Université de Bretagne Sud
CONCURRENCE
Liste circulaire de taches

Luc.Courtrai@univ-ubs.fr

1 Barrière

Une barrière bloque un ensemble de threads tant que tous les threads ne sont pas arrivés à cette barrière.

Ecrire en Java la classe Barrier d'après le squelette ci-après

```
public class Barrier {
   public Barrier(int nbThreads)
   public synchronized  void synchr()
...
```

Le constructeur reçoit en argument *nbThreads* le nombre de threads à synchroniser sur la barrière. La méthode *synchr()* signale à la barrière l'arrivée du thread courant. Les *nbThreads* -1 premiers threads arrivés à la barrière sont bloqués à l'appel de *synchr* et le dernier thread débloque l'ensemble.

La barrière peut être utilisée plusieurs fois pour resynchroniser le même ensemble de threads.

Exemple d'utilisation :

```
class Test extends Thread {
  private Barrier bar;
  private String mot;
  public Test(Barrier bar,String mot){
   this.bar=bar;
   this.mot=mot;
  }
  public void run() {
    while (true) {
      bar.synchr();
      System.out.println(mot);
    }

  }
}

public class TestBarriere  {
  public static void main(String args[]){
    final int nbT=3;
    Barrier unBar= new Barrier(nbT);

    for(int i=0;i < nbT;i++){
      Test th=new Test(unBar,String.valueOf(i));
      th.start();
    }
  }
}
```

Une exécution possible :

```
1
2
0
0
1
2
1
0
2
...
```

2 Tâche cyclique

Écrire le programme Java SuiteMot qui, affiche en boucle une suite de mots passée en argument du programme. Chaque mot est pris en charge par un thread différent pour l'affichage.

Voici un exemple d'exécution

```
>java SuiteMot   cet exercice est trop cool .
cet
exercice
est
trop
cool
.
cet
exercice
est
trop
cool
.
cet
exercice
...
```

Dans cet exemple, avec 6 threads spécifiques : t1,t2,t3,t4,t5,t6, l'ordonnancement est le suivant : t1, t2, t3, t4, t5, t6, t1, t2, t3, t4, t5, t6, t1, t2, t3, t4, t5, t6 ...

La solution consiste à construire une liste (cyclique) des threads (les t1,t2, t3, t4, t5, t6). Chaque thread doit connaître le thread suivant (pour implanter la liste). Les threads sont au départ endormis (arrêtés). L'algorithme d'un thread est alors : effectuer sa tâche (afficher son mot), réveiller le suivant, s'endormir ....(ceci en boucle). Le programme principal doit après avoir créé tous les threads, réveiller le premier.

Faites bien attention au démarrage du système (il faut que tous les threads soient bien endormis avant de réveiller le premier).

Le programme de s'arrête pas.