TP : Rappels sur Java

Partie 1 : La sérialisation

Exercice 1 :

Créez une classe Personne qui implémente l'interface Serializable. La classe Personne a des propriétés telles que le nom, l'âge, et l'adresse.

Sérialisez une instance de la classe Personne dans un fichier, puis désérialisez-la et affichez les informations de la personne.

Exercice 2 :

Créez une application qui permet à l'utilisateur d'ajouter des objets sérialisés de type Product (qui a les propriétés : nom du produit et le prix) à une liste. Enregistrez cette liste dans un fichier. Permettez également à l'utilisateur de charger la liste à partir du fichier et de la consulter.

Exemple de fonctions demandées :

1- loadProductList : Charge la liste de produits à partir du fichier sérialisé.
2- getUserChoice : Affiche le menu principal de l'application et gère les choix de l'utilisateur.
3- addProduct : Ajoute un produit à la liste.
4- displayProductList : Affiche la liste de produits à l'utilisateur.
5- saveProductList : Sérialise la liste de produits et la sauvegarde dans un fichier.

Partie 2 : Les Threads

Exercice 3 :

Écrivez un programme Java qui crée deux threads. Chaque thread doit simplement imprimer les nombres de 1 à 5 à la console, avec une pause d'une seconde entre chaque impression.

Exercice 4 :

Modifiez le programme de l'exercice précédent pour que les deux threads partagent une variable compteur. Utilisez un mécanisme de synchronisation (par exemple, un bloc synchronized) pour garantir que les threads s'exécutent en alternance et que le compteur s'incrémente correctement.

Les objectifs de cet exercice sont les suivants :

1- Les deux threads, le producteur et le consommateur, doivent partager une variable compteur.
2- Utilisez un mécanisme de synchronisation, comme un bloc synchronized, pour garantir que les threads s'exécutent en alternance et que le compteur s'incrémente correctement.

Voici les détails de ce que vous devez faire :

1- Créez une classe SharedCounter qui contiendra la variable compteur partagée. Cette classe devrait avoir une méthode increment() qui incrémente le compteur de manière sécurisée en utilisant la synchronisation.

2- La classe Producer doit prendre en compte la variable SharedCounter et s'incrémenter le compteur chaque fois qu'elle effectue une action (par exemple, la production d'un élément).

3- La classe Consumer doit également utiliser la même instance de SharedCounter et s'incrémenter le compteur chaque fois qu'elle effectue une action (par exemple, la consommation d'un élément).
4- Utilisez un bloc synchronized pour garantir que l'incrémentation du compteur se fait en toute sécurité. Cela signifie que le compteur ne doit être incrémenté que par un thread à la fois.
5- Les deux threads (producteur et consommateur) doivent s'exécuter en alternance. Le producteur effectue une action, puis le consommateur effectue une action, et ainsi de suite.
6- Affichez des messages pour indiquer quand le compteur est incrémenté, à la fois par le producteur et le consommateur.

Lorsque vous avez terminé, exécutez le programme pour observer comment le compteur partagé s'incrémente correctement sans conflits de données entre les threads. Assurez-vous que le mécanisme de synchronisation est correctement mis en place pour éviter les problèmes de concurrence.