



QUALITÉ DE DÉVELOPPEMENT

LA PROGRAMMATION CONCURRENT EN JAVA

🎓 2A - Bachelor Universitaire de Technologie
🏛️ IUT d'Orsay - Université Paris-Saclay - 2023/2024



Idir AIT SADOUNE

idir.ait-sadoune@universite-paris-saclay.fr

PLAN

- Systèmes multitâches
- Les processus et les threads en Java
- Quelques méthodes de la classe Thread

[Retour au plan](#) - [Retour à l'accueil](#)

PLAN

- > Systèmes multitâches
- > Les processus et les threads en Java
- > Quelques méthodes de la classe Thread

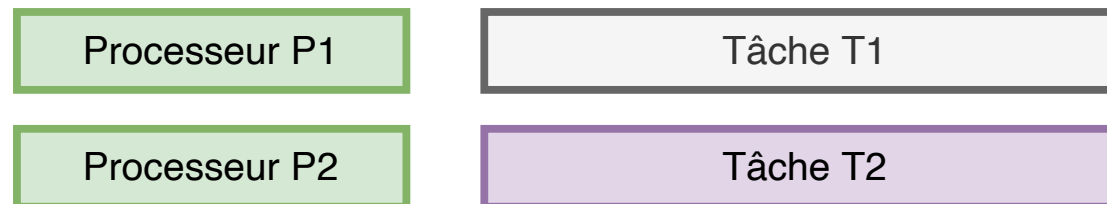
[Retour au plan](#) - [Retour à l'accueil](#)

SYSTÈMES MULTITÂCHES

Un système **multitâches** permet d'exécuter plusieurs programmes informatiques en parallèle.

SYSTÈMES MULTIPROCESSEURS

- **Vrai-parallélisme :**
un système **multiprocesseurs** permet un **parallélisme**, où chaque tâche est exécutée par un processeur.



SYSTÈMES MONOPROCESSEUR

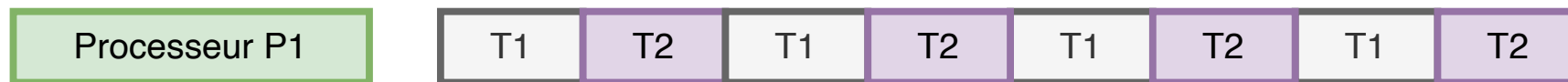
- **Pseudo-parallélisme :**

les systèmes **monoprocasseur** permettent un **parallélisme**, où toutes les tâches s'exécutent sur le même processeur.

- **Systèmes à traitement par lots**



- **Systèmes à temps partagé**



LA PROGRAMMATION CONCURRENTE

Un **paradigme de programmation** qui tient compte de l'existence de plusieurs unités d'exécution de base dans un programme :

- les processus et les threads.

