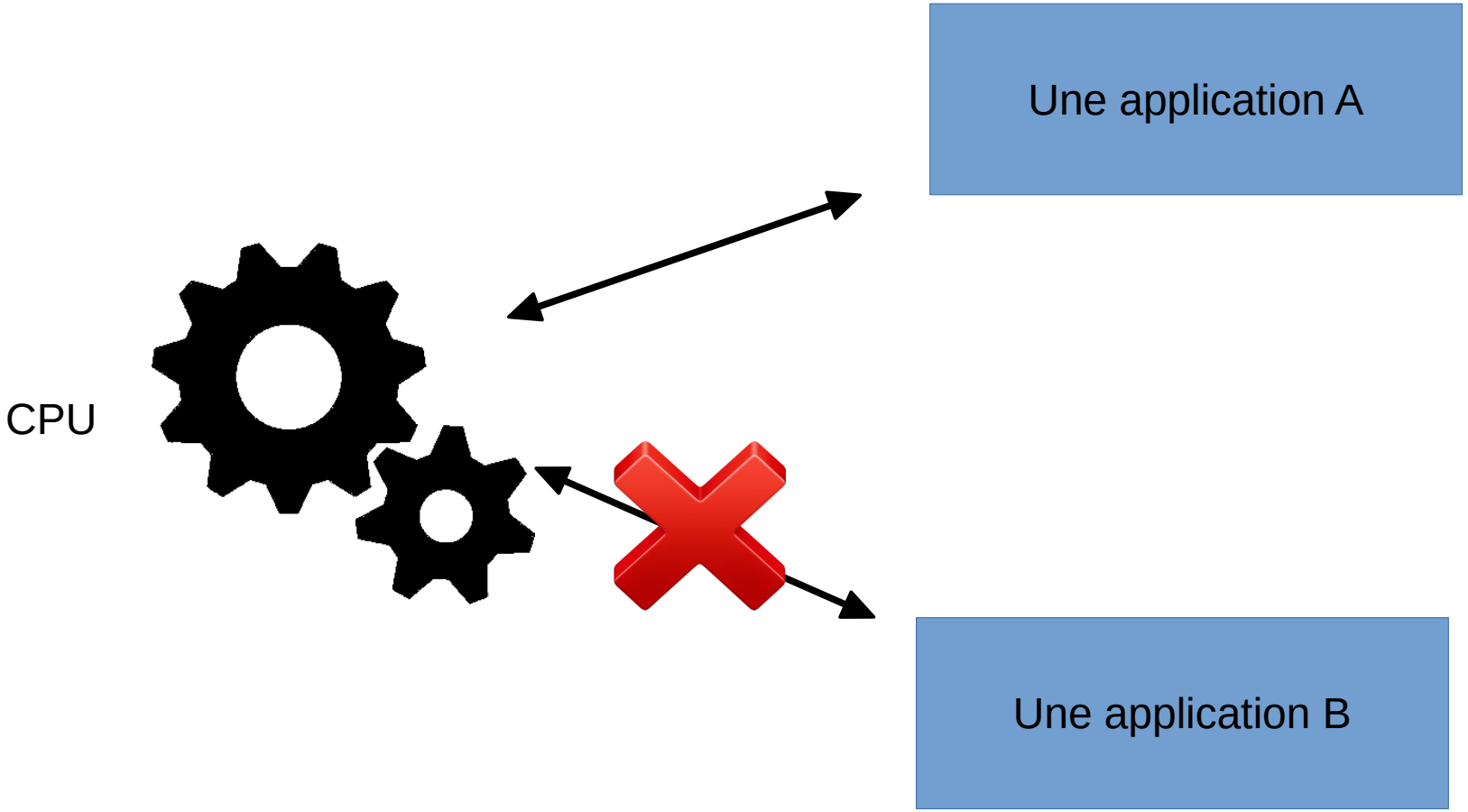
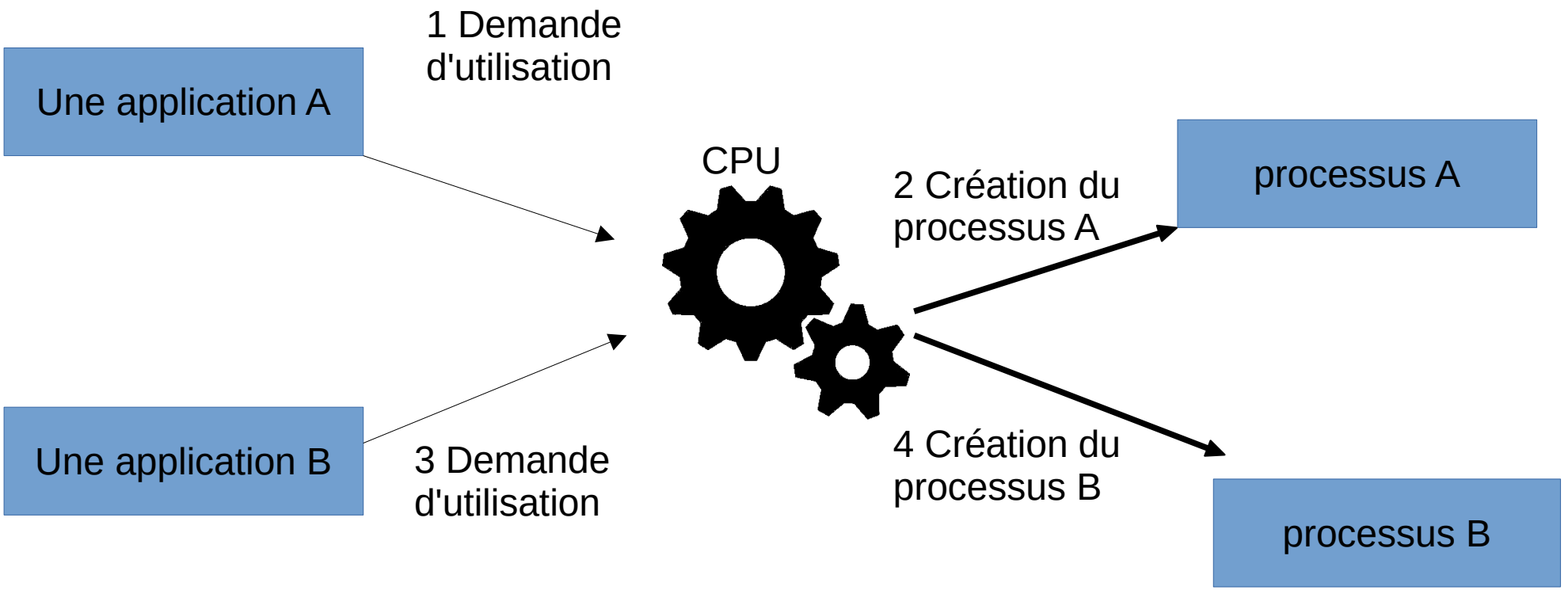
Multithreading et concurrence d'accès

