TP : *N°8* Module : *POO- JAVA*
Filière : IID1. Prof *: Noreddine GHERABI*

Objectifs :
- ✓ Manipulation des Threads

Exercice 1:

Ecrivez un programme dont le thread principal lance et nomme trois threads. Chaque thread ainsi créé doit effectuer 20 fois les actions suivantes :

- Attendre un temps aléatoire compris entre 0 et 200 ms,
- Afficher son nom
- Le premier Thread le plus prioritaire et le troisième ayant une priorité minimale.
- Le 2ème Thread devra attendre la fin de l'exécution des deux autres threads avant de terminer son exécution.

Exercice 2 :

L'objectif de cet exercice est de réaliser des opérations de deux threads qui partagent un même objet (ici un compte en banque commun).

1. Pour cela vous allez écrire un programme qui comprendra deux classes, **Compte** et **Personne**.

La classe Compte comprend :

- Un attribut privé solde initialisé à 100 représentant le solde courant du compte
- Les getters et les setters
- Une méthode **retirer(float montant)** permettant de retirer un certain montant du compte.

La classe Personne implémente Runnable et représente le comportement que deux personnes (Ahmed et Amal) ont tous les deux. Leur comportement est assez particulier puisque Said et Ali réfléchissent (s'endorment 2 secondes), et en particulier pendant qu'ils effectuent un retrait. Cette classe contient :

- Un attribut compte de type Compte représentant le compte en banque de Ahmed et Amal.
- Une méthode **Retrait(float montant**) permettant à Ahmed et Amal d'effectuer un retrait sur leur compte en banque. Le comportement de cette méthode est le suivant :

La personne voulant effectuer le retrait vérifie le solde, puis s'endort 2 secondes, puis à son réveil effectue le retrait. Le nom de la personne effectuant le retrait doit être signalé.

- Une méthode run() décrivant le comportement de Ahmed et Amal. Il s'agit d'effectuer en boucle (par exemple 10retraits de 200DH).
- En fin la méthode main(String[] args) crée deux threads (Ahmed et Amal) avec le même Runnable (ici de type Personne), puis les lance.

2. Ajouter une classe **SoldeCompte** qui contient un attribut **solde**, une méthode **DeposerArgent(float montant)** qui ajoute un montant dans le compte, et une autre méthode **RetirerArgent** (float montant) pour Retirer un montant. Ces deux méthodes doivent être synchronisées.
    - Si le solde est égal à 0 l'opération « **RetirerArgent »** doit attendre jusqu'à un dépôt d'argent qu'il soit effectué. (utiliser wait(), notify)
3. Créer une classe qui implémente l'interface Runnable et qui a pour tâche déposer et retirer une somme d'argent.
4. Dans la méthode principale créer deux Threads, ces deux Threads effectuent plusieurs opérations de retrait et de dépôt d'argent simultanément. (le programme affiche à chaque fois le solde courant après chaque opération)