Laboratoire 7

Les fils d'exécutions, exclusion mutuelle et synchronisation

Objectifs d'apprentissage

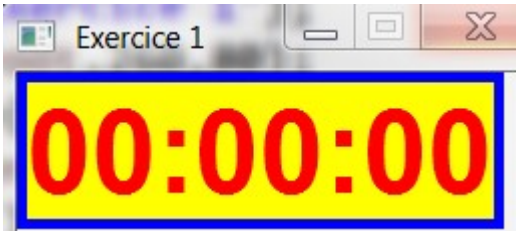
- créer des fils d'exécution
- programmer des méthodes en exclusion mutuelle (synchronized)
- synchroniser des fils d'exécution (notify() et wait())

Exercice 1

Il s'agit de programmer une application qui affiche l'heure en créant un objet *Thread* qui met à jour une étiquette (Label) à chaque seconde.

1- Créer une classe **Horloge, sous classe de Application et qui implémente Runnable**
Celle classe contient une variable d'instance *Label lblHorloge* où sera affichée l'heure sous le format heure : minute : seconde.

- Reproduire l'interface graphique suivante:



- Dans la méthode *start*, afficher le nom du fil d'exécution courant. Bien comprendre de quel fil il s'agit.
- Démarrer la mise à jour continue de l'étiquette, en forçant l'exécution de la méthode *run* de la classe *Horloge.* Utiliser l'instruction : *new Thread(this).start();* Bien comprendre la référence à *this* dans ce cas-ci
- La méthode *run(),* Afficher le nom du fil courant. S'agit-il du même fil d'exécution que dans la méthode start()?
  Dans une boucle infinie (*while (true)*), la méthode *run()* met à jour *lblHorloge* à chaque seconde (utilisez Thread.sleep(1000))
  Utilisez l'instruction: ***LocalTime tempsActuel = LocalTime.now();*** pour obtenir l'heure courante. Utilisez les méthodes *getHour(), getMinute() et getSecond()* sur l'objet *tempsActuel* pour récupérer l'heure sous forme heure, minute et seconde. Utilisez la méthode **format** de *DecimalFormat* ou *String* pour définir le bon format.
  Voici un exemple d'exécution :

Exercice 2

Vous devez implanter les deux classes *Compteur* et *TestThread* comme suit :

La classe Compteur contient :

- Une variable d'instance intCompteur
- Un constructeur qui initialise la valeur *intCompteur* à la valeur reçue en argument
- Deux méthodes synchronisées *inc()* et *dec()* qui, respectivement incrémente et décrémente la valeur *intCompteur.* Ces méthodes doivent aussi afficher la valeur de *intCompteur* en précisant le nom de la méthode.

La classe TestThread implémente l'interface *Runnable* et contient :

- Une variable d'instance cpt de type *Compteur*
- Une variable d'instance *bInc* de type booléen. Elle sera vraie pour une incrémentation et fausse dans le cas d'une décrémentation.
- Un constructeur qui initialise les deux variables d'instance à des valeurs reçues en argument.
- Le constructeur crée et lance le thread en cours.
- Une méthode *run(),* une boucle infinie, qui en fonction de la valeur du booléen *bInc* appelle la bonne méthode de *Compteur* et suspend le thread en cours pour une durée aléatoire. Utiliser l'instruction :*Thread.sleep((int)Math.random()*1000);* pour faire dormir un thread jusqu'à 1 sec.
- Une méthode main() qui crée un objet compteur *monCompteur* dont la valeur initiale est 1 puis Lance deux threads concurrents, deux objets *thread1* et *thread2* de la classe *TestThread*, qui accèdent tous deux à l'objet *monCompteur*. Thread1 incrémente l'objet compteur alors que thread2 le décrémente.

Exercice 3

Reprendre l'exemple du producteur/consommateur vu en classe, en considérant un tampon de taille supérieure à un. Cette taille doit être initialisée dans le constructeur à l'aide d'une variable reçue en argument. Les contraintes de synchronisation restent les mêmes : on ne peut consommer à partir d'un tampon vide et on ne peut produire dans un tampon plein.