Programación concurrente, paralela y distribuida

Se llama así a la programación que tiene en cuenta la existencia de distintos procesos en el mismo instante de tiempo.

Dos procesos son concurrentes cuando entre la primera y última unstrucción de uno de ellos se ejecutan instrucciones del otro proceso.

Se produce concurrencia entre procesos cuando hay un solapamiento temporal en la ejecución de instrucciones de los dos procesos.

El solapamiento no implica simultaneidad, aunque la incluye . La simultaneidad se produce cuando hay instrucciones de dos procesos ejecutándose a la vez, en el mismo tiempo. La existencia de procesos en el mismo instante de tiempo (concurrencua) no necesita de la sumultaneidad de ejecución de instrucciones .

Cuando hay concurrencia entre procesos (sea simultánea o no) se pueden dar situaciones de colaboración entre procesos para conseguir un mismo fin, o competencia entre procesos por los recursos .

Incluso se pueden dar ambas situaciones entre dos mismos procesos dependiendo de los momentos de la ejecución. Estas situaciones de colaboración y competencia se regulan mediante mecanismos de comunicación y sincronización entre procesos .

Beneficios

Los beneficuos de la programación concurrente son:

Mejor aprovechamiento de la CPU

Aumento de la velocidad de la aplicación completa

Mejora en el funcionamiento de algunos tipos de aplicaciones como: sistemas de control, aplicacuones web (servidores) , GUI, sumulación física, ssggbbdd donde cada acceso de una cuenta genera un proceso.

Programación concurrente en java

Igual que hemos visto que desde un programa java se puede hacer que el sistema operativo genere nuevos procesos desde el proceso que se inicia mediante el programa J ava, dentro de un mismo proceso se pueden producir "subprocesos" denominados hilos. Un hilo es una parte de un programa que ejecuta sus instrucciones de forma independiente de otros hilos .

Los hilos de un proceso comparten el contexto del proceso

(variables de entorno) pero cada hilo tiene una copia local y propia de este contexto.

Diferencias entre procesos e hilos :

— los hilos comparten el espacio de memoria del proceso que los creó, tanto datos como direcciones de memoria, mientras los procesos tienen espacios de memorua undependuentes (cedidos por el sistema operativo) y la interacción entre procesos la proporciona el sistema operativo,

— hilos y procesos tienen distintos estados en los que pueden estar, pero los cambios de estado de los procesos son más lentos y costosos. A los hilos por la anterior característica, derivada de pertenecer al mismo proceso, se les llama procesos ligeros,  el unicio y el fin de un hilo son más rápidos que los de un proceso,  la gestión de los procesos la realiza el núcleo del sistema operativo, mientras que la gestión del hilo la realiza el proceso donde se han generado .

Implementación de programación concurrente en java

La programación concurrente en java se realiza dividiendo el programa en hilos. En java se ut Iluza la clase Th read (hilo) para crear programas con hl los concurrentes.

En java hay dos formas de utilizar o crear h 11 os:

— crear una clase que desciende de la clase Thread, y sobrecargar el método r un ( ) (extends Thread)  implementar la interfaz Runnable (implements Runnable) , y declarar el método run ( ) . Este método se utiliza cuando una clase ya desciende de otra, y por tanto no se puede hacer descender también de Thread, en estos casos se crean los hilos implementando la interfaz Runnable .

Ejemplo de HiloSimp1e heredando de la clase Thread. En el método run() se incluyen las sentencias que se ejecutarán simultáneamente con las otras partes del programa. Al terminar el método run ( ) , termina el hilo que se genera con él.

public cl ass HiloSenci110 extends Thread{ public void run ( ) { for (int i

System. out . println ("Paso "+i+" en el hilo. •

Luego se utiliza el hilo desde el main de otra clase: public class HiloSenci110Main { public sta tic void main (String[] args) 

HiloSenci110Thread hst=new HiloSenc1110Thread ( ) ; hs t. start ( ) ; for (int i

System.out.println ("Paso "+i+" en el padre") ;

Para hacer 10 mismo con la inter faz Runnable:

public class HiloSenciIIoRunnab1e implements Runnable{ public void run ( ) for (1 nt i =

System. out . print In ( "Pasoen el hi 10. ")

Luego se utiliza la c lase del hi 10 en la c lase donde estå el método main

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

HiloSenciIIoRunnab1e hsr new HiloSenci110Runnab1e ( ) ; Thread hi 10 new Thread (hsr) ; hi 10. start ( ) ; for (int i —

System. out . print In ( "Paso en el padre")

Se pueden agrupar las c lases anterxores en una misma clase.

public class ClaseHi10 extends Thread { // constructor objetos ClaseHiIo public ClaseHi10 (String nombre hi 10) {

// 11 ama al constructor de la c lase de la que desciende super (nombre hi 10) sout ("Se crea el hi 10 : "+getName ( ) )

/ /método run ( ) public voxd run ( ) { for (int 1=0, i < 10, i++) sout ( "Hilo.  pa so. 

