# TD10-Threads

——M1——

## 1 Exercise 1

1. Créer une méthode prenant en argument un entier id et retournant un `Runnable` (sous forme de classe anonyme) dont vous redéfinirez la méthode run afin qu'elle fasse une boucle infinie affichant "Thread id - i", où i est le nombre de fois où la boucle a été effectuée.

2. Créer 2 threads (prenant en argument du constructeur le Runnable créé par votre méthode précédente) avec id 1 et 2 et les lancer. Que remarquez-vous sur la valeur des compteurs de chaque thread ?

## 2 Exercise 2

On modifie le code de l'exercice précédent.

1. Modifiez votre code pour que le nombre de threads à lancer simultanément soit donné en argument (args) de votre programme.

2. On souhaite que l'utilisateur puisse interrompre un thread en particulier en entrant son ID au clavier (utiliser un `Scanner`). Modifier le code de vos threads pour que les affichages ne se fassent plus à chaque tour de boucle mais uniquement lorsqu'on sort de la boucle. Modifier la condition de la boucle pour qu'elle ne soit plus infinie mais lorsque le thread courrant est interrompu (visible par `Thread.currentThread().isInterrupted()`).

3. Votre programme doit s'arrêter lorsque tous les threads ont été interrompus.

## 3 Exercise 3

Le code suivant lance deux threads faisant des opérations financières.

```
public class Account {
  private int val;
  private String oper;

  public static void main(String[] args) {
    final Account acc=new Account();
    Thread retrait = new Thread(new Runnable() {
      @Override
      public void run() {
        for(int i=0;i<5000;++i) {
          acc.oper = "retrait";
          acc.val = 50;
          System.out.println(acc.oper + " " + acc.val + "euros");
```

```
      }
    }
  });
  Thread depot = new Thread(new Runnable() {
    @Override
    public void run() {
      for(int i=0;i<5000;++i) {
        acc.oper = "depot";
        acc.val = 100;
        System.out.println(acc.oper + " " + acc.val + "euros");
      }
    }
  });
  retrait.start();
  depot.start();
 }
}
```

1. Ce programme est-il thread-safe ?
2. Est-ce qu'il y a des retraits de 100 ou des dépots de 50? Pourquoi ?
3. Ajouter des sections critiques avec des blocs `synchronized` pour éviter ce comportement.
4. Pourquoi ne faut-il pas faire une synchronisation sur `this` en général ? Faites-le sur un `Object` privé de la classe.
5. Est-il possible de replacer les affichages faits dans la méthode run par un affichage faisant appel à des getters pour `oper` et `val` ?
6. Modifier la classe pour qu'elle utilise des verrous (`ReentrantLock`) plutôt que des blocs synchronized. Quel est l'avantage de faire cela ?