

Travaux pratiques 4  
Programmation multi-threadée en Java

### Exercice 1 – Hello World !

Implémentez deux versions Java multi-threadées du fameux programme "Hello World !" où chaque thread affiche :

- un message ("Hello World !" ou quelque chose de plus original si vous préférez) ;
- un identifiant de thread unique entre 0 et le nombre de threads moins 1 (vous devrez créer vous même cet identifiant);
- l'identifiant de thread créé par Java (que vous pouvez récupérer en utilisant la méthode **getId()** de la classe **Thread**) ;
- le nom de thread créé par Java (que vous pouvez récupérer en utilisant la méthode **getName()** de la classe **Thread**).

Version 1 : classe qui hérite de la classe **Thread**.

Version 2 : classe qui implémente l'interface **Runnable**.

### Exercice 2 – Modèle producteur-consommateur

On veut développer un programme multi-threadé en Java qui implémente un schéma simple de producteur-consommateur. Ce dernier contient les éléments suivants :

- une boîte de messages qui ne peut contenir qu'un message à la fois ;
- un producteur qui génère (produit) un message et le place dans la boîte si celle-ci est vide ;
- un consommateur qui récupère et affiche (consomme) le message de la boîte s'il y en a un.

Chaque message doit être utilisé une et une seule fois : lorsqu'un message est récupéré, il doit être retiré de la boîte et chaque message placé dans la boîte doit être récupéré.



Ce schéma peut être implémenté à l'aide des 4 classes suivantes :

**1. BoiteMsg**

- permet de stocker un message sous forme de chaîne de caractères ;
- méthode **déposer** : si possible, dépose un message passé en paramètre ;
- méthode **recupérer** : si possible, retourne le message de la boîte.

**2. Producteur**

- créé et remplit un tableau de messages ;
- dépose progressivement chaque message dans la boîte en dormant entre 1000 et 5000 millisecondes entre chaque dépôt ;
- place un message "termine" quand tous les messages ont été déposés.

**3. Consommateur**

- tant que le message "termine" n'est pas reçu, récupère et affiche les messages en dormant entre 1000 et 5000 millisecondes entre chaque récupération.

**4. Test**

- Crée l'ensemble des objets et des threads nécessaires à l'exécution du programme.

Proposez un programme Java qui implémente le schéma décrit précédemment.

[Facultatif] Généralisez ensuite votre programme de sorte à ce qu'il utilise plusieurs producteurs, plusieurs consommateurs et une boîte de messages pouvant stocker plusieurs messages plutôt qu'un seul. Proposez un affichage qui indique les événements qui se produisent et les producteurs/consommateurs impliqués.