TD PAC - Java n° 9

Processus parallèles (Threads)

Exercice 1 Observation du comportement

Cet exercice a pour objectif d'étudier le comportement de l'ordonnateur. Pour cela, vous devez réaliser deux classes :

1. `AfficherUnNombre` qui hérite de `Thread` qui affiche n fois un même nombre.
2. `Test` qui lance 5 threads `AfficherUnNombre` avec les nombres 1, 2, 3, 4, 5.
3. Dans un premier temps, chaque thread affiche 50 fois son nombre
4. Dans un deuxième temps, ils affichent 10000 fois leur nombre
5. Quelle conclusion tirez-vous de ces tests ?

Exercice 2 Compteurs...

Cet exercice est basé sur une classe `Compteur`, qui a un nom (par exemple "Toto") et qui compte (et affiche) les entiers de 1 à n . Il marque une pause aléatoire entre chaque nombre (de 0 à 500 millisecondes par exemple).

Un compteur affiche chaque nombre précédé par son identité (Toto affichera par exemple, "Toto : 3") et il affiche un message du type "*** Toto a fini de compter jusqu'à 10 en xxx millisecondes" quand il a fini.

Rappel : La fonction `long System.currentTimeMillis()` retourne la date courante en milliseconde.

1. Écrivez la classe compteur et testez-la en lançant 10 compteurs qui comptent jusqu'à 10.
 - La classe `Compteur` hérite de `Thread`.
 - La classe `Compteur` n'hérite pas de `Thread` mais implémente `Runnable`.
2. Modifiez la classe `Compteur` pour que chaque compteur affiche son ordre d'arrivée : le message de fin est du type : "Toto a fini de compter jusqu'à 10 en position 3".
3. Quand tous les compteurs auront fini, c'est maintenant le programme principal qui affiche "Tout le monde a fini", suivi de l'ordre d'arrivée des compteurs.
4. C'est toujours le programme principal qui affiche l'ordre d'arrivée, mais pour cette question, il affiche l'ordre d'arrivée au fur et à mesure de l'arrêt des threads, sans attendre qu'ils aient tous fini.
5. Modifiez votre code pour que les différentes instances de `Compteur` partagent une unique variable. Le lancement de 10 compteurs comptant jusqu'à 10 doit permettre d'incrémenter la variable de 1 à 100.
 - Quel problème peut se présenter (il est nécessaire de réduire le temps entre les affichages à 50 millisecondes maximum)
 - Modifier votre programme pour contourner ce problème ?

Exercice 3 Mobiles.

1. On veut “promener” des mobiles sur une fenêtre. Pour cela, on dispose du code suivant :

```
class UnMobile extends JPanel implements Runnable
{
    int largeur, hauteur,debut,debutDessin;
    final int pas=10, temps=200,cote=40;

    UnMobile(int l, int h) {
        super();
        largeur=l;
        hauteur=h;
        setSize(l,h);
    }

    public void run () {
        for (debutDessin=0;debutDessin<largeur-pas;debutDessin+=pas) {
            repaint();
            try {
                Thread.sleep(temps);
            }
            catch (InterruptedException e){
                e.printStackTrace();
            }
        }
    };

    public void paintComponent (Graphics cg) {
        super.paintComponent (cg);
        cg.fillRect(debutDessin, hauteur/2, cote, cote);
    }
}
```

A chaque création d’un mobile, on crée un JPanel de dimension *largeurxhauteur*, dans laquelle se déplace un mobile de taille *cotexcote* Ecrire une classe *UneFenetre*, ayant un attribut *UnMobile*. Son constructeur ajoutera le mobile à la fenêtre, créera un thread avec le mobile, affichera la fenêtre et lancera le thread. Tester cette classe.

2. Modifier le code de façon à ce que le mobile ne s’arrête pas au bord de la fenêtre mais reparte alors dans l’autre sens.
3. On veut maintenant disposer deux mobiles sur la fenêtre. Modifier le code en conséquence.