CONCEPTES AVANÇATS DE SISTEMES OPERATIUS
Departament d'Arquitectura de Computadors

Aplicacions client-servidor en Java.

(Seminaris de CASO)

Autors

José Manuel Pabón Mansilla
Josá Antonio Escartín Vigo
Eduard Tomás Pérez

Objectiu de la presentació.

- L'objectiu d'aquesta presentació és mostrar com es poden aplicar una sèrie de coneixements adquirits a les classes de teoria sota un llenguatge de programació diferent en molts sentits, Java.

- Es proporcionaran una sèrie d'exemples pràctics per tal de que la seva comprensió sigui ràpida i senzilla.

Pròleg

- Les comunicacions en Java es dissenyen com aplicacions client servidor. Els principals components característics d'aquestes aplicacions que mostrarem en la nostra exposició son:
  - Sockets.
  - Threads.
  - Monitors.
  - Sincronització.

Sockets

- A diferència de la crida de Unix socket, Java ens proporciona una interfície diferent per al client i per al servidor, distingint entre dos tipus de sockets:
  - ServerSockets , per al servidor.
  - Sockets, per al client.

- Aquests dos tipus de Sockets són objectes, i per tant caldrà instanciar-los.

Sockets de Client: Interfície

- **Socket** (InetAddress address, int port)
  - És la forma més usual.
  - Adreça a la que es connecta i per quin port.
- **Socket**()
  - Crea un socket sense cap paràmetre addicional.
- **Socket**(String host, int port)
  - Com l'anterior però té per paràmetre el nom del host.
- **connect**(SocketAddress endpoint).
  - Només és necessari si hem creat el socket amb la segona creadora.

ServerSockets:Interfície

- **ServerSocket**(int port)
  - Amb un 0 s'agafa qualsevol port per defecte
- **ServerSocket**(int port, int backlog)
  - Backlog és equivalent al listen, defineix la cua de peticions màxima abans de refusar connexions.
- **Socket accept**()
  - Ens retorna un nou socket per tractar amb el client.

Comunicació via sockets

- Per comunicar-nos amb els sockets necessitem canals de entrada i/o sortida que haurem de demanar explícitament.
  - Socket sk;
  - InputStream skin = sk.getInputStream();
  - OutputStream skout = sk.getOutputStream();
- També ens calen uns filtres per passar les dades a un format unificat de transmissió (UTF).
  - DataInputStream dis = new DataInputStream(skin);
  - String st = dis.readUTF( );

Intruducció als Threads.

- Java ens ofereix dues maneres de crear els Threads.
  - Extensió de la classe Thread.
  - Implementació de la classe Runnable.
- La classe Thread no és res més que una implementació de Runnable amb més de 35 mètodes.
  - Principalment només ens interessaran dos:
    - run( ) : és el codi que cal implementar amb el nostre algoritme.
    - start( ): és el mètode que cridarem quan vulguem executar el fil.
    - stop( ): acaba un Thread explícitament.(No fer-la servir).
    - isAlive( ): ens diu si un thread ha acabat.

La classe Thread

- Constructores per a la classe:
  - **Thread( )**
    - **Per a estendre el thread?.**
  - **Thread**(Runnable target)
  - **Thread**(Runnable target, String name)
- Existeixen els ThreadGroup que ens serveixen per tenir un control de un conjunt de threads.
  - **Thread**(ThreadGroup group, Runnable target)
  - **Thread**(ThreadGroup group, Runnable target, String name, long stackSize)

Exemple de Thread (I)

```
class ThreadUnic extends Thread {
        public SimpleThread(String str) {
                setName(str);
        }
        public void run() {
                for (int i = 0; i < 10; i++) {
                        System.out.println(i + " " + getName());
                        try {  sleep((int)(Math.random() * 1000));
                        } catch (InterruptedException e) { // Tractar excepció }
                }
                System.out.println("Finalitza: " + getName());
        }
}
```

Exemple de Thread (II)

```
class DosThreads {
    public static void main (String[] args) {
        new SimpleThread(”Primer").start();
        new SimpleThread(”Segon").start();
    }
}
```

- Exectutant-lo veuriem que l’ordre de finalització és indeterminat.
- Aquí hem vist un exemple com a extensió de la classe Thread. Ara en veurem un altre com implementació de runnable.

Exemple de Thread (III)