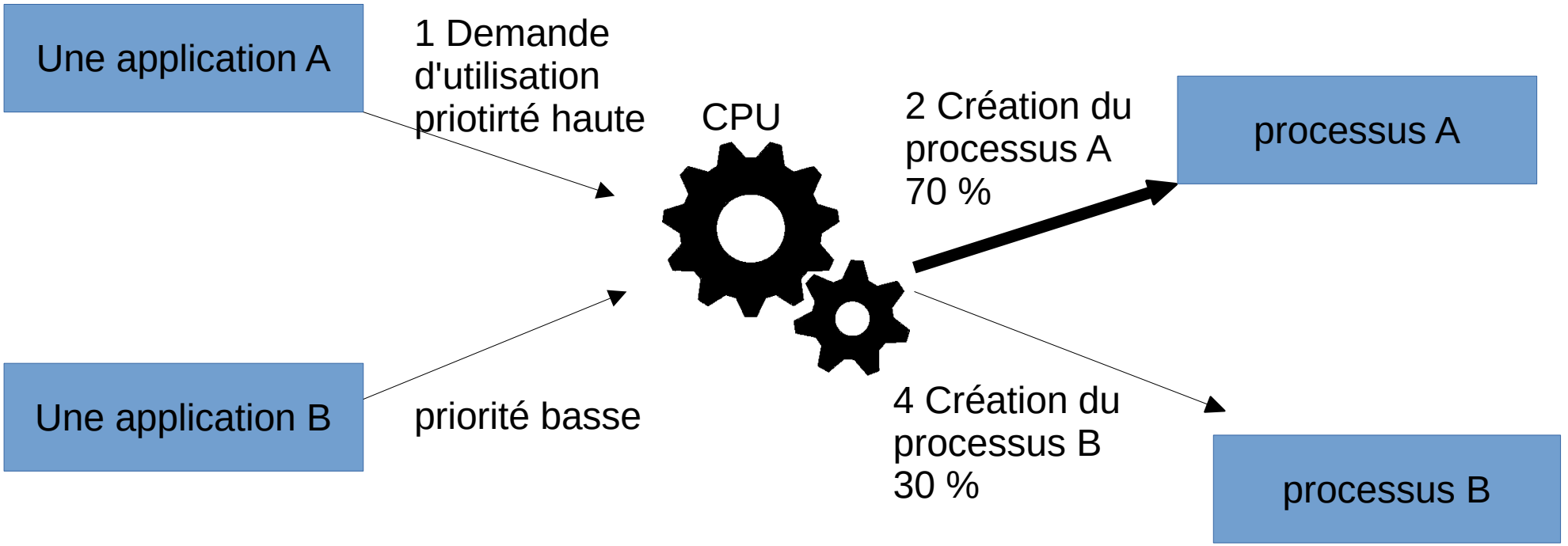
Monoprocessing



Multiprocessus



allocation du processeur



Les Threads

- Thread = canal d' exécution

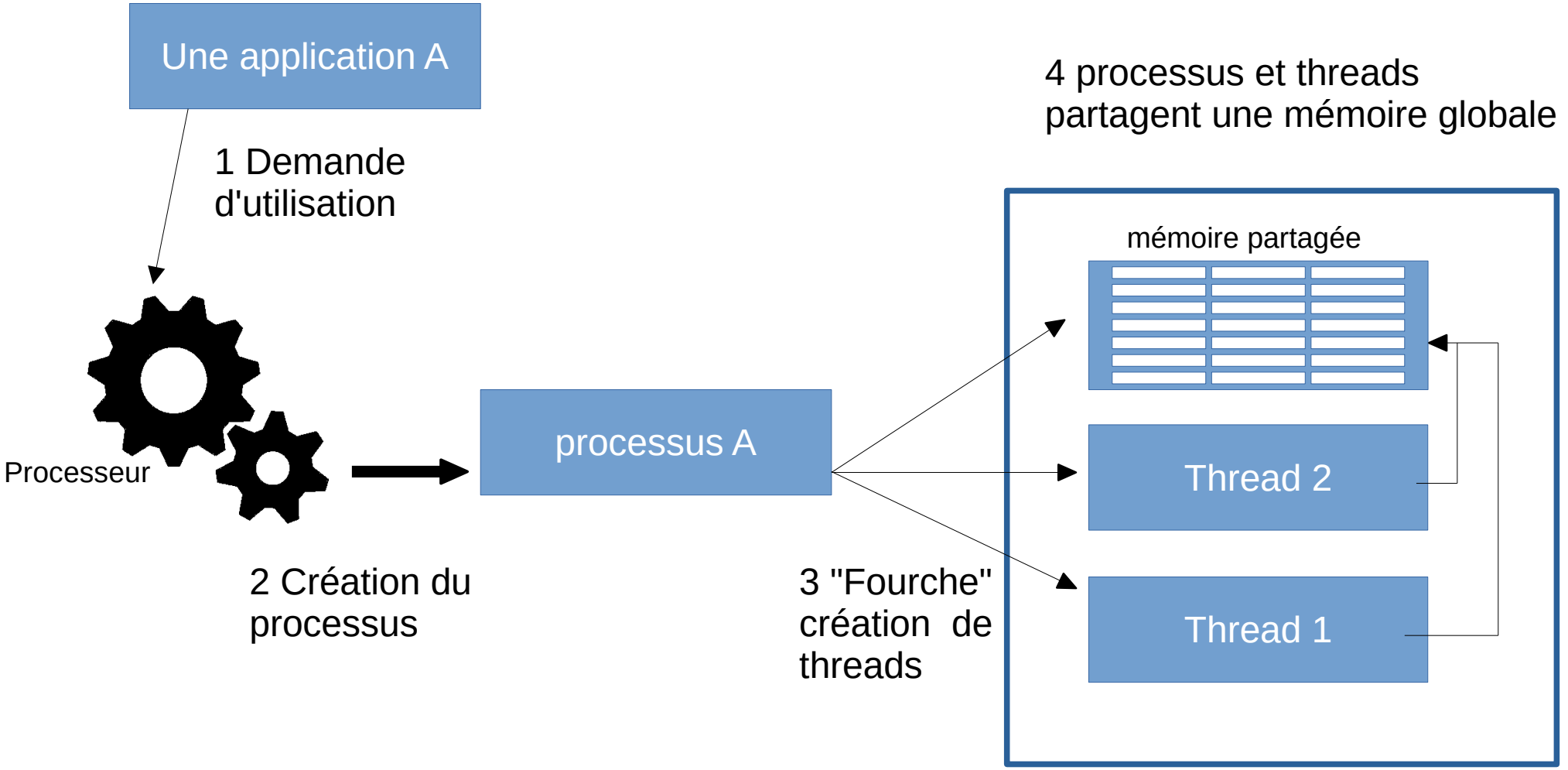
- Un thread est une unité d'exécution faisant partie d'un programme. Cette unité fonctionne de façon autonome et parallèlement à d'autres threads. Le principal avantage des threads est de pouvoir répartir différents traitements d'un même programme en plusieurs unités distinctes pour permettre leurs exécutions "simultanées".

