# Software Distribuït - T5 - Threads

Eloi Puertas i Prats

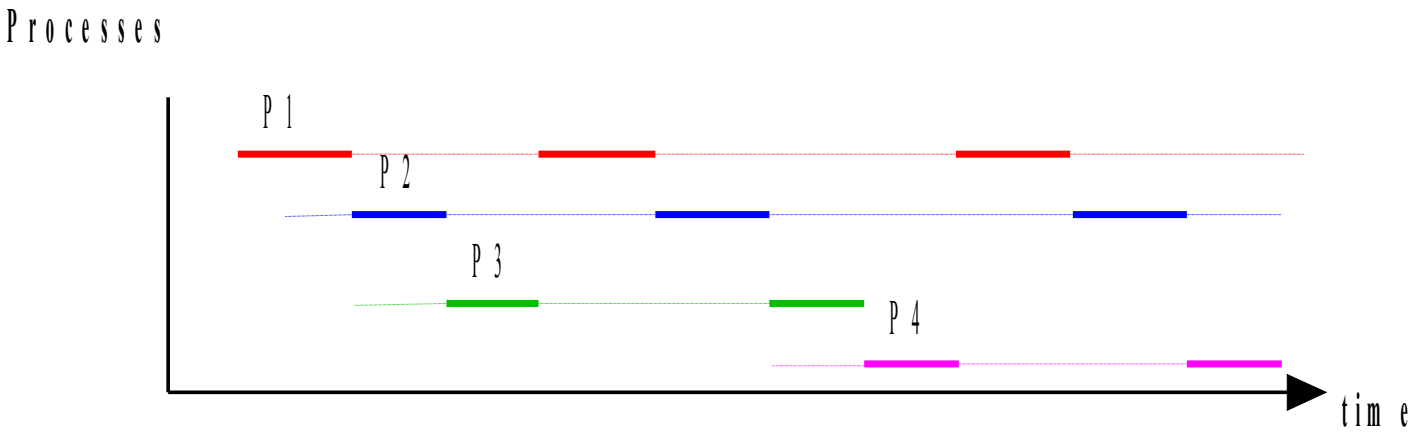
Universitat de Barcelona
Grau en Enginyeria Informàtica

6 de març de 2024

UNIVERSITAT DE BARCELONA

# Processos concurrents

Els Sistemes Operatius d'avui en dia, poden fer com si múltiples processos s'executin concurrentment en una sola CPU, fent que els recursos siguin de temps compartit entre els processos.

