



Département: Mathématiques & Informatique

Multithreading

Licence Professionnelle : Développement Informatique

Pr: Youssef Ouassit

Plan du chapitre

- Introduction
 - Définition
 - Raison d'être
- Création de Thread
 - Par implémentation
 - Par héritage
- Gestion de Thread
 - Méthodes de gestion
 - Diagrammes d'état

Introduction

Qu'est-ce qu'un Thread?

- Un ordinateur qui exécute un programme :
 - Possède un CPU
 - Stocke le programme à exécuter
 - Possède une mémoire manipulée par le programme
 - ➔ Multitasking géré par l'OS
- Un thread (« file ») a ces mêmes capacités
 - A accès au CPU
 - Gère un processus
 - A accès à la mémoire, qu'il partage avec d'autres files
 - ➔ Multithreading géré par la JVM

Introduction

Pourquoi le multithreading?

- Un programme moderne est composé de
 - Une interface graphique
 - Quelques composantes pouvant agir de manière autonome
- Sans multithreading
 - Les composantes ne pourraient agir que lorsque l'interface est suspendue
- Plus généralement, le multithreading
 - Permet de réaliser plusieurs processus indépendants en parallèle
 - Permet de gérer les files d'attente
 - Permet au besoin de synchroniser les différents processus entre eux

Création de Thread

Mise en œuvre (1/3)

- Par implémentation de l'interface
 - Usage
 - ➔ `public void MaClasse implements Runnable`
 - Avantages et inconvénients
 - ☺ Meilleur sur le plan orienté objet
 - ☺ La classe peut hériter d'une autre classe
 - ☺ Consistance
- Par héritage de la classe Thread elle-même
 - Usage
 - ➔ `public void MaClasse extends Thread`
 - Avantages et inconvénients
 - ☺ Code simple (l'objet est un Thread lui-même)
 - ☹ La classe ne peut plus hériter d'une autre classe

Création de Thread

Mise en œuvre (2/3)

```
public class MaFile implements Runnable {
    public void run(){
        byte[] buffer=new byte[512];
        int i=0;
        while(true){
            if(i++%10==0)System.out.println(""+i+" est divisible par 10");
            if (i>101) break;
        }
    }
}

public class LanceFile {
    public static void main(String[]arg){
        Thread t=new Thread(new MaFile());
        t.start();
    }
}
```



Le constructeur de la classe Thread attend un objet Runnable en argument

Création de Thread

Mise en œuvre (3/3)

```
public class MyThread extends Thread {  
    public void run(){  
        byte[] buffer=new byte[512];  
        int i=0;  
        while(true){  
            if(i++%10==0) System.out.println(""+i+" est divisible par 10");  
            if(i>101) break;  
        }  
    }  
}  
  
public class LaunchThread{  
    public static void main(String[]arg){  
        MyThread t=new MyThread();  
        t.start();  
    }  
}
```

Grâce à l'héritage, un objet de type MyThread est lui-même Runnable, on peut donc appeler un constructeur sans argument



Création de Thread

Méthodes de gestion (1/4)

- `t.start()`
 - Appeler cette méthode place le thread dans l'état "runnable"
➔ Eligible par le CPU
- `t.yield()` throws `InterruptedException`
 - La VM arrête la file active et la place dans un ensemble de files activables. (runnable state)
 - La VM prend une file activable et la place dans l'état actif (running state)
- `t.sleep(int millis)` throws `InterruptedException`
 - La VM bloque la file pour un temps spécifié (état « d'attente »)
- `t.join()` throws `InterruptedException`
 - Met la file en attente jusqu'au moment où la file `t` est terminée (a fini sa méthode `run()`). Le thread appelant redevient alors activable.

Gestion des Thread

Méthodes de gestion (2/4)

- `Thread.yield()` throws `InterruptedException`
 - La VM arrête la file active et la place dans un ensemble de files activables. (runnable state)
 - La VM prend une file activable et la place dans l'état actif (running state)
- `Thread.sleep(int millis)` throws `InterruptedException`
 - La VM bloque la file pour un temps spécifié (état « d'attente »)
- `t.join()` throws `InterruptedException`
 - Met la file en attente jusqu'au moment où la file t est terminé (a fini sa méthode `run()`). Le thread appelant redevient alors activable.

Gestion des Thread

Méthodes de gestion (3/4)

```
public class ExJoin {  
    public static void main(String[]arg){  
        Thread t=new Thread(new FileSecondaire());  
        t.start();  
        for(int i=0;i<20;i++){  
            System.out.println("File principale en cours d'exécution "+i);  
            try{  
                Thread.sleep(10);  
            } catch (InterruptedException ie){}  
        }  
        try {  
            t.join();  
        } catch (InterruptedException ie) {}  
        System.out.println("t termine son exécution, fin programme");  
    }  
}
```

Gestion des Thread

Méthodes de gestion (4/4)

```
class FileSecondaire implements Runnable{
    public void run(){
        for(int i=0;i<40;i++){
            System.out.println("File secondaire en execution "+i);
            try{
                Thread.sleep(10);
            } catch (InterruptedException ie) {}
        }
        System.out.println("Fin de file secondaire");
    }
}
```

Gestion des Thread

Diagrammes d'état