- Sur une machine monoprocesseur, c'est le système d'exploitation qui alloue du temps d'utilisation du CPU pour accomplir les traitements de chaque threads, donnant ainsi l'impression que ces traitements sont réalisés en parallèle.

- Sur une machine multiprocesseur, le système d'exploitation peut répartir l'exécution sur plusieurs coeurs, ce qui peut effectivement permettre de réaliser des traitements en parallèle.

source: https://www.jmdoudoux.fr/java/dej/chap-threads.htm

Environnement multi-Thread



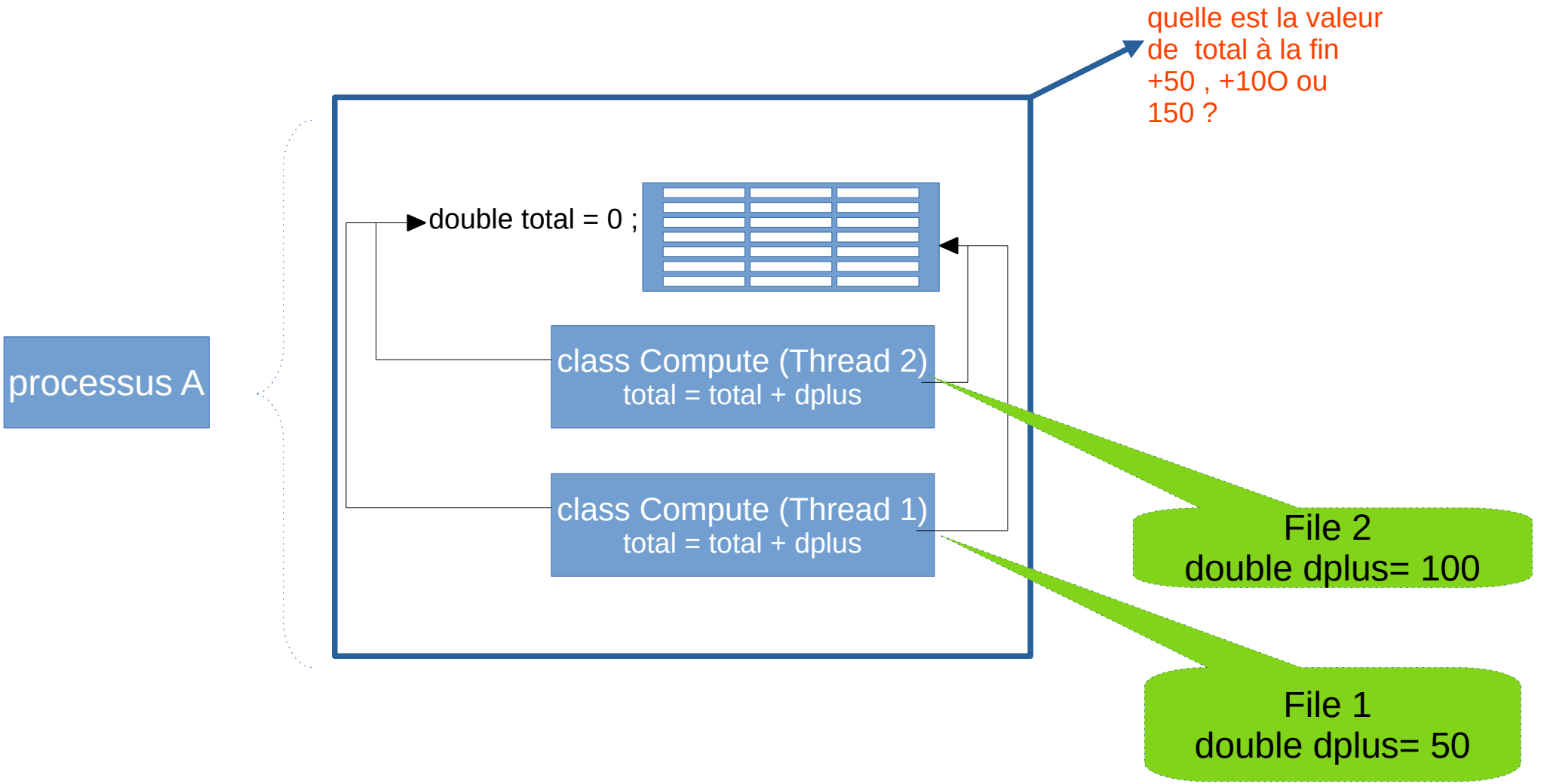
Multi-threading

- Portée
  - possibilité d'exécution sur une CPU mono ou multi-thread
  - ressource machine optimisée
  - compatible programmation asynchrone
- Limites
  - synchroniser les accès aux ressources devient plus complexe
  - nicvezu de compétence spécifique requis

