JAVA

EXERCICES

Travail sur les Thread

Créez un package **testThread** afin d'y mettre toutes les classes du cours et de l'exercice.

I. Exercices

A. Des threads indépendants

Un "compteur" a un nom (Toto par exemple) et il compte de 1 à n (nombre entier positif quelconque). Il marque une pause aléatoire entre chaque nombre (de 0 à 5000 millisecondes par exemple).

Un compteur affiche chaque nombre (Toto affichera par exemple, "Toto : 3") et il affiche un message du type "*** Toto a fini de compter jusqu'à 10" quand il a fini.

Écrivez la classe compteur et testez-la en lançant plusieurs compteurs qui comptent jusqu'à 10. Voyez celui qui a fini le plus vite.

B. Des threads un peu dépendants

Modifiez la classe **Compteur** pour que chaque compteur affiche son ordre d'arrivée : le message de fin est du type : "Toto a fini de compter jusqu'à 10 en position 3".

C. Un problème d'accès concurrent

Voici 2 classes **Compte** (correspond à un compte bancaire) et **Operation** (thread qui effectue des opérations sur un compte bancaire).

1. *Examinez le code et faites exécuter la classe **Operation**. Constatez le problème : **Operation** effectue des opérations qui devraient laisser le solde du compte inchangé, et pourtant, après un moment, le solde ne reste pas à 0. Expliquez.*
2. *Modifiez le code pour empêcher ce problème.*
3. *Dans le code de **Operation**, remplacez l'opération nulle par 2 opérations **ajouter** et **retirer** qui devraient elles aussi laisser le solde du compte à 0 (elles sont en commentaire dans le code). Lancez l'exécution et constatez le problème. Modifiez le code pour que ça marche.*

II. Annexes

A. Classe Compte :

```
public class Compte {
  private int solde = 0;

  public void ajouter(int somme) {
    solde += somme;
    System.out.print(" ajoute " + somme);
  }

  public void retirer(int somme) {
    solde -= somme;
    System.out.print(" retire " + somme);
  }

  public void operationNulle(int somme) {
    solde += somme;
    System.out.print(" ajoute " + somme);
    solde -= somme;
    System.out.print(" retire " + somme);
  }

  public int getSolde() {
    return solde;
  }
}
```

B. Classe Operation :

```
import java.lang.Thread;

public class Operation extends Thread {
  private Compte compte;

  public Operation(String nom, Compte compte) {
    super(nom);
    this.compte = compte;
  }

   public void run() {
    while (true) {
      int i = (int) (Math.random() * 10000);
      String nom = getName();
      System.out.print(nom);
//        compte.ajouter(i);
//        compte.retirer(i);
      compte.operationNulle(i);
      int solde = compte.getSolde();
      System.out.print(nom);
      if (solde != 0) {
      System.out.println(nom + ":**solde=" + solde);
      System.exit(1);
      }
    }
  }

  public static void main(String[] args) {
    Compte compte = new Compte();
    for (int i = 0; i < 20; i++) {
      Operation operation = new Operation("" + (char)('A' + i), compte);
      operation.start();
    }
  }
}
```