Thème 7 - TME

Exercice

Lors de chaque TME, vous devrez créer un répertoire pour chaque exercice, y mettre les fichiers s'y rapportant et soumettre ce répertoire à la fin du TME.

Exercice 10 – Optimisation des accès en lecture/modification

Nous souhaitons mettre en place un système dans lequel 3 classes de processus, des Producteurs, des Lecteurs et des Effaceurs accèdent à un ensemble de données. Cet ensemble n'est pas de taille fixe et est représenté en mémoire par une liste (une instance de la classe `ArrayList`). Nous nous placerons dans un cas simple où les données manipulées sont des entiers.

- les producteurs ajoutent des données, toujours à la fin de la liste ;
- les lecteurs parcourent la liste et affichent les données lues ;
- les effaceurs parcourent la liste pour y rechercher une valeur et la suppriment s'ils la trouvent. Ils affichent un message d'erreur si la valeur n'est pas présente.

Dans un premier temps, *nous ne nous préoccupons pas des problèmes de synchronisation.*

Question 1

Écrivez une classe `EnsembleDonnees` offrant aux 3 classes de processus les méthodes nécessaires à la réalisation de leurs opérations. Nous ferons les hypothèses suivantes :

- un producteur ne fait aucune vérification avant d'ajouter une donnée, il peut donc y avoir des doublons dans la liste ;
- lorsqu'un effaceur doit supprimer un élément présent en plusieurs exemplaires, c'est la *dernière* occurrence de l'élément qui est supprimée. La méthode de suppression doit lever une exception (que vous définirez) lorsque la donnée n'est pas présente.

Question 2

Un lecteur fait 3 affichages de l'ensemble de données, avec une attente temporisée après chaque affichage.

Un producteur fait 3 séries d'ajouts à la liste. Il rend le processeur après chaque ajout (sans temporisation) et exécute une attente temporisée après chaque série. Vous pouvez choisir le nombre de valeurs ajoutées à chaque série (typiquement 4 ou 5), et tirer aléatoirement les valeurs ajoutées.

Un effaceur tire aléatoirement 3 valeurs à supprimer. Il exécute une attente temporisée après chaque suppression.

Écrivez le code correspondant à chacune des classes de processus, ainsi qu'un programme de test qui crée 2 instances de chaque classe.

Lancez l'exécution. Que se passe-t-il ?

On souhaite maintenant synchroniser les accès à l'ensemble de données, en optimisant le parallélisme. Les accès en lecture peuvent se faire en parallèle alors qu'une modification de la liste est exclusive de tout autre accès.

Question 3

Proposez une modification de l'implémentation de la classe `EnsembleDonnees` qui réalise ces contraintes. Que constate-t-on à l'exécution ?

Question 4

Un thread ne peut pas acquérir dans l'ordre le verrou de lecture puis le verrou d'écriture. Modifiez l'implémentation pour garantir la cohérence des données tout en préservant autant que possible le parallélisme.