/ /método main ( )

Public static void main ( ) {

//creaciön de dos objetos de claseHi10

ClaseHiIo 0b j CHInew ClaseHi10 ("Hilo 1") ClaseHiIo objCH2new ClaseHiIo ("Hilo 2") objCH1 . start ( ) ; objCH2 . start ( ) ; sout ("dos hi Ios in 1 ciados" 

En el ej emplo anterior, se crea una c lase que extiende la c lase

Thread, se implementa el método run ( ) de Ios hilos, y se

Implementa método ma 1 n que genera objetos h 110 de la clase.

Algunos métodos aplicables a los hilos .

— start ( ) hace que el hilo comience la ejecución del código contenido en el método run ( ) ,  boolean isA1ive ( ) devuelve true or false dependiendo de si el hilo está "vivo"  sleep (long miliseconds) detiene la ejecución del hilo durante los mulisegundos  run ( ) el código que contiene son las sentencias que procesará el hilo. El método start ( ) hace una llamada a este método después de inicializar el hilo. Al finalizar run() el hilo termina. Es el únxco método de la interfaz Runnable.

— String toString()

Devuelve una cadena con información del hilo, donde se incluye: el nombre, prioridad, grupo al que pertenece  long getld() identificador del hilo — void yield() el hilo se detiene y permite la ejecución de otros hilos,

— String get Name ( ) devuelve el nombre del hilo, — setName (String nombre) cambia el nombre del hilo al indicado en el parámetro, — int get Priority() — set Priori ty (int prio) respecto a la prioridad está comprendida entre Thread.MIN PRIORITY y Thread.MAX PRIORI TY — voud interrupt ( ) interrumpe la ejecución del hilo  boolean interrupted ( ) devuelve si el hilo está interrumpido

 Thread currentThread ( ) devuelve la referencia al objeto Th read que se está ejecutando,

— boolean isDaemon ( ) devuelve si el hilo es daemon, son estos hilos con baja prioridad, en segundo plano, un ejemplo es el recolector de basura.

 setDaemon (boolean es Demonio) si el parámetro es true, el hilo es Daemon, y si es false, el hilo es de usuarxo, — void stop ( ) detiene el hilo, es un método en desuso,

— int activeCount ( ) devuelve el número de hilos activos en el grupo de hilos del hilo sobre el que se lanza, — Thread.State get Sta te ( ) devuelve el estado del hilo: NEW, RUNNABLE , BLOCKED, WAITING, TIMED WAITING, TERMINATED.

Realiza una clase MetodosHi10s que extienda la clase Thread y cumpla los siguientes requisitos :

 El método run() que ut i II zan los hulos generados cuando se lanza el método start ( ) mostrará la siguiente información del hilo en ejecución: el identificador, el nombre, la cantidad de hilos, la prioridad del hilo, y, el resultado del método toString ( )

 El método main ( ) mostrará la misma un formación que el método run ( ) , pero en este caso del hilo ma un (sí, main, también genera un hilo java) . También en el método main ( ) , dentro de un bucle de 5 pasos, se declaran objetos de la misma clase, y luego se lanza el método start() de cada objeto (pues los objetos son hilos) . Los objetos hilo generados en el bucle tendrán nombres formado por "hilo" y el índice del bucle, prioridades al azar entre la mí numa y la máxuma,

Grupos de hilos

Todo hilo en java debe pertenecer a un grupo. Si no se indica expresamente entonces los hilos pertenecen al grupo al que pertenece el procedimiento main a partir del que se generan los hilos.

Se utiliza la clase ThreadGroup para manejar grupos de hilos en las aplicaciones java.

En la clase Th read hay constructores que permiten crear los hilos dentro del grupo que se quiera.

Thread (grupo ThreadGroup, destino Runnable, nombre S t ring)

Thread (ThreadGroup grupo, Runnable destino, String nombre)

Realiza un programa java GruposHi10s, que al ejecutarse genera diez hilos. Los hilos pertenecerán a uno cualquiera de dos grupos llamados grupo 1 y grup02. Para cada hilo se elegirá al azar el grupo al que pertenece. El código que ejecuta cada hilo mostrará la información del hilo, entre ella el grupo al que pertenece. El hilo al que pertenece el main también muestra la misma Información que cada uno de los hilos.

Realiza un programa java llamado TicTacToc que al ej ecu tarse : crea e inicia tres hilos distintos, los métodos run de los hilos tienen bucles infinitos, dentro de cada bucle, un hilo pone en pantalla Tic, otro Tac, y el otro Toc, y así hasta el infinito. . .

se puede utilizar la función sleep para poder ver la palabra por pantalla.

Realiza un programa java llamado OchentaHi10s que al ejecutarse crea e inicia ochenta hilos. Cada hilo iniciará un bucle de un millón de pasos. Cada hilo en cada paso del bucle, escribe una letra al azar siempre en la misma columna de la pantalla. La columna será distinta para cada hilo.

11