TD - Thread en JAVA

Equipe pédagogique : Dr. Jihène Tounsi (c) et Marwa Laabidi (TD/TP)

Exercice 1 :

Proposez une application permettant de simuler une course de 1 km entre plusieurs coureurs. Chaque coureur doit afficher un message après avoir franchi 100m. La durée d'un parcours de 100m est une durée aléatoire d'au maximum une seconde.

P.S : (int)(Math.random()*n) → Retourne un entier strictement inférieur à n et supérieur ou égale à zéro.

Exercice 2 :

2.1 Il vous est demandé d'implémenter l'algorithme d'ordonnancement FIFO avec le langage JAVA. Nous allons supposer que tous les processus arrivent au même temps au lancement de l'application. Pour se faire vous devez suivre le schéma suivant :

- La classe « **Processus** » est **une classe objet** ayant comme attribut :
  - Calcul : durée d'exécution du processus dans le processeur.
  - TA : un entier pour calculer le temps d'attente.
  - TR : un entier pour calculer le temps de réponse.
  - Nom : nom du processus

La classe « **Processus** » doit implémenter les méthodes suivantes :

- Un constructeur ayant comme paramètres les attributs (calcul et nom) définis précédemment.
- Des accesseurs (**get**) et des mutateurs (**set**) pour les différents attributs

La classe « **Ordonnanceur** » est **une classe Thread** ayant comme attribut :

- File_prêt : une liste dynamique stockant les processus à ordonnancer (à utiliser ArrayList ou Vector)
- File_Termine : une liste dynamique stockant les processus qui ont terminé leur exécution (à utiliser ArrayList ou Vector)
-
- Horloge : entier calculant l'évolution du temps.

La classe « **Ordonnanceur** » doit implémenter les méthodes suivantes :

- Un constructeur non paramétré.
- Une méthode « ***run*** » définissant le comportement du thread. C'est-à-dire la logique de l'ordonnancement FIFO.

Nous allons supposer que l'initialisation de la file est faite manuellement au niveau du programme principal (examen Mai 2011).

2.2 Proposer une méthode pour l'ordonnancement « Tourniquet » (examen Mai 2012) en rajoutant l'attribut Quantum à la structure de la classe Ordonnanceur.

Exercice 3 :

Il vous est demandé de créer une application qui permet de simuler le fonctionnement d'un Opérateur bancaire. Pour se faire vous devez créer trois classes :

- La classe **Compte** : décrivant les caractéristiques d'un compte ainsi que les opérations qui peuvent être réalisées sur son solde bancaire.
- La classe **Opérateur** qui simule le fonctionnement de l'opérateur bancaire (Retrait ou dépôt d'argent dans un compte)
- La classe **Banque** qui est votre programme initial qui lance plusieurs opérateurs en même temps.