Programació de Serveis
i processos

UD 02. Programació Multifil

Threads en Java

IES Jaume II El Just
Tavernes de la Valldigna
Departament d'Informàtica
Curs 2021-22

Continguts

1 Introducció

Els threads, fils d'execució o processos lleugers, són les unitats més menudes de processament que poden ser programades pels sistemes operatius, i que permeten a un mateix procés, executar diferents tasques de forma simultània. Cada fil d'execució executa una tasca concreta, i ofereix així al programador la possibilitat de dissenyar programes que executen funcions diferents de forma concurrent.

Aquesta tècnica de programació amb fils es coneix com multithreading o multifil, i permet simplificar el disseny d'aplicacions concurrents i millorar el rendiment en la creació de processos.

Recordeu que la principal diferència entre els processos i els fils és que els fils d'un mateix procés comparteixen els mateixos recursos (memòria, fitxers oberts…). Així doncs, quan creem diversos objectes d'una mateixa classe, pot que diversos fils d'execució accedisquen a una dada o objecte concret. Diem, en aquest cas, que **la dada o objecte està en conflicte**, i a la secció de codi que accedeix a aquestes dades o objectes, li diem **secció crítica**.

Alguns dels exemples més habituals a les que es fa ús de fils d'execució, poden ser les aplicacions multimèdia i les aplicacions client-servidor.

2 Threads en Java

2.1 Creació de threads

Per tal de crear threads en Java podem optar per dues opcions:

- Crear **una classe que herete de la classe Thread (`java.lang.Thread`)**, o bé
- crear una classe que **implemente la interfície Runnable (`java.lang.Runnable`)**.

L'**opció aconsellada** a la documentació de Java és fer ús de la **interfície Runnable**. Ambdós mecanismes són pràcticament equivalents, amb la diferència que si fem la nostra classe descendent de `Thread`, per les característiques de Java, aquesta no pot heretar d'altra classe, pel que si volem que la nostra classe pertanga a certa jerarquia i a més tinga les característiques dels Threads, hem d'utilitzar interfícies.

Per crear un thread mitjançant la implementació de `Runnable`, farem el següent:

1. Definir una classe que implemente la interfície `Runnable` i el mètode `public void run()`, amb la funcionalitat que s'haurà d'executar en el fil:

```
class classeThreadable implements Runnable{
        // Declaració d'atributs i mètodes

        @Override
        public void run() {
                // Funcionalitat a realitzar
                }
        }
}
```

2. Crear un thread a partir de la classe anterior:

```
Thread fil=new Thread(classeThreadable);
```

El més habitual serà emmagatzemar els diferents threads en un vector en lloc de fer-ho en un objecte solt:

```
// Creem el vector vetor_fils, per emmagatzemar els diferents fils d'execució
Thread vector_fils[]=new Thread [longitud];
...
// I després creem els diferents threads en la corresponent posició del
 ↪  vector:
Thread fil=new Thread(classeThreadable);
vector_fils[posicio_i]=fil;     //  Sent i un índex qualsevol del vector
```

3. Llançar el thread amb `start()` i esperar-lo amb `join()`:

```
    fil.start();
    // O bé, si el tenim al vector
    vector_fils[posicio].start();
    // ...
    fil.join();
    // O bé
    vector_fils[posicio].join();
```

2.1.1 start() i run()

Si ens adonem, per crear el thread, hem fet ús del mètode `start()`, i la classe que implementa Runnable, defineix un mètode `run()`. Quina relació hi ha entre estos dos mètodes?

El mètode `run()`, que és el que definim nosaltres a la classe, conté el codi que volem executar *asíncronament* en un thread que llançarem posteriorment. Si invocàrem `run()` directament, el mètode s'executaria de forma síncrona (en el mateix thread en què estem), mentre que si la nostra és una subclasse de `Thread` o implementa `Runnable`, el mètode `start()` ens crea el fil i fa que en un moment donat, el fil d'execució llance el mètode `run()` de forma asíncrona, tornant de seguida el control al fil actual, i deixant el nou thread executant el que hi havera al codi de `run()` fins que aquest acabe.

