TP n°12
Programmation Concurrente

Dans ce TP, nous allons approcher la programmation concurrente. L'idée de la programmation concurrente est de réaliser plusieurs traitement "en même temps". Faites un nouveau projet Java.
Dans un premier temps, nous allons faire deux actions de façon séquentielle. Puis comprendre que ces deux actions doivent se faire simultanément.

Une baignoire solide

Nous allons écrire le code d'une baignoire qui va se remplir avec de l'eau par un robinet. Pour cette modélisation nous avons besoin d'une classe Baignoire et d'une classe Robinet.

Une **Baignoire** est composée des choses suivantes :

- Un volume maximum (constante de type int). C'est la capacité de la baignoire.
- Un volume (variable de type int). Ce volume correspond à la quantité d'eau dans la baignoire. Il ne peut pas dépasser le volume maximum et il ne peut pas être négatif. Si le volume est à zéro, on considère que la baignoire est vide.

1° - Ajoutez un constructeur à votre classe et les méthodes get/set dont vous avez besoin.

Un **Robinet** est composé des choses suivantes :

- Un objet de type baignoire que ce robinet rempli.
- Un volume débité (constante de type int). On considère que le robinet une fois ouvert débite en continu le même volume d'eau.

2° - Ajoutez un constructeur à votre classe permettant de créer un Robinet.

3° - Ajoutez une méthode "debite" qui va remplir la baignoire jusqu'à ce qu'elle soit pleine. Attention à ne pas faire déborder la baignoire. A chaque débit, faites un affichage du volume d'eau que contient la baignoire.

Voici la méthode main permettant de tester votre code dans une classe Main :

```
public static void main(String[] args) {

    Baignoire baignoire = new Baignoire(1000);
    Robinet robinet = new Robinet(baignoire, 50);

    robinet.debite();
}
```

Une baignoire qui fuit…

Sauf que notre baignoire elle fuit… Ajoutez cette fonctionnalité dans la classe Baignoire avec un nouvel attribut :

- Un volume fuite (variable de type int). Représente le volume d'eau qui fuit en continu de la baignoire à partir du moment ou il y a de l'eau dedans.

4° - Ajoutez une méthode fuite dans la classe Baignoire afin de simuler cette fuite. Attention la fuite se produit tant qu'il y a de l'eau dans la baignoire.
A chaque fuite, faites un affichage du volume d'eau que contient la baignoire.

Voici la méthode main permettant de tester votre code dans une classe Main :

```
public static void main(String[] args) {

    Baignoire baignoire = new Baignoire(1000, 10);
    Robinet robinet = new Robinet(baignoire, 50);

    robinet.debite();
    baignoire.fuite();
}
```

Sauf qu'une baignoire ça fuit en même temps que ça se rempli…

Faites en sorte que ces deux opérations ("debite" du robinet et "fuite" de la baignoire) se fassent ensemble, et non pas de façon synchrone, c'est à dire l'une après l'autre.

Pour cela utilisez la programmation concurrente, votre main doit maintenant être :

```
public static void main(String[] args) {

    Baignoire baignoire = new Baignoire(1000, 10);
    Robinet robinet = new Robinet(baignoire, 50);

    Thread threadBaignoire = new Thread(baignoire);
    Thread threadRobinet = new Thread(robinet);

    threadBaignoire.start();
    threadRobinet.start();
}
```

Vous pouvez également tenter de colmater la fuite lorsqu'il n'y a pas d'eau dans la baignoire. Réduisez la fuite d'une unité lorsque vous le pouvez.