TP4: HELLO

Ce programme est une implémentation Java qui utilise des **sémaphores** pour coordonner des threads et produire la sortie séquentielle **"HELLO"** de manière contrôlée:

Déclaration des importations:

```
package TPsSE;
import java.util.concurrent.Semaphore;
import java.util.logging.Level;
import java.util.logging.Logger;
```

- **package TPsSE** : Définit le package auquel appartient cette classe.
- **import java.util.concurrent.Semaphore** : Permet d'utiliser la classe Semaphore, un mécanisme de synchronisation pour contrôler l'accès aux ressources partagées.
- **import java.util.logging** : Gère les logs en cas d'erreur, comme les exceptions.

Classe principale et initialisation des sémaphores:

```
public class TPHello {
  static Semaphore SH, SE, SL, SO;
```

- **TPHello** : La classe principale contenant le code.
- **static Semaphore SH, SE, SL, SO;** : Définit quatre sémaphores, chacun contrôlant un caractère spécifique :
  - **SH** : Contrôle le début avec le caractère "H".
  - **SE** : Utilisé pour passer au caractère "E".
  - **SL** : Contrôle les deux "L".
  - **SO** : Finalise la séquence avec le "O".

Classe PH (Thread pour "H"):

```
public static class PH extends Thread {
  @Override
  public void run() {
    int i = 0;
```

```
    while (i < 1000) {
      try {
        SH.acquire(); // Attend que SH soit disponible
        System.out.print("H"); // Affiche "H"
        SE.release(); // Libère SE pour permettre "E"
      } catch (InterruptedException ex) {
        Logger.getLogger(TPHello.class.getName()).log(Level.SEVERE, null, ex);
      }
    }
  }
}
```

- **SH.acquire()** : Le thread attend que le sémaphore **SH** soit disponible (valeur > 0).
- **System.out.print("H");** : Affiche la lettre "H".
- **SE.release()** : Libère le sémaphore **SE**, permettant au thread PE (pour "E") de continuer.

Classe PE (Thread pour "E"):

```
public static class PE extends Thread {
  @Override
  public void run() {
    int i = 0;
    while (i < 1000) {
      try {
        SE.acquire(); // Attend que SE soit disponible
        System.out.print("E"); // Affiche "E"
        SL.release(); // Libère SL pour le premier "L"
        SL.release(); // Libère SL pour le deuxième "L"
      } catch (InterruptedException ex) {
        Logger.getLogger(TPHello.class.getName()).log(Level.SEVERE, null, ex);
      }
    }
  }
}
```

- **SE.acquire()** : Attend que le thread PH (pour "H") libère **SE**.

- **SL.release() deux fois** : Permet de gérer les deux occurrences de "L" (chaque SL.release() correspond à un caractère "L").

Classe PL (Thread pour "L"):

```
public static class PL extends Thread {
  @Override
  public void run() {
    int i = 0;
    while (i < 1000) {
      try {
        SL.acquire(); // Attend que SL soit libéré
        System.out.print("L"); // Affiche "L"
        SO.release(); // Libère SO pour le prochain thread
      } catch (InterruptedException ex) {
        Logger.getLogger(TPHello.class.getName()).log(Level.SEVERE, null, ex);
      }
    }
  }
}
```

- **SL.acquire()** : Le thread attend que le sémaphore **SL** soit libéré deux fois (par le thread PE).
- **SO.release()** : Libère **SO**, qui est utilisé pour afficher "O".

Classe PO (Thread pour "O"):

```
public static class PO extends Thread {
  @Override
  public void run() {
    int i = 0;
    while (i < 1000) {
      try {
        SO.acquire(); // Attend que SO soit libéré pour le premier "L"
        SO.acquire(); // Attend que SO soit libéré pour le deuxième "L"
        System.out.println("O"); // Affiche "O" et passe à la ligne
        SH.release(); // Relance SH pour un nouveau cycle
      } catch (InterruptedException ex) {
        Logger.getLogger(TPHello.class.getName()).log(Level.SEVERE, null, ex);
```

```
        }
      }
    }
}
```

- **SO.acquire() deux fois** : Attend que les deux "L" soient affichés avant d'afficher "O".
- **SH.release()** : Relance le cycle en permettant au thread PH de recommencer.

Méthode main:

```
public static void main(String[] args) {
   SH = new Semaphore(1);
   SE = new Semaphore(0);
   SL = new Semaphore(0);
   SO = new Semaphore(0);

   PH ph = new PH();
   PE pe = new PE();
   PL pl = new PL();
   PO po = new PO();

   po.start();
   pl.start();
   pe.start();
   ph.start();
}
```

- **Initialisation des sémaphores** :
  - **SH = 1** : Permet à "H" de commencer immédiatement.
  - Les autres sémaphores sont initialisés à **0**, bloquant les threads respectifs jusqu'à ce qu'ils soient libérés.
- **Création des threads** : Les threads pour chaque lettre sont créés et démarrés.

Résumé de la synchronisation:

1. **"H"** démarre grâce à **SH = 1**, puis libère **SE**.

2. **"E"** démarre après **SE.release()**, libère **SL** deux fois pour gérer deux "L".
3. Chaque "L" est géré indépendamment, et chaque affichage libère **SO**.
4. Après les deux "L", **"O"** est affiché, libérant **SH** pour recommencer.

La sortie est donc :

HELLO
HELLO
HELLO
...
...
1000 FOIS

ENSEIGNANT: RAYANE KHOULOUFI.

reyankheloufi@icloud.com **:EMAIL**