**TP : Java – Processus légers**

1. **La classe Cpt.**
   1. Le nombre maximum de processus légers Co-existants au cours de l’exécution du Cpt est : 3
      * 1. Le thread main créé lors de l’exécution de la classe Cpt.
        2. Les deux threads Thr1 et Thr2 créé lors de l’appel de la méthode   
           « start () » pour chacun d’eux.
   2. Le nombre d’incrémentation de la variable global \_cpt, par chaque processus léger est 100000.
   3. A la fin de l’exécution du code, la valeur affichée est variable.
   4. On a deux threads Thr1 et Thr2 qui gèrent une variable commune ‘\_cpt’.  
      Après plusieurs exécutions, on a trouvé que la valeur de ‘\_cpt’ change à chaque exécution. Donc, on a problème de synchronisation.
2. **La classe Cptb.**
   1. La différence entre Thr1.run () et Thr2.start () :

Thr1.run () : Ne crée pas de nouveau thread, elle fait appel à la méthode   
« run () » dans le thread au courant d’exécution « main ».

En d’autres termes, « run » est équivalent à « main ».

Thr2.start () : crée un nouveau thread, dans lequel ; elle exécute la méthode « run () ». Elle a son propre scenario d’exécution.

* 1. Le nombre maximum de processus légers Co-existants au cours de l’exécution du Cptb est : 2
     + 1. Le thread main créé lors de l’exécution de la classe Cptb.
       2. Le thread Thr2 créé lors de l’appel de la méthode « start () ».
  2. Dans la classe Cptb, on n’a pas de variable partagée entre Thr1 et Thr2.
  3. Le nombre d’incrémentation de la variable global ‘\_cpt’, par chaque processus léger est 100000.
  4. La valeur affichée à la fin de l’exécution du code est 200001.
  5. On a deux threads qui s’exécutent séparément. La variable ‘\_cpt’ n’est pas commune entre eux. Donc pas de problème de synchronisation. Après plusieurs exécutions, on a eu un résultat fixe = 200001.

1. **La classe Train**

Dans cette classe, on a deux threads qui s’exécutent en parallèle.   
Ce programme finit son exécution avant la fin des threads créés.   
Imprime « Fin de main ».

Dans cette classe, il y a appel à la méthode « sleep (I) », qui fait attendre le thread en courant ‘I’ millisecondes. Et comme ‘TGV’ doit attendre (10\*500) millisecondes, et ‘CORAIL’ doit attendre (20\*500) millisecondes. Le Schedule va être attribué à ‘TVG’ en premier.

**Remarque :**

Dans GitHub, il existe les deux programmes « Cpt » et Cptb » du TP.

« P2 », « P2B » et « P2C » existe en code java.

« P2D » est la deuxième partie de « P2C ».