2.1.2 join()

Parem especial atenció al `join`. Tots els programes multifil tenen un fil principal que ha d'esperar que acaben els fils associats, per a la qual cosa utilitzem el mètode `join`. Pe què s'anomena així? Quan creem un thread, es produeix una bifurcació, es creen dos *camins* d'execució diferents, pel que quan un procés acaba i ha d'esperar a l'altre, es produeix una unificació, un *join*.

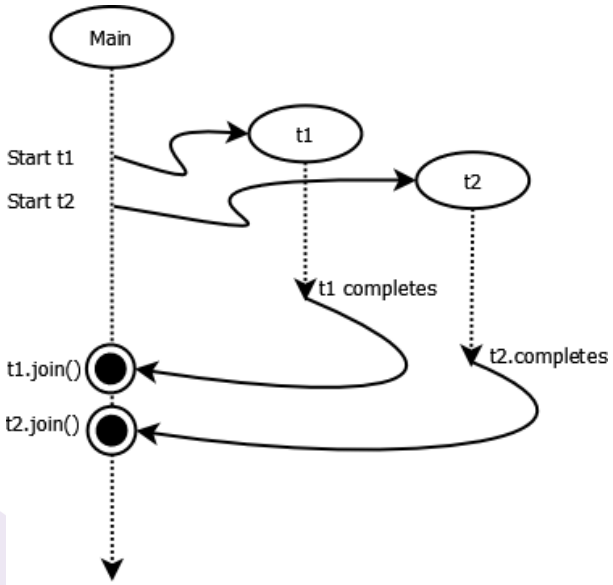

Figura 1: Exemple de join

2.1.3 Altres mètodes de la class thread

Veiem a la següent taula una síntesi dels mètodes comentats i altres mètodes utilitzats en la gestió de threads:

Mètode	Descripció
`long getId()`	Retorna l'identificador del thread, consistent en un enter llarg positiu generat automàticament en la creació del fil. Aquest número és únic i inalterable durant el cicle de vida del thread
`void setName(String Nom)`	Permet donar-li un nom al thread
`String getName(String Nom)`	Retorna el nom que li hem assignat al thread
`Thread currentThread()`	Retorna un objecte de la classe Thread que representa el fil d'execució actual.
`void setPriority(int prioritat)`	Permet establir la prioritat del fil, de forma indicativa, ja que el sistema operatiu no està obligat a respectar-la. La prioritat màxima i mínima venen donades per les constants `MAX_PRIORITY` i `MIN_PRIORITY`.
`int getPriority()`	Retorna la prioritat del fil
`void Thread.sleep(long ms)`	Permet deixar el procés en suspensió durant un temps concret
`void join()`	Finalitza el fil d'execució, tornant el control al fil principal que va llençar el fil secundari, amb la possibilitat d'escollir el temps d'espera en mil·lisegons
`void yield()`	Torna a un procés a l'estat de *llest per executar*, de manera que permet que passen a executar-se altres fils de la mateixa prioritat (però no posa en pausa el fil)
`void interrupt()`	Interromp l'execució del fil
`boolean interrupted()`	Retorna si el fil actual ha estat interromput
`boolean isAlive()`	Retorna cert si el fil està viu
`public Thread.State getState()`	Retorna l'estat del fil, que pot ser `NEW`, `RUNNABLE`, `BLOCKED`, `WAITING`, `TIMED_WAITING`, `TERMINATED`
`String toString()`	Heratat d'Object, retorna una cadena amb una representació del thread, incloent-hi el nom, la prioritat i el grup

Podem trobar més informació sobre la interfície Runnable i la classe Thread i la resta de mètodes

relacionats amb ella a la documentació de l'OpenJDK:

- https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/java.base/java/lang/Thread.html.
- https://cr.openjdk.java.net/~iris/se/11/latestSpec/api/java.base/java/lang/Runnable.html