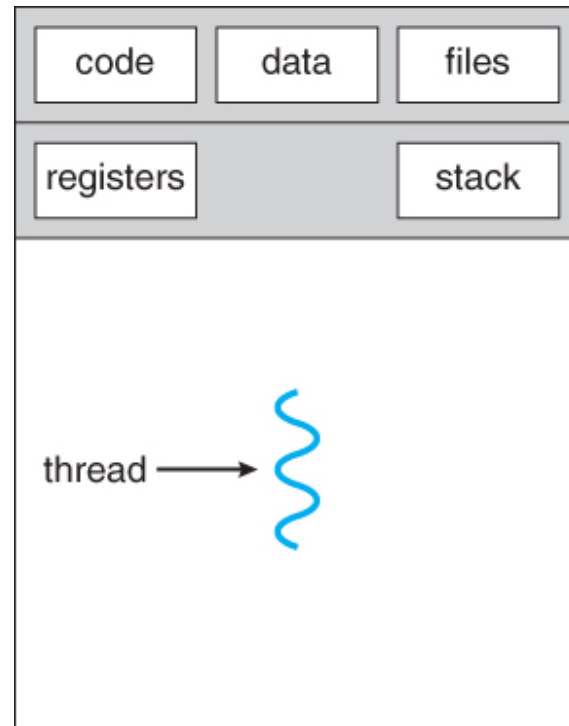
LA NOTION DE PROCESSUS

- Les processus sont considérés comme synonymes de programmes.
 - ▢ un processus est un programme en cours d'exécution.
- Un processus possède un environnement d'exécution autonome:
 - ▢ chaque processus a son propre espace mémoire.
- Pour gérer la communication entre les processus, la plupart des OS utilisent les IPC (Inter Process Communication) :
 - les pipes et les sockets (à étudier dans le cours suivant)

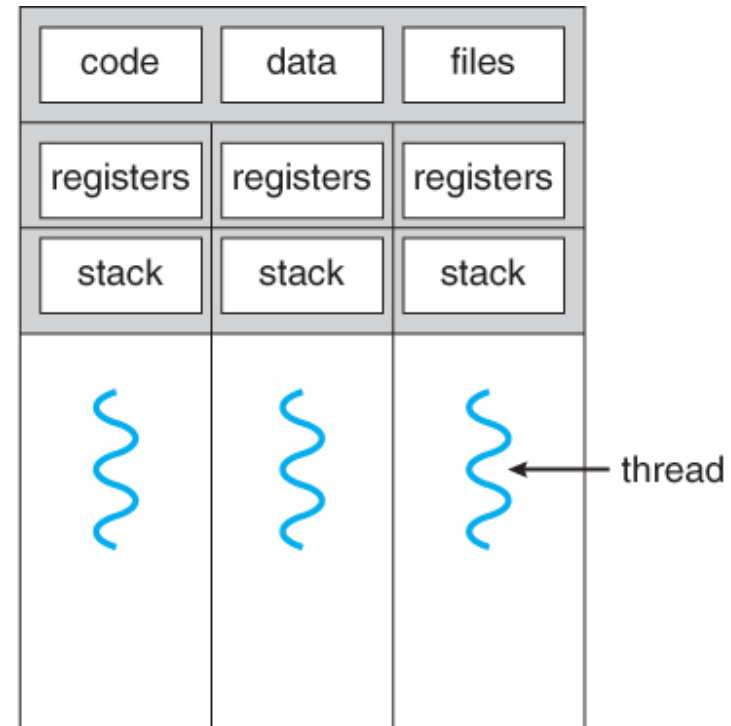
LA NOTION DE THREAD

- Les threads sont aussi appelés processus légers
 - ▢ la création d'un thread nécessite moins de ressources qu'un processus.
- Les threads sont créés par des processus. Les threads partagent les ressources du processus, y compris la mémoire et les fichiers ouverts.
- Cela rend la communication efficace mais potentiellement problématique
 - ▢ synchronisations et gestion des ressources partagées

PROCESSUS VS THREAD



single-threaded process



multithreaded process

PLAN

- Systèmes multitâches
- Les processus et les threads en Java
- Quelques méthodes de la classe Thread

[Retour au plan](#) - [Retour à l'accueil](#)

LES PROCESSUS EN JAVA

- La description du comportement d'un processus en **Java** se fait en implémentant la méthode **main()** dans une classe.

```
1 public class ProcessInstanciation {  
2     public static void main(String[] args) {  
3         System.out.println("Run with main method");  
4     }  
5 }
```

- La création d'un processus se fait en exécutant la méthode **main()**.

```
Run with main method
```

LES THREADS EN JAVA

- La **création** d'un thread se fait de deux manières:
 1. en dérivant la classe **Thread** et en **surchargeant** la méthode **run()**.
 2. en implémentant l'**interface Runnable** contenant la méthode **run()**.
- L'**exécution** d'un thread se fait en utilisant la méthode **start()**.
Le code métier (la méthode **run()**) sera exécuté en tâche de fond.

