de type `String` qui instancie l'équipe avec un nom (cf. note **a)**.-une méthode `réflexion` qui gère une réflexion de l'équipe. Un temps de réflexion `tReflexion` est d'abord affecté à l'équipe de façon aléatoire (entre 0 et 5 secondes; cf note **b);** durant ce temps de réflexion, le thread n'a pas besoin de ressources, un temps de retard `tRetard` (en secondes) est affecté à l'issue de la réflexion à l'équipe de façon aléatoire (entre 0 et 5 secondes).

-une méthode `run`() qui lance une réflexion, décale le compte à rebours pour prendre en compte le retard du à la réflexion, écrit le nom de l'équipe, le temps de réflexion utilisé et le retard pris.

Programmer la classe `Equipe` (champs, variable de classe et méthodes). On fera attention au fait que la méthode sleep peut générer une `InterruptedException`, qui devra être gérée dans la méthode `run`.

Notes: a) la classe `thread` contient un constructeur à un paramètre de type `String` qui permet d'associer un nom à un thread. `getName()` permet d'obtenir le nom de la thread.

b) la méthode de classe `random()` de la classe `math` fournit un double au hasard entre 0 et 1.

c) Pour s'assurer du blocage du flux de sortie sur écran, on peut synchroniser "system.out".

3) La classe `Simulation`

utilise un compte à rebours de 10 secondes;

crée les deux équipes de noms "meteo" et "sécurité" et les démarre;

regarde toute les secondes si le temps restant est positif, et si oui affiche le temps restant, et si non affiche décollage et fini.

4) Modifier les programmes pour permettre autant d'interventions que possible par les équipes tant que le compte à rebours n'est pas fini.

5) a)Donner trois exemples de résultats affichés. Pour chacun des exemples, on indiquera le nom de la thread et les valeurs obtenues par la randomisation.

b) modifier, si ce n'est fait, votre programme pour assurer la bonne gestion de l exception `InterruptedException` qui peut être lancée lors d'un `sleep`.