2.2 Exemples

2.2.1 Exemple 1

El següent programa exemple1.java crea una classe que rep un nom en el constructor i l'emmagatzema, i que implementa un mètode `run()` que obté informació sobre el thread actual (l'id, el nom del thread i l'atribut *nom* de la classe -que no són el mateix-), i els mostra per pantalla. Finalment, el mètode `main()` crea tres objectes i tres fils diferents i els llança en paral·lel.

```
public class exemple1 implements Runnable {
    /*
        Exemple tipus "Hola Món" amb threads
    */

    String nom;
    // Constructors
        exemple1(){ this.nom="Anònim"; }
        exemple1(String nom){this.nom=nom; }

    @Override
    public void run() {
        // Aquest és el mètode que s'exeutarà quan
        // s'invoque al mètode start del thread.

        try{
            // Agafem la referència al thread actual
            Thread filActual=Thread.currentThread();

            // I imprimim informació sobre el seu nom i algunes propietats
            System.out.println("Hola Món dels threads. Sóc "+this.nom+":"
            +"\n\tEl meu id de thread és "+filActual.getId()
            +"\n\tEl nom de thread és "+filActual.getName()
            +"\n\tLa meua prioritat és "+filActual.getPriority()+"\n");


        }  catch (Exception e) {
```

```
            e.printStackTrace();
        }
    }


    public static void main(String[] args) {
        try{
            // Creem alguns objectes d'exemple
            exemple1 ex1=new exemple1("C3PO");
            exemple1 ex2=new exemple1("R2D2");
            exemple1 ex3=new exemple1("BB8");

            // I els fils corresponents
            Thread fil1=new Thread(ex1);
            Thread fil2=new Thread(ex2);
            Thread fil3=new Thread(ex3);

            // Llancem els fils
            fil1.start();
            fil2.start();
            fil3.start();

            // I els juntem amb l'actual quan acaben
            fil1.join();
            fil2.join();
            fil3.join();

        }catch(Exception e){
                e.printStackTrace();
        }
    }

}
```

> **Quina és la diferència entre `filActual` i `this`**
>
> *Per què a la línia 21 mostrem el nom amb `this.nom` i després, a la línia 23, fem ús de `filActual.getName()`?*
>
> Si no ens parem a pensar un poc, podem confondre la referència `this` de l'objecte amb el propi fil d'execució, quan no es tracta del mateix. Tingueu en compte que una cosa és el propi objec-

> te, el qual hem instanciat i li hem donat un nom, i altra diferent és el *thread* que es crea quan invoquem al mètode `run()` a través de l'`start()`. Amb `this.nom` tindrem accés al nom de l'objecte que li hem donat, mentre que amb `filActual.getName()` obtenim el nom que el sistema li ha donat a aquest thread.

Tenint en compte això, entendrem millor l'eixida del programa:

```
$ java exemple1
Hola Món dels threads. Sóc BB8:
    El meu id de thread és 13
    El nom de thread Thread-2
    La meua prioritat 5

Hola Món dels threads. Sóc C3PO:
    El meu id de thread és 11
    El nom de thread Thread-0
    La meua prioritat 5

Hola Món dels threads. Sóc R2D2:
    El meu id de thread és 12
    El nom de thread Thread-1
    La meua prioritat 5
```

2.2.2 Exemple 2

En aquest exemple, hem fet una variació de l'anterior, per tal que reba una llista de noms com a arguments, i cree tants fils com hem indicat, fent ús de vectors per emmagatzemar els objectes i els threads.

```
public class exemple2 implements Runnable {
    /*
        Exemple tipus "Hola Món" amb threads i vectors
    */

    String nom;
    // Constructors
    exemple2(){ this.nom="Anònim"; }
    exemple2(String nom){this.nom=nom; }
```

```
    @Override
    public void run() {
        // Aquest és el mètode que s'exeutarà quan
        // s'invoque al mètode start del thread.

        try{
            // Agafem la referència al thread actual
            Thread filActual=Thread.currentThread();

            // I imprimim informació sobre el seu nom i algunes propietats
            System.out.println("Hola Món dels threads. Sóc "+this.nom+":"
            +"\n\tEl meu id de thread és "+filActual.getId()
            +"\n\tEl nom de thread és "+filActual.getName()
            +"\n\tLa meua prioritat és "+filActual.getPriority()+"\n");


        }  catch (Exception e) {
            e.printStackTrace();
        }
    }


    public static void main(String[] args) {
        try{

            // Definim un vector per emmagatzemar els threads
            Thread[] LlistaThreads=new Thread[args.length];

            for (int i=0;i<args.length;i++){
                // Creem l'objecte
                exemple2 nom=new exemple2(args[i]);
                // I el thread que l'utilitza
                LlistaThreads[i]=new Thread(nom);
                // I el llancem
                LlistaThreads[i].start();
            }

            // Una vegada llançats, els juntem tots
            for (int i=0;i<args.length;i++){
                LlistaThreads[i].join();
            }
```

```
        }catch(Exception e){
                e.printStackTrace();
        }
    }

}
```

