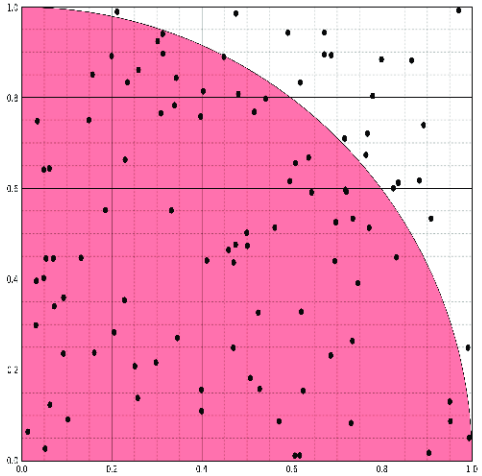
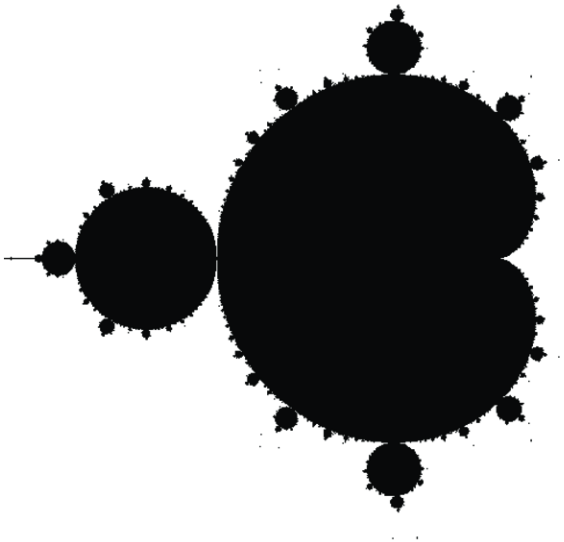
Retour au premier exemple

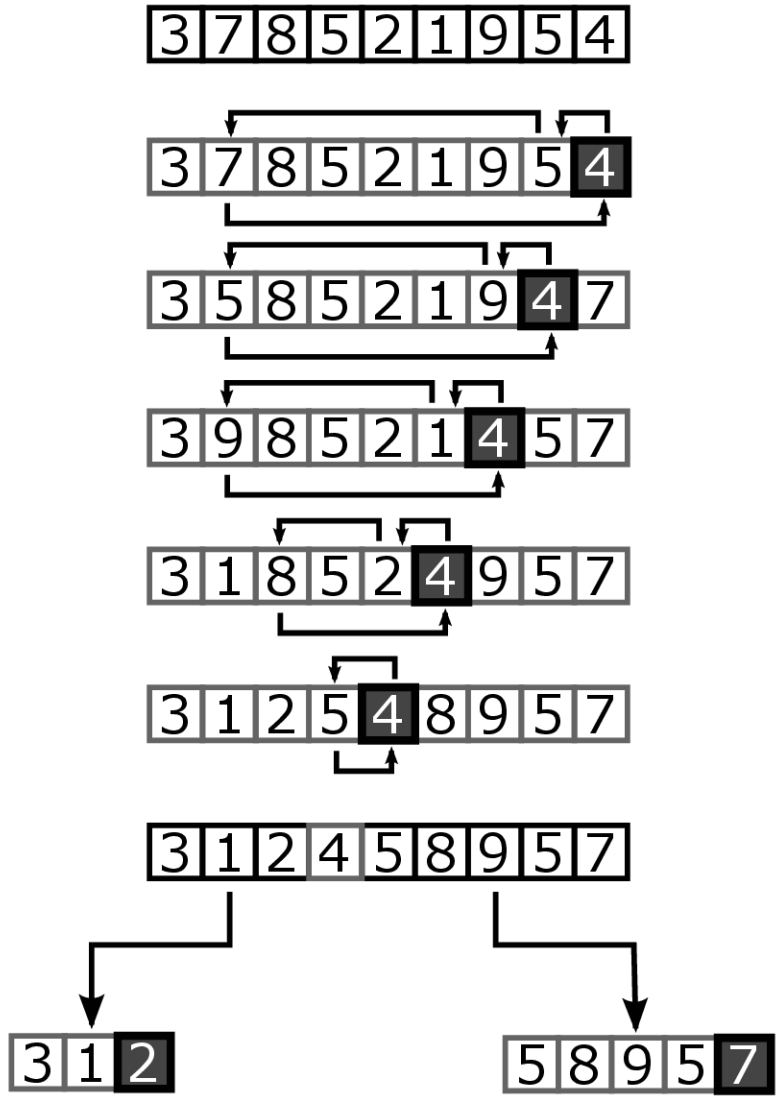
```
public static void main(String[] args) throws Exception {
  A a = new A();              // Création d'un objet a de la classe A
  a.start();                  // Lancement du thread a
  a.valeur = 1;               // Modification de l'attribut valeur
  a.fin = true;               // Modification de l'attribut fin
}
static class A extends Thread {
  public int valeur = 0 ;
  public volatile boolean fin = false ;

  public void run() {
    while(! fin) {} ;         // Attente active
    System.out.println(valeur) ;
  }
}
```

Ce programme termine-t-il ? Peut-il afficher 0 ?

Exemples de négligences sans conséquence







Les données partagées doivent être synchronisées pour être visibles.

Exemple d'exécution qui n'est pas séquentiellement consistante

```
class NotSoSimple {
  int a = 1, b = 2;

  void writer() {
    a = 3;
    b = 4;
  }
  void reader() {
    System.out.print("b = " + b +", ");
    System.out.println("a = " + a);
  }
}
```

Ce programme peut légalement afficher **`"b = 4, a = 1"`**.

① La mise-à-jour de la valeur de **`a`** ne semble pas avoir été réalisée, *ou bien*

② Le programme ne semble pas s'exécuter dans l'ordre des instructions de **`writer()`**.

Garantie des programmes bien synchronisés

Bonne nouvelle :

≪ Le modèle mémoire Java garantit que tous les résultats d'un programme **sans data-race** sont **séquentiellement consistants !** ≫

C'est bien ce que l'on veut !

Ce que ça veut dire en pratique.

Si la seule explication possible du résultat d'un programme est que

- ou bien les écritures n'ont pas été correctement réalisées en mémoire ;
- ou bien les instructions du programme n'ont pas été réalisées dans l'ordre

c'est-à-dire que l'exécution n'est pas *séquentiellement consistante*, alors le programme contient nécessairement une *data-race*. ≫

Ça peut aider pour débugger !

Une situation de data-race

```
class A {
  int f;
  public A() { f = 42 ; }
}
class B {
  A a;
  static void writer() { a = new A() ; }
  static void reader() { if ( a != null ) println(a.f) ; }
}
```

Deux threads appliquent simultanément les deux méthodes **`writer()`** et **`reader()`** sur un même objet de la classe B.

Que va afficher le second thread ?