ETENDRE LA CLASSE Thread

```
1 public class FromThreadClass extends Thread {
2     public FromThreadClass(String id) {
3         super(id);
4     }
5
6     public void run() {
7         System.out.println("[Thread " + this.getName() + "] Run");
8     }
9 }
```

```
1 public class FromProcessClass {
2     public static void main(String[] args) {
3         FromThreadClass th1 = new FromThreadClass("TH1");
4         th1.start();
5     }
6 }
```

[Thread TH1] Run

IMPLÉMENTER L'INTERFACE `Runnable`

```
1 public class FromRunnableInterface implements Runnable {
2     private String name ;
3     public FromRunnableInterface(String n){
4         this.name = n;
5     }
6
7     public void run() {
8         System.out.println("[Thread "+ this.name +"] Run");
9     }
10 }
```

```
1 public class FromProcessClass {
2     public static void main(String[] args) {
3         Thread th1 = new Thread(new FromRunnableInterface("TH1"));
4         th1.start();
5     }
6 }
```

[Thread TH1] Run

PLAN

- Systèmes multitâches
- Les processus et les threads en Java
- Quelques méthodes de la classe Thread

[Retour au plan](#) - [Retour à l'accueil](#)

LES MÉTHODES `sleep()` ET `join()`

```
1 public class ThreadFils extends Thread {
2     public ThreadFils(String name) {
3         super(name);
4     }
5
6     public void run() {
7         try {
8             System.out.println("[Thread "+this.getName()+" Debut");
9             System.out.println("[Thread "+this.getName()+" Sleep ...");
10            this.sleep(2000);
11            System.out.println("[Thread "+this.getName()+" Fin");
12        } catch (InterruptedException e) {
13            e.printStackTrace();
14        }
15    }
16 }
```

LES MÉTHODES `sleep()` ET `join()`

```
1 public class Main {  
2     public static void main(String[] args) {  
3         try {  
4             System.out.println("[Main Process] Debut");  
5             ThreadFils f1 = new ThreadFils("f1");  
6             f1.start();  
7             f1.join();  
8             System.out.println("[Main Process] Fin");  
9         } catch (InterruptedException e) {  
10             e.printStackTrace();  
11         }  
12     }  
13 }
```

```
[Main Process] Debut  
[Thread f1] Debut  
[Thread f1] Sleep ...  
[Thread f1] Fin  
[Main Process] Fin
```

LES MÉTHODES `sleep()` ET `join()`

```
1 public class Main {  
2     public static void main(String[] args) {  
3         try {  
4             System.out.println("[Main Process] Debut");  
5             ThreadFils f1 = new ThreadFils("f1");  
6             f1.start();  
7             f1.join(1000);  
8             System.out.println("[Main Process] Fin");  
9         } catch (InterruptedException e) {  
10             e.printStackTrace();  
11         }  
12     }  
13 }
```

```
[Main Process] Debut  
[Thread f1] Debut  
[Thread f1] Sleep ...  
[Main Process] Fin  
[Thread f1] Fin
```

LA MÉTHODE `interrupt()`

```
1 public class Main {  
2     public static void main(String[] args) {  
3         try {  
4             System.out.println("[Main Process] Debut");  
5             ThreadFils f1 = new ThreadFils("f1");  
6             f1.start();  
7             f1.join(1000);  
8             f1.interrupt();  
9             System.out.println("[Main Process] Fin");  
10        } catch (InterruptedException e) {  
11            e.printStackTrace();  
12        }  
13    }  
14 }
```

[Main Process] Debut

[Thread f1] Debut

[Thread f1] Sleep ...

[Main Process] Fin

```
java.lang.InterruptedException: sleep interrupted  
at java.lang.Thread.sleep(Native Method)
```

```
at ThreadFils.run(ThreadFils.java:15)
```

MERCI

[Retour à l'accueil](#) - [Retour au plan](#)