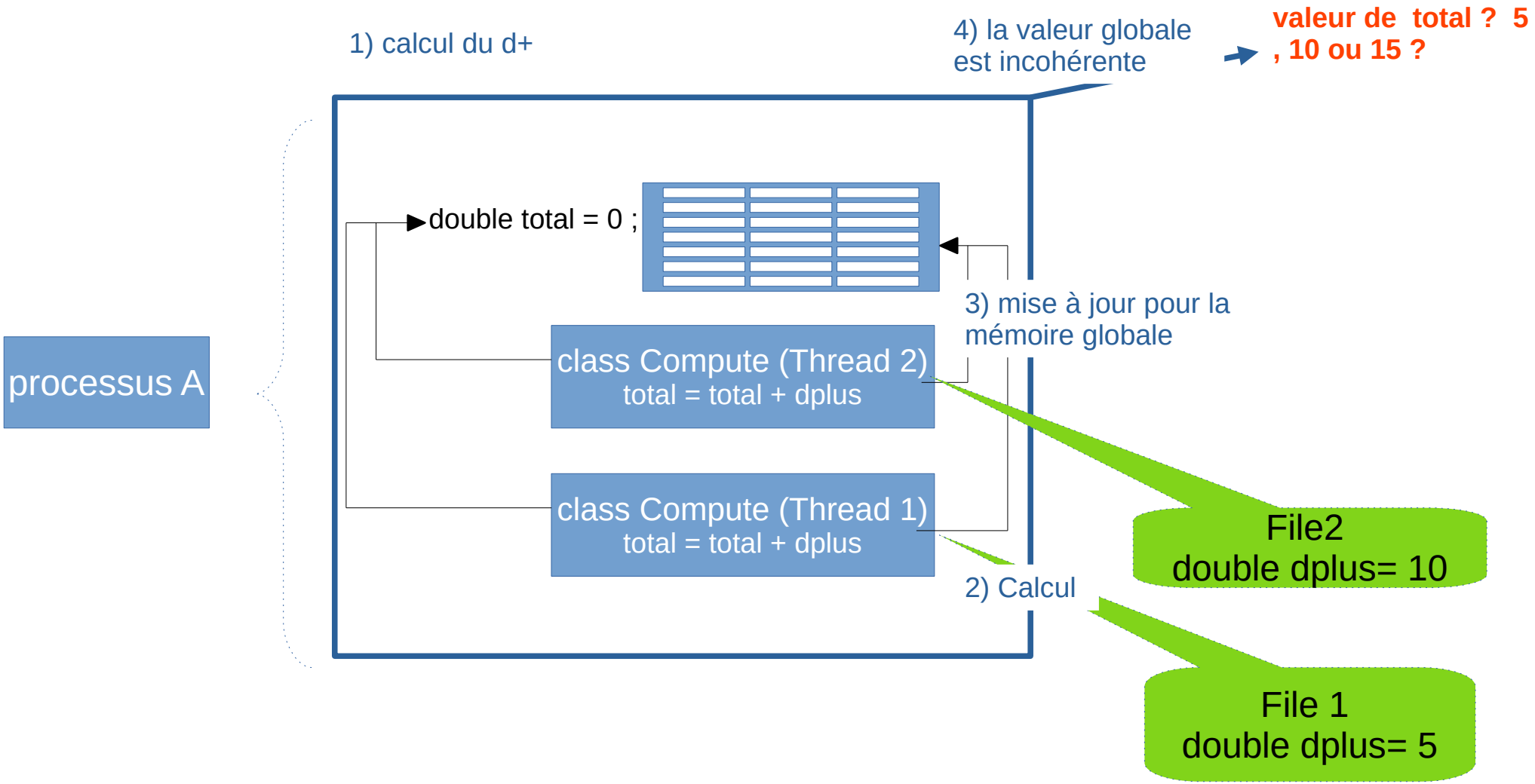
Problèmatiques liées au multi-threading

- plusieurs tâches consomment et modifient les mêmes variables
- séquencer les accès doit éviter les incohérences
- on veut éviter les deads locks (ou interblocages)