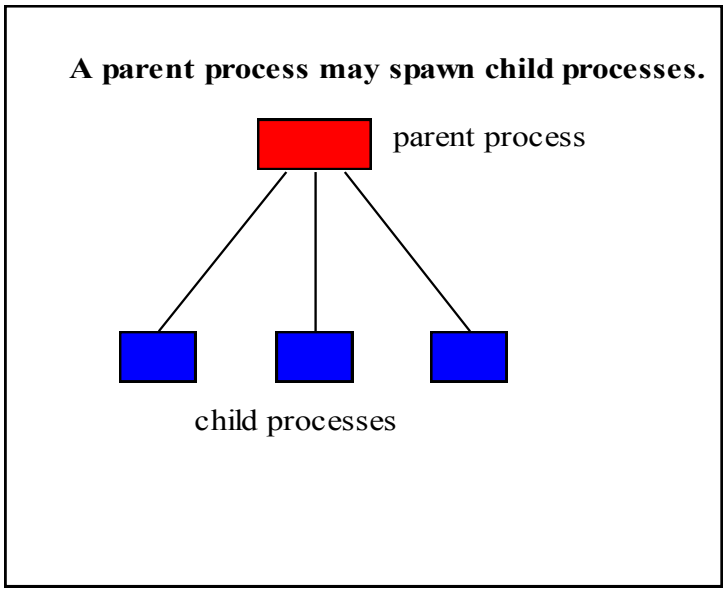


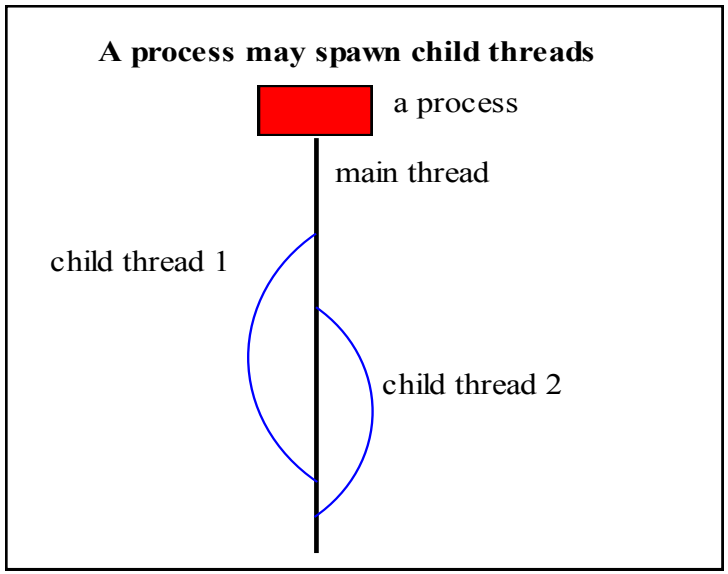
# Perquè són útils els threads

- Aprofitament de múltiples processadors.
- Simplicitat del modelat.
- Gestió más sencilla d'events asíncrons.
- Interfícies d'usuari amb millor resposta.

# Fils d'execució concurrents dins d'un procés. Threads.

Un procés pot tenir fils d'execució paral.lels (Threads). Tots els fils estan compartint els mateixos recursos dins del procés.




**Concurrent processing within a process**

# Threads a Java

La màquina virtual de java pemet que una aplicació tingui múltiples fils d'execució, executant-se concurrentment.
Java proporciona una classe Thread:

- `public class Thread extends Object implements Runnable`

Quan s'engega una JVM, normalment existeix un sol fil d'execució. Es continua executant els fils fins que:

- S'ha cridat al mètode exit() de la classe Runtime
- Tots les threads han acabat, ja sigui perquè han retornat de la crida del mètode run o perquè han llençat una excepció que s'ha propagat més enllà del mètode run.

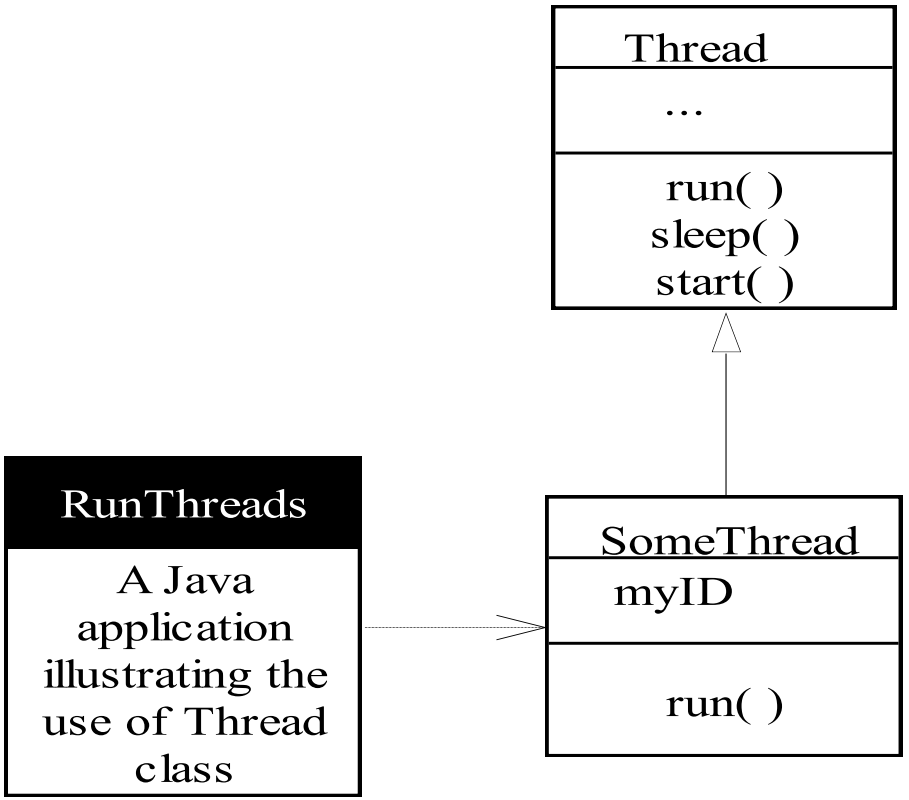
# Creació de Threads en Java

- Creant una classe que heredi de la classe **Thread**
- Creant una classe que implementi la interfície **Runnable** i passant-la com a paràmetre al constructor de la classe **Thread**