En aquest cas, l'eixida serà:

```
$ java exemple2 Leia Luke Han Chewie
Hola Món dels threads. Sóc Luke:
    El meu id de thread és 12
    El nom de thread és Thread-1
    La meua prioritat és 5

Hola Món dels threads. Sóc Han:
    El meu id de thread és 13
    El nom de thread és Thread-2
    La meua prioritat és 5

Hola Món dels threads. Sóc Leia:
    El meu id de thread és 11
    El nom de thread és Thread-0
    La meua prioritat és 5

Hola Món dels threads. Sóc Chewie:
    El meu id de thread és 14
    El nom de thread és Thread-3
    La meua prioritat és 5
```

2.2.3 Exemple 3

En el següent exemple, veiem la classe que implementa el main pot ser diferent a la classe que implementa Runnable. A més, en aquest exemple, hem fet ús també del sleep, per fer una xicoteta pausa abans que acabe el thread, i mostrar un missatge de comiat:

```
class Saluda implements Runnable {
    /*
        Exemple tipus "Hola Món" amb threads i vectors
```

```
    */

    String nom;
    // Constructors
    Saluda(){ this.nom="Anònim"; }
    Saluda(String nom){this.nom=nom; }

    @Override
    public void run() {
        // Aquest és el mètode que s'exeutarà quan
        // s'invoque al mètode start del thread.

        try{
            // Agafem la referència al thread actual
            Thread filActual=Thread.currentThread();

            // I imprimim informació sobre el seu nom i algunes propietats
            System.out.println("Hola Món dels threads. Sóc "+this.nom+":"
            +"\n\tEl meu id de thread és "+filActual.getId()
            +"\n\tEl nom de thread és "+filActual.getName()
            +"\n\tLa meua prioritat és "+filActual.getPriority()+"\n");

            filActual.sleep(1000);
            System.out.println("Eixint de "+this.nom);


        }  catch (Exception e) {
            e.printStackTrace();
        }
    }
}

public class exemple3 {
    public static void main(String[] args) {
        try{

            // Definim un vector per emmagatzemar els threads
            Thread[] LlistaThreads=new Thread[args.length];

            for (int i=0;i<args.length;i++){
                // Creem l'objecte
                Saluda nom=new Saluda(args[i]);
```

```
                // I el thread que l'utilitza
                LlistaThreads[i]=new Thread(nom);
                // I el llancem
                LlistaThreads[i].start();
            }

            // Una vegada llançats, els juntem tots
            for (int i=0;i<args.length;i++){
                LlistaThreads[i].join();
            }


        }catch(Exception e){
                e.printStackTrace();
        }
    }

}
```

L'eixida és:

```
$java exemple3 Leia Luke Han Chewie
Hola Món dels threads. Sóc Han:
    El meu id de thread és 13
    El nom de thread és Thread-2
    La meua prioritat és 5

Hola Món dels threads. Sóc Luke:
    El meu id de thread és 12
    El nom de thread és Thread-1
    La meua prioritat és 5

Hola Món dels threads. Sóc Chewie:
    El meu id de thread és 14
    El nom de thread és Thread-3
    La meua prioritat és 5

Hola Món dels threads. Sóc Leia:
    El meu id de thread és 11
    El nom de thread és Thread-0
    La meua prioritat és 5
```

```
Eixint de Leia
Eixint de Luke
Eixint de Chewie
Eixint de Han
```

Si l'executem, comprovarem com entre l'últim Hola Mon i els missatges de "Eixint de…" passa una estona.