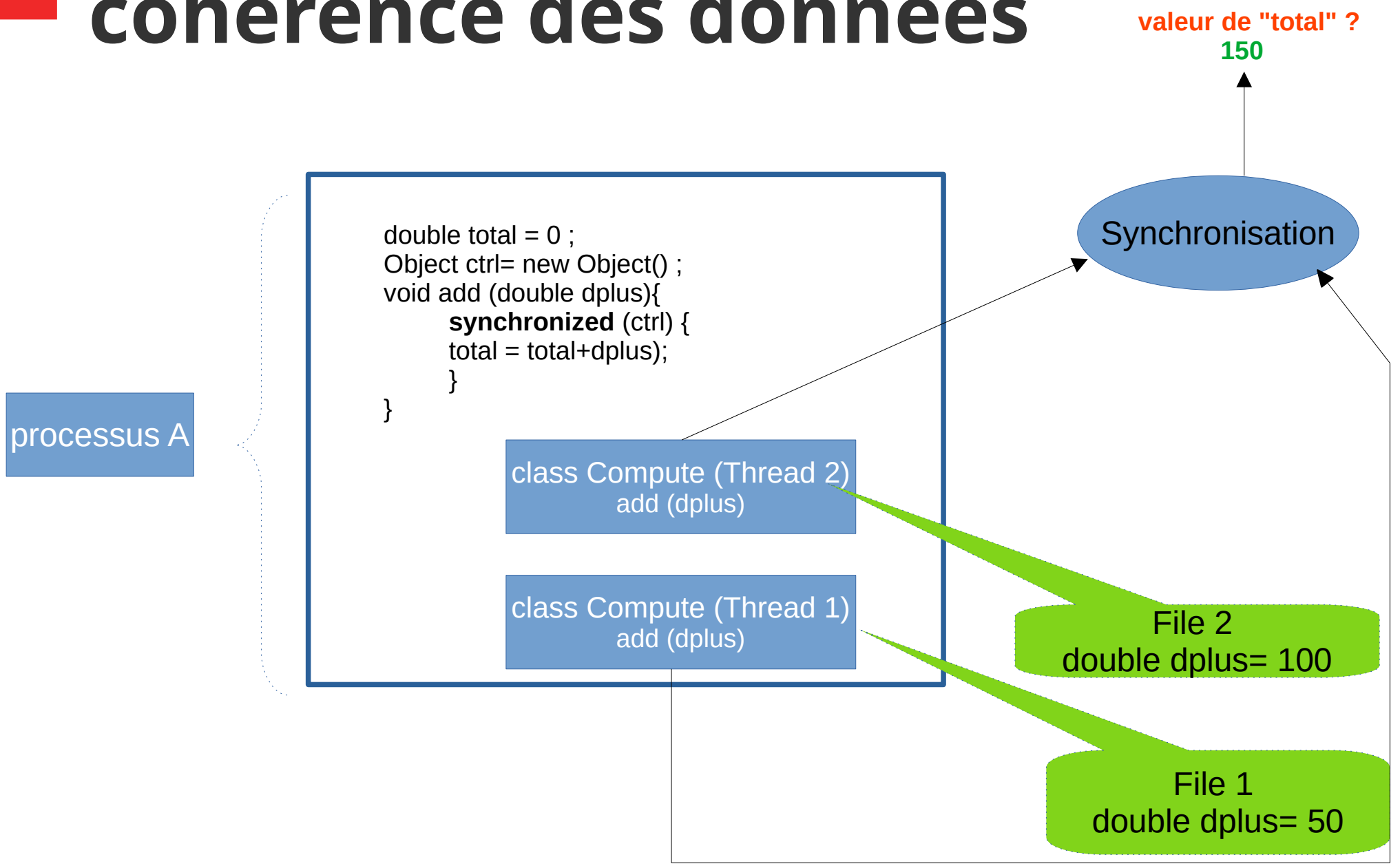
Multi-threading - la cohérence des données en question



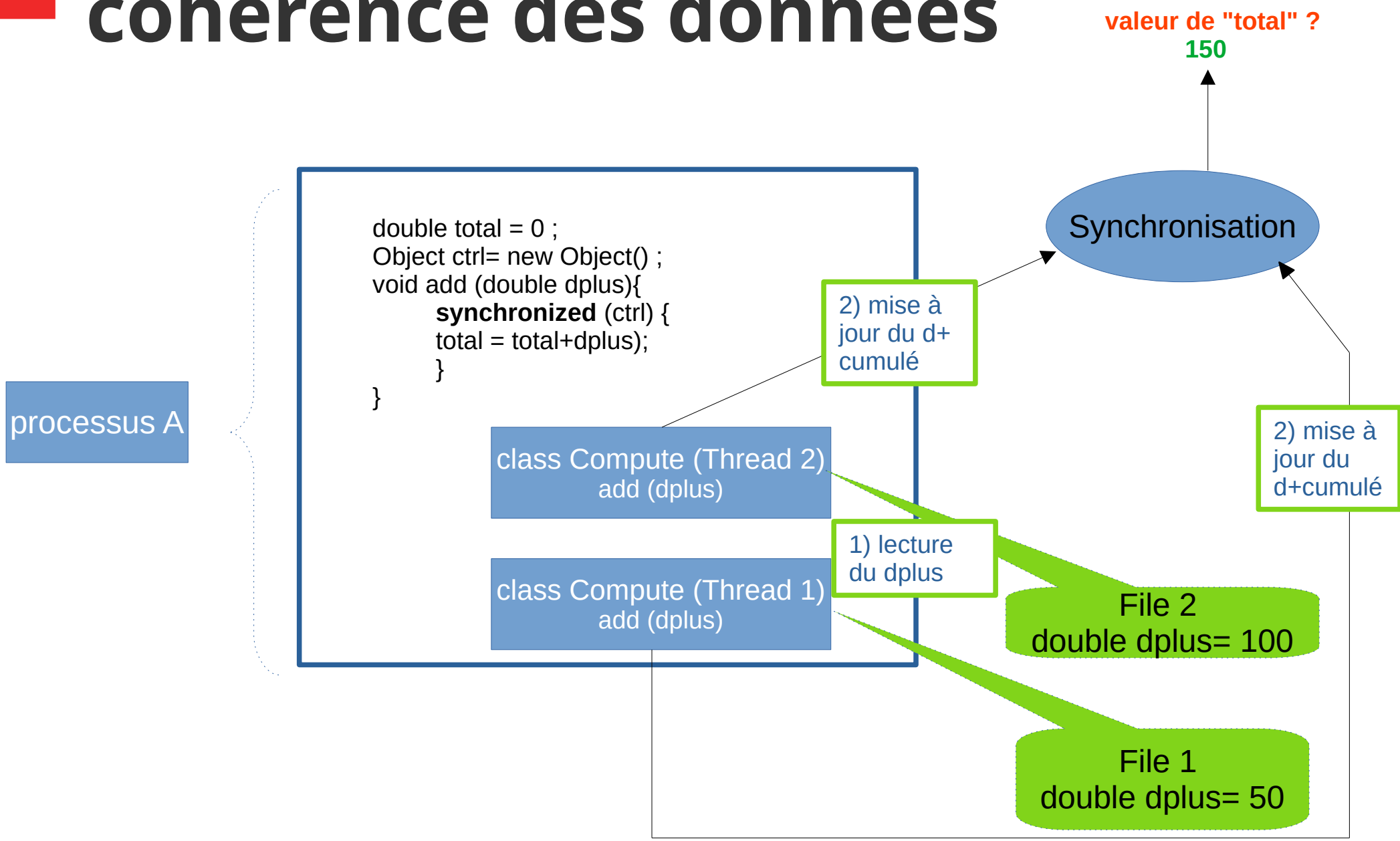
Multi-threading - la cohérence des données en question