UNIVERSITAT DE BARCELONA

# Extenent un Thread en java

Declarar una class que sigui una subclasse de **Thread**. Aquesta subclasse ha de sobrescriure el mètode **run()** de la classe Thread. Llavors es pot declarar una instància d'aquesta subclasse i començar la seva execució amb el mètode **start()**



# Extenent un Thread en java

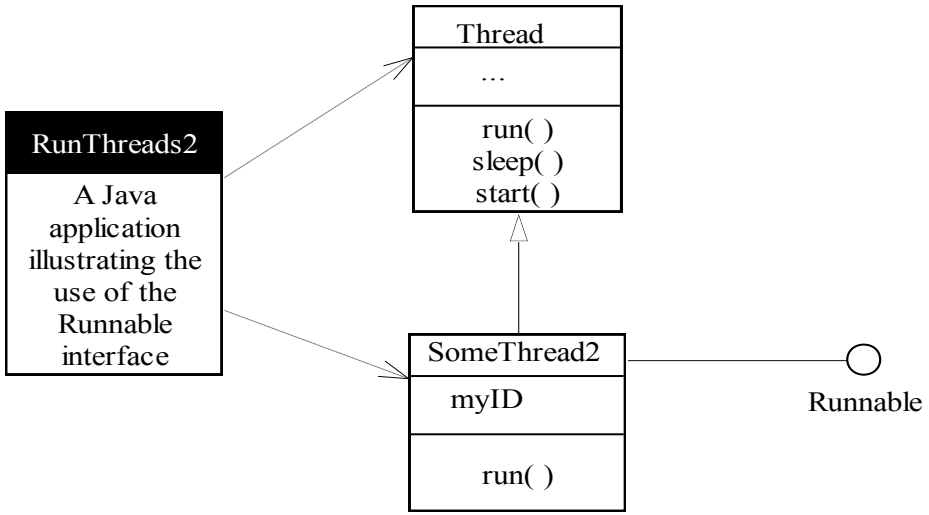
```
public class SimpleThread extends Thread {
    public SimpleThread(String str) {
        super(str);
    }
    public void run() {
        for (int i = 0; i < 10; i++) {
            System.out.println(i + " " + getName());
            try {
                sleep((long)(Math.random() * 1000));
            } catch (InterruptedException e) {}
        }
        System.out.println("DONE! " + getName());
    }
}
```

# Engegant dos threads que extenen de Thread

```
public class TwoThreadsDemo {
    public static void main (String[] args) {
        new SimpleThread("Jamaica").start();
        new SimpleThread("Fiji").start();
    }
}
```

# Implementat un Runnable i associar-lo amb un Thread en java

Una altra forma de declarar un Thread és declarant una classe que implementi la interfície **Runnable**. Llavors, aquest classe ha d'implementar forçosament el mètode **run()**. Finalmment, una instància d'aquesta classe es passa com a argument quan es crea el nou Thread, i s'engega amb el mètode **start()**.



# Implementat un Runnable

```
class SimpleThread implements Runnable {
 public String name = "";
 public SimpleThread(String str) {
          name = str; }
 public String getName(){
          return name; }
 public void run() {
        for (int i = 0; i < 10; i++) {
              System.out.println(i + "■" + this.getName())
              try {
               Thread.sleep((long)(Math.random() * 1000));
              } catch (InterruptedException e) {} }
        System.out.println("DONE!■" + this.getName());
 }}
```

# Engegant dos threads que implementen Runnabe

```
public class TwoThreadsDemo {
    public static void main (String[] args) {
        new Thread(new SimpleThread("Jamaica")).start();
        new Thread(new SimpleThread("Fiji")).start();
    }
}
```

# Resultat de l'Execució

Code Source TwoThreadsDemo.java

UNIVERSITAT DE BARCELONA