TP n° 4: Classe et Héritage (Suite)

Deuxième Partie

Exercice 3 : Implémentation des interfaces

Dans cet exercice nous allons étudié la classe `Thread` et l'interface `Runnable` qui permettent la programmation concurentielle et la gestion du multithreading en Java.
En consultant la JavaDoc sur la classe `Thread` et l'interface `Runnable`

- Quelle est la méthode de l'interface `Runnable` qui doit être définie ? Étudier son fonctionnement.
- Étudier le fonctionnement des méthodes suivantes :
  - *void start()* (définie dans la classe Thread)
  - *void join()* (définie dans la classe Thread)
  - *void notify()* (définie dans la classe Object)
  - *void wait()* (définie dans la classe Object)
  - *Thread.sleep(int)*
- Expliquer le fonctionnement d'un bloc synchronisé par l'utilisation du mot-clé *synchronized*

Nos voudrons créer deux processus qui s'exécutent simultanément et qui s'envoient des informations à l'aide d'une pile partagée entre eux (mode LIFO).

- Un processus est appelé "émetteur" et le deuxième est appelé "récepteur". Pour que l'émetteur envoie un message au recepteur il doit l'insérer dans la pile, le récepteur doit récupérer ce message en l'enlevant de la pile.
- La pile possède une taille limitée, l'émetteur ne pourra pas remplir la pile si elle saturée. De la même manière le récepteur ne pourra pas récupérer un message si la pile est vide.
- Tant que le récepteur trouve que la pile est vide il doit attendre jusqu'à ce qu'elle soit remplie (au moins un élément) par l'émetteur (l'émetteur **notifie** le récepteur).
- Tant que l'émetteur trouve que la pile est saturée il doit attendre jusqu'à ce qu'elle soit vidée (d'au moins un élément) par le récepteur (le récepteur **notifie** l'émetteur).
- L'émetteur doit envoyer un message toutes les secondes et le récepteur doit lire un message toutes les trois secondes.

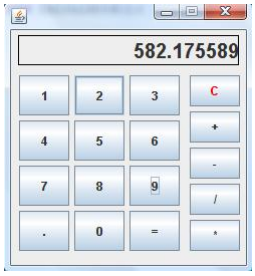
Utiliser les méthodes étudiées ci-dessus pour implémenter ce scénario.

Astuces

- Les deux classes **Emetteur** et **Recepteur** doivent hériter de la classe **Thread**.
- Pour que la pile soit partagée entre les deux classes, définir **Emetteur** et **Recepteur** comme classes ***imbriquées*** dans la classe principale.
- La classe principale contient la méthode **main** dans laquelle les deux processus sont lancées.

Projet : Calculatrice

On voudra rassembler tous les programmes faits en TP1, TP2 et TP3 sur l'implémentation d'une calculatrice décimale et binaire. Le rendu final doit être une interface graphique d'une calculatrice comme le montre la figure suivante :



— Deux modes sont possibles le mode décimal et le mode binaire, pour chaque mode il existe deux types de calculatrice :

1. **type 1 :** Calculatrice basique qui fait les opérations deux à deux.
2. **type 2 :** Calculatrice étendue qui fait plusieurs opérations en une seule ligne (de gauche à droite)

— Les opérations possibles sont : l'addition, la soustraction, la multiplication et la division.
— Pour pouvoir donner l'opération à calculer nous pouvons soit utiliser les boutons de la calculatrice, ou le clavier pour l'écrire sur la zone de texte.
— Un bouton est nécessaire pour initialiser le calcule et effacer la zone de texte.
— On peut aussi convertir un nombre décimal en binaire, ou l'inverse, en passant du mode décimal au mode binaire et vice versa.

Utiliser toutes les notions vues en TP et en cours pour construire ce logiciel. Vous pouvez utiliser :
— L'héritage entre classes **Basique** et **Etendue** ;
— L'implémentation d'une interface **Calculer** ;
— Définition ou redéfinition des méthodes d'addition, de soustraction, etc.
— Surcharge ou abstraction des méthodes de calcul pour les deux modes ;
— Etc.

Un rapport technique du logiciel doit être remis, qui contient :
— Introduction contenant la définition du projet ;
— Modèle adopté pour l'implémentation (classes, interfaces, diagramme, etc.) ;
— Organigrammes et détails de l'implémentation ;
— Rendu final de l'interface avec détails.