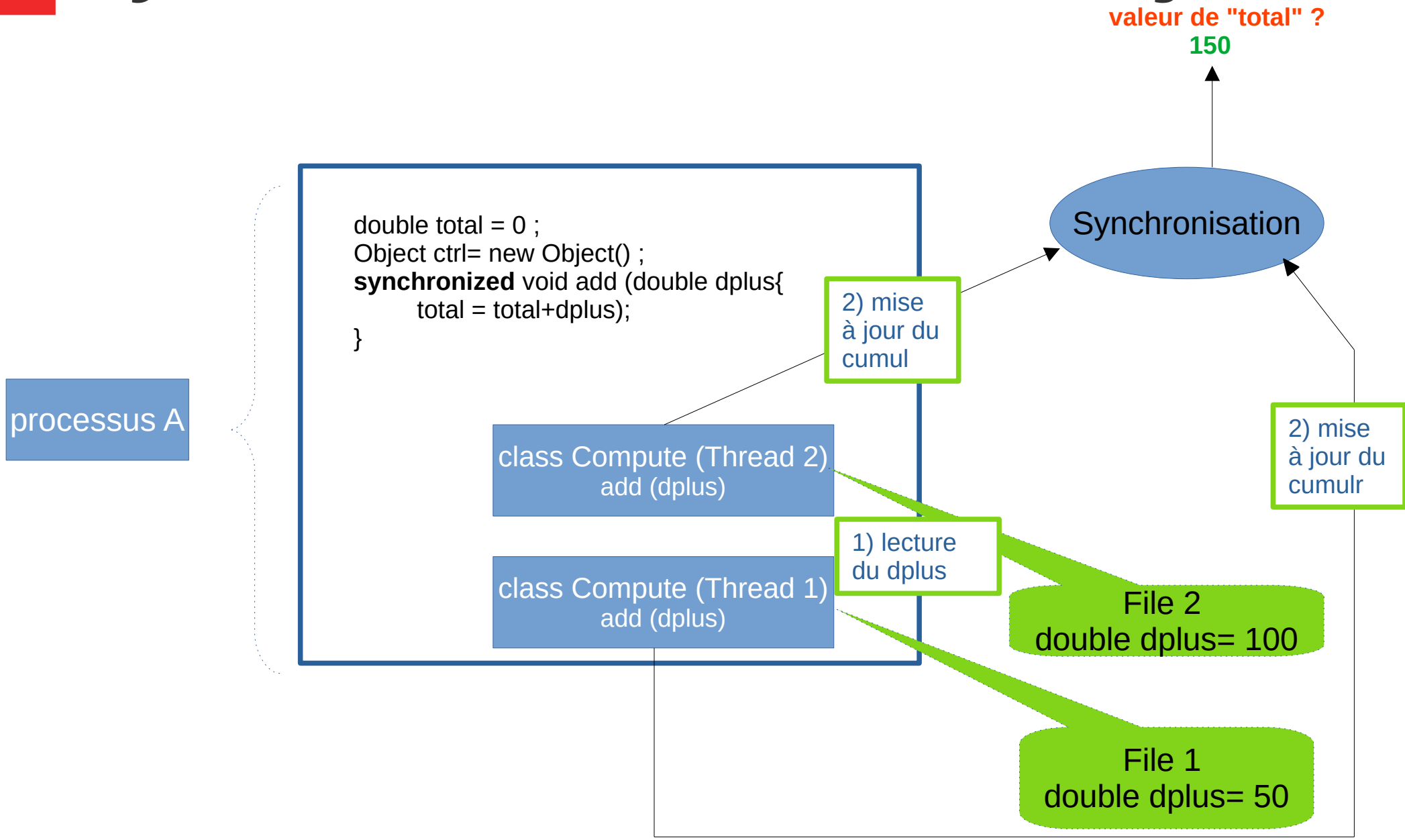
Synchroniser pour conserver la cohérence des données



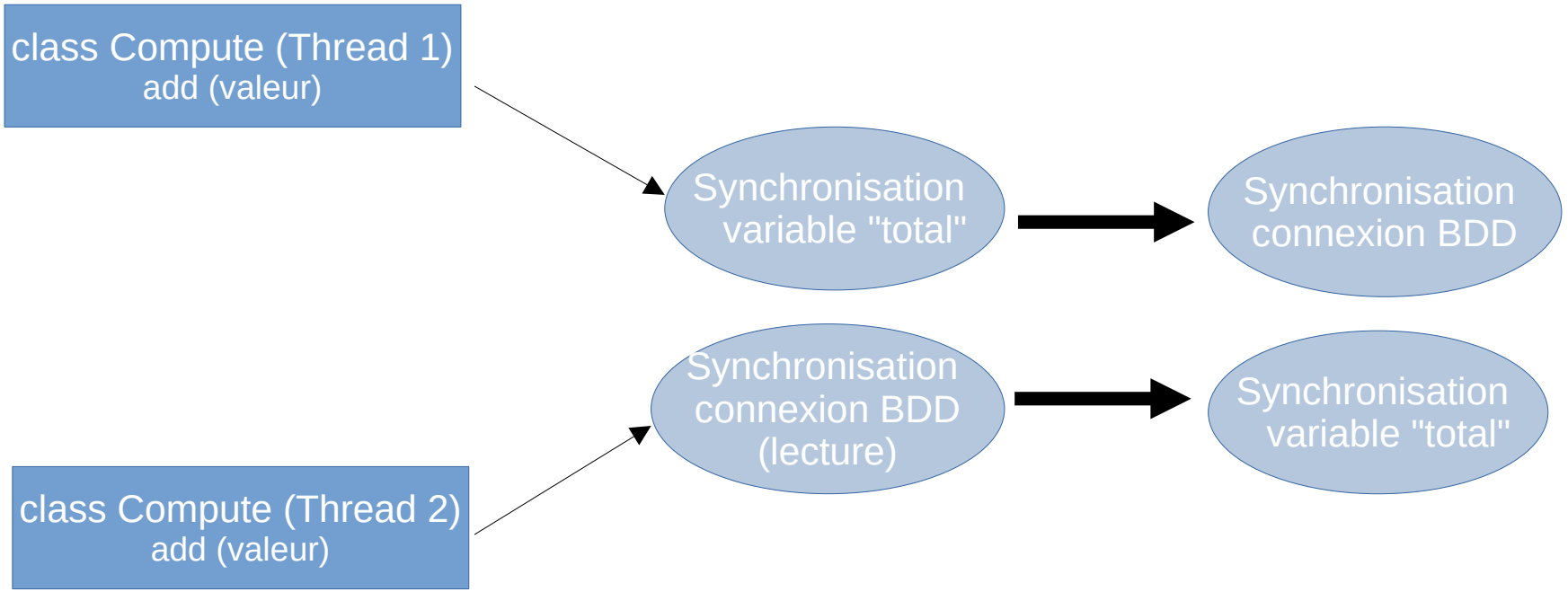
Synchroniser pour conserver la cohérence des données