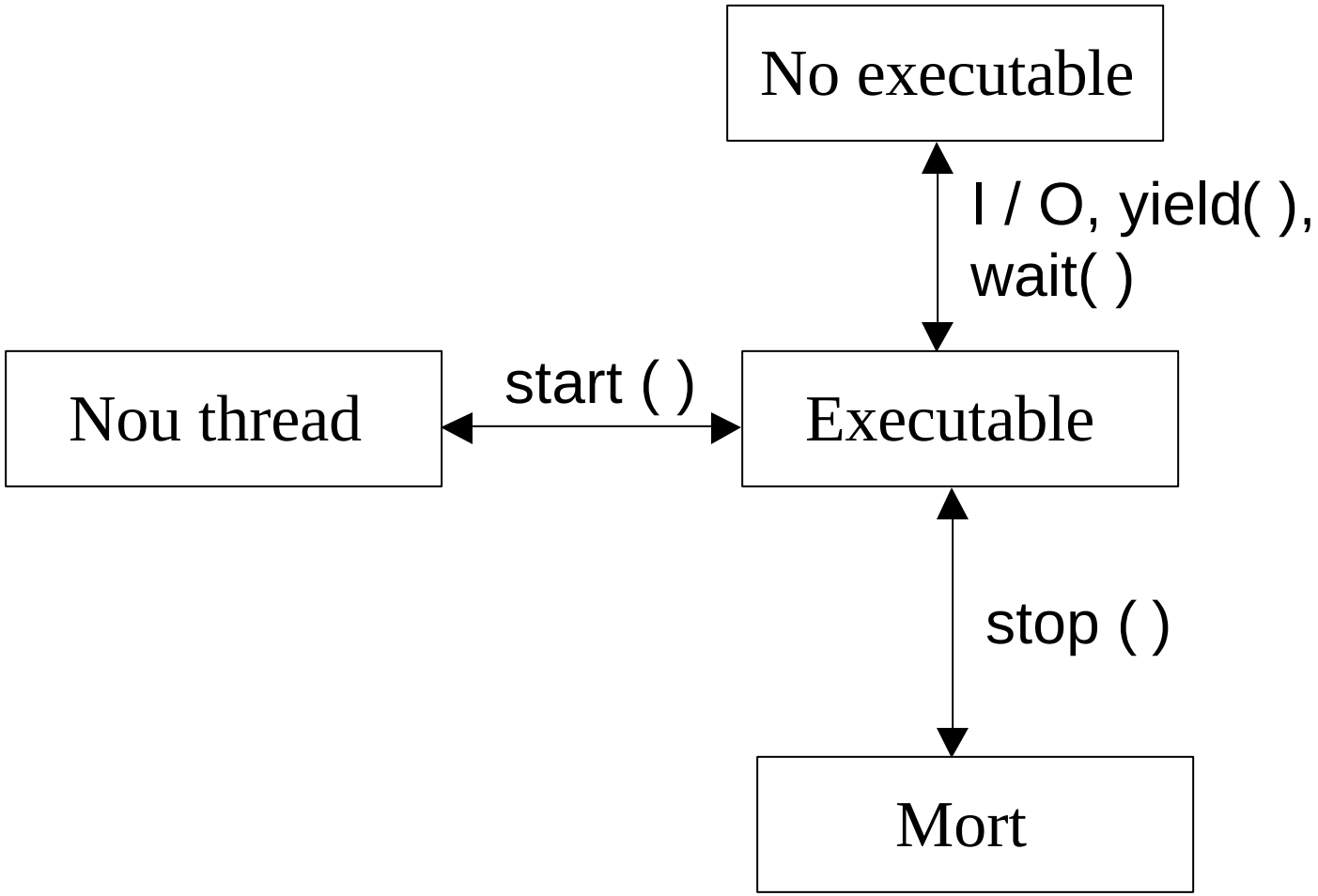
```
Class PingPong implements Runnable{
    String frase;
    int retard;
    PingPong ( String que, int temps ) {
        frase = que;
        retard = temps;
     }
    public void run(){
        try{
            for ( int i = 0; i < 30; i++){
            System.out..println( word + " ");
         }catch( InterruptedException e) { return;}
    }
}
```

Exemple de Thread (IV)

```
Class principal{

    public static void main ( String[ ] args){
        Runnable ping = new PingPong(“ping”,33);
         new Thread(ping).start();
        Runnable pong = new PingPong(“pong”,100);
         new Thread(pong).start();
    }
}
```

Esquema de l'execució d'un thread



Monitors

- Importancia de l'us dels monitors

  – Necesitat d'alguns threads de compartir dades

  (els threads hauran de tenir en compte l'estat dels altres threads amb els que interactua)

    • Ex. Acces a bases de dades, variables globals, etc.

  – Obtenim un model Productor/consumidor

Monitors (II)

- Concepte de monitor
  - Classe dissenyada especificament pel programador que controla l'accés a unes dades per part de diferents threads.

- Objectiu:
  - Conservar la consistència de les dades i establir un protocol d'accés.

Sincronització

- Metodes que ens ofereix Java:
  - wait( ): adorm l'execució del fil.
  - notify( ): desperta un altre fil (i només un)
  - notifyAll( ): desperta la resta de fils adormits
- Hem de vigilar perque Java no controla les abraçades mortals

Sincronització

```
class Monitor{

        private int elem;
        private boolean llegible = false;
```

```
public synchronized void SetEnter (int
val) {
      while(llegible){
            try{ wait(); }
            catch (Exception e){ }
      }
      Enter = val;
      llegible = true;
      notify();
}
```

```
public synchronized int GetEnter(){
      int enter;
      while(!llegible){
            try{wait();}
            catch (Exception e){}
      }
      llegible = false;
      notify();
      return enter;
}
```

Bibliografia

- Pàgina web de la api de java.
  - http://java.sun.com/j2se/1.4.2/docs/api/.
- Curs practic d'aprenentatge
  - http://programacion.com/java/tutorial/threads/
- Assignatura de programacio concurrent
  - http://www-dse.doc.ic.ac.uk/concurrency/