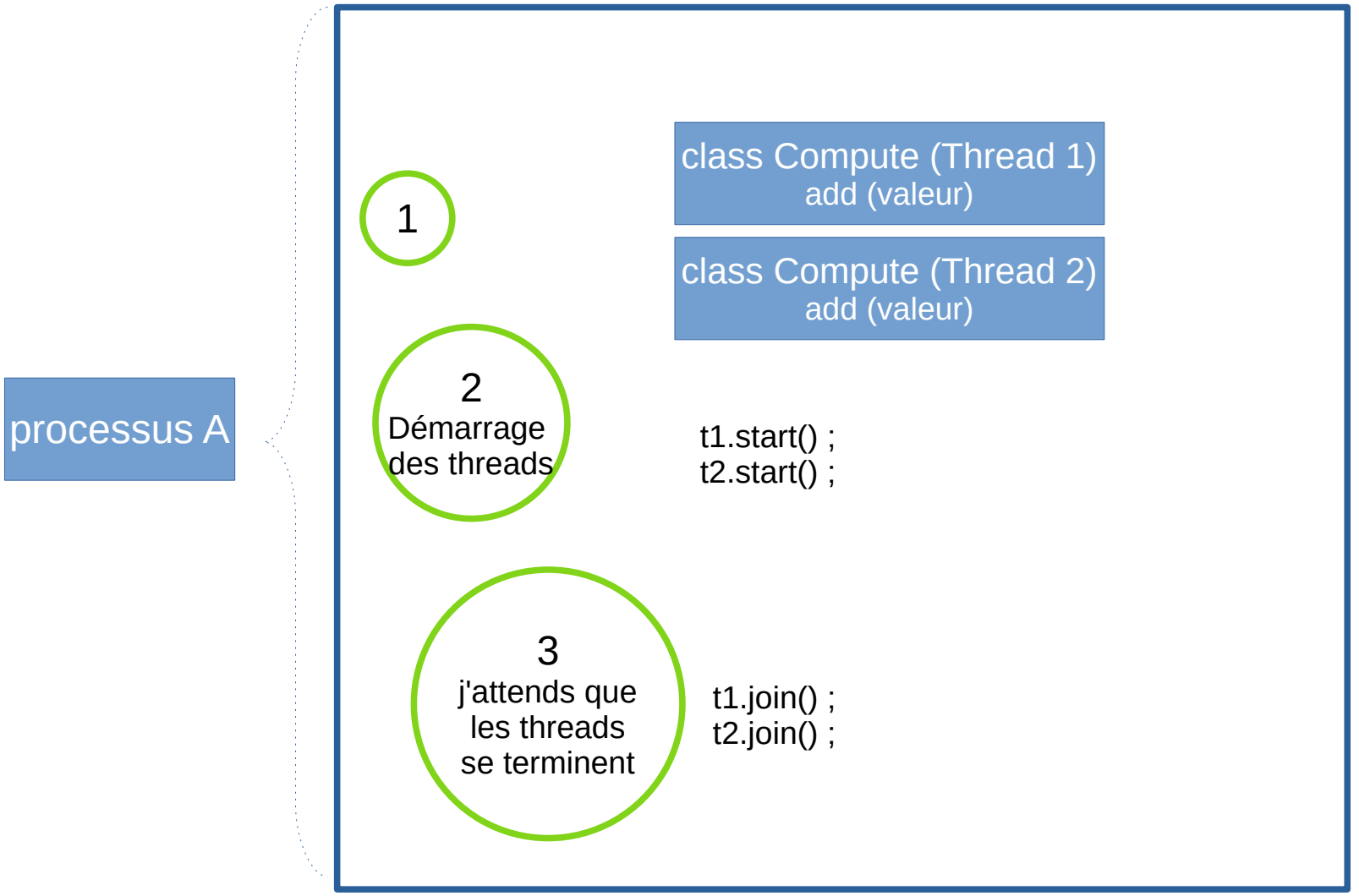
Synchroniser totale sur l'objet



Synchronisation & exclusion mutuelle



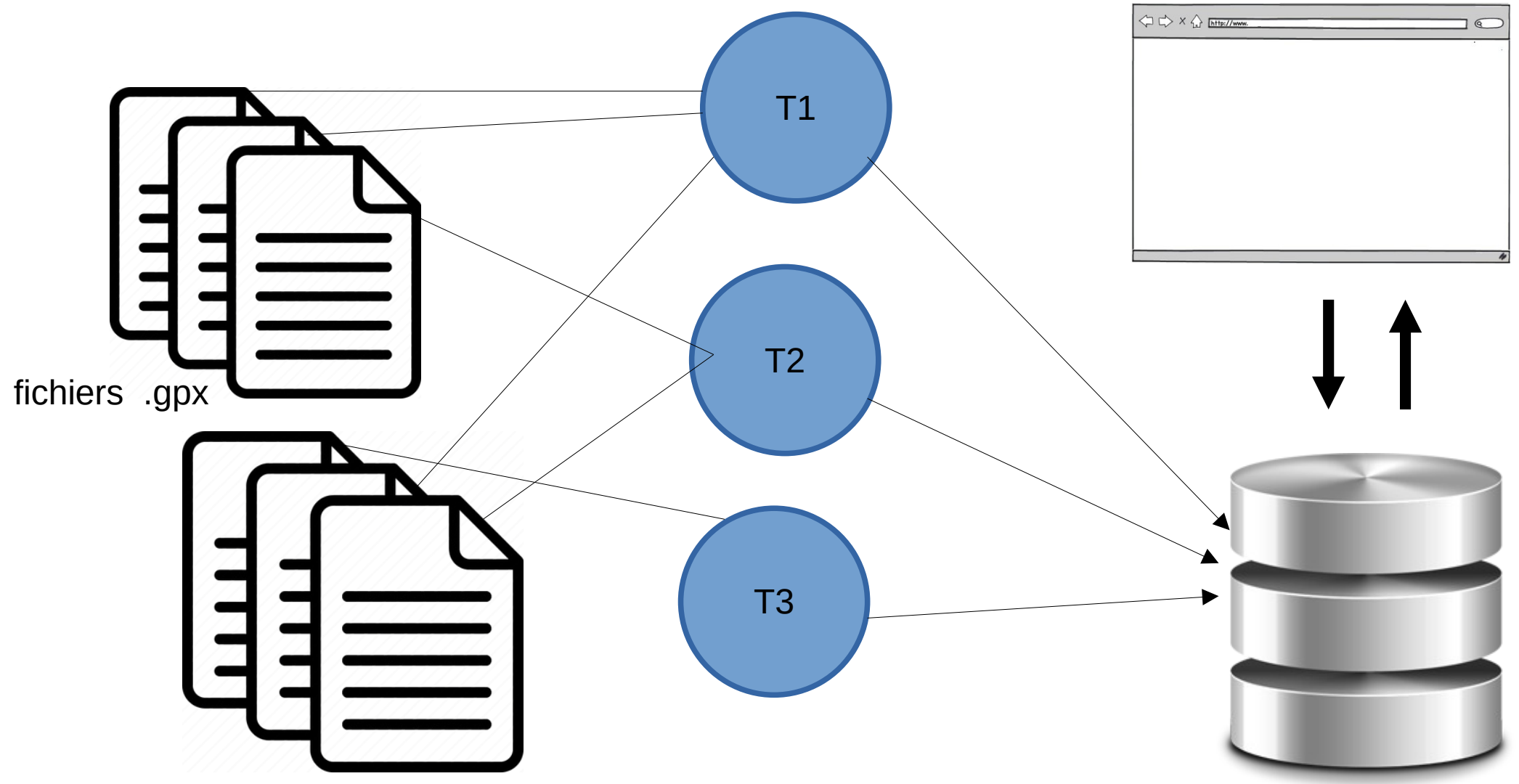
Jonction des threads (ou attendre que les tâches soient accomplies)



Démo

- mettre en œuvre les basiques du multi-threading
- par un projet simple de gestion des threads
- visant à la prise en main du pilotage des threads

Synthèse de l'application



Etapes : Lecture du fichier > Traitement> Sauvegarde > consultation

Modèle de données

```java
@Data
public class Trace{

    private Long id;


    private List<Double[]> listePositions;
    private Integer dplus;
    private Long idClient ;


    private LocalDateTime timeStamp;
```

- une trace est composée d'une liste de positions
- une trace concerne un client
- une position est un tableau de coordonnées (x,y,z)
- on stocke le dplus calculé de la trace

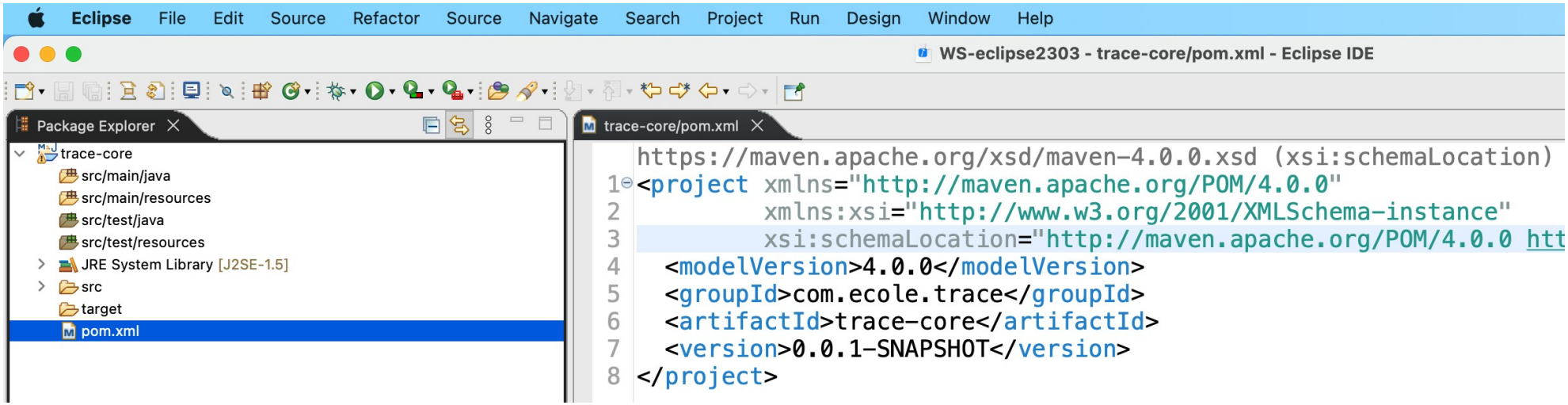
Java et le Multi-threading

- java simplifie la prise en charge parallèle de plusieurs tâches
- java simplifie le contrôle simple des exécutions parallèle par synchronisation
- java simplifie le contrôle avancé des exécutions parallèle en définissant des stratégies

java et le multi-threading

- Contrôle simple des threads en java
  - Lancer et stopper des threads
  - Déterminer le nombre de processeurs
  - les classes Thread et Runnable
  - l'interface ExecutorService

Création du projet



gestion basique de threads

Démo live Code ...

thread.start() ;

Démarrer un thread

thread.join() ;

attendre l'exécution d'un thread dans le thread principal

Multi-threading en java

- Contrôle simple des threads
  - problématique : sans synchronisation
  - avec synchronisation simple (synchronized)

  à suivre ...