

EJERCICIOS

1. EJERCICIO CONCURRENCIA BÁSICA

Pedir al usuario hasta qué número queremos contar, de cuánto en cuánto y cuántos procesos vamos a utilizar para ello, y por último mostrar cuántas veces ha contado cada uno de los procesos. Implementarlo.

2. EJERCICIO . RUNNABLE Y PRIORIDADES

Realizar un programa que cuente hasta 100000 entre tres procesos y que al finalizar muestre un resumen de las veces que ha contado cada uno. Debemos tener en cuenta que cada proceso tiene prioridades distintas, y tendrá que informarse en el resumen final. Implementar la clase Runnable para realizarlo. Hacerlo con los procesos sincronizados y sin sincronizarlos. ¿Cuenta más veces el proceso que tiene más prioridad?

3. EJERCICIO . GRUPOS Y PRIORIDADES

Pedir al usuario número de procesos a crear, contamos sincronizadamente entre todos ellos, teniendo en cuenta que el mínimo número de procesos será 2, y el primero de ellos estará en un único grupo y con prioridad mínima, y resto de procesos creados tendrá prioridad máxima en otro grupo.

Analizar si el proceso de prioridad mínima cuenta menos veces que alguno de los procesos de prioridad máxima.

4. EJERCICIO . MATANDO PROCESOS A TRAVÉS DEL GRUPO

Centro de Enseñanza Concertada
Gregorio Fernández

Crear desde una clase Aplicación 100 procesos que contarán individualizada e indefinidamente. Desde el proceso padre, después de 1000 milisegundos, matará a todos los hijos sin crear un array de procesos en ese padre. Mostrar al morir el hijo el número de proceso que es y a qué cantidad ha llegado contando (este mensaje lo muestra el hijo).

5. EJERCICIO MATANDO PROCESOS. RUNNABLE

Crear un proceso padre que creará 5 hijos y contarán entre todos hasta 5000, el primero que llegue al contador máximo (5000), matará al resto de "hermanos". Implementa los procesos con el interface Runnable.

6. EJERCICIO. WAIT Y NOTIFY.

5 procesos van a "luchar" por recuperar una cadena de texto, que se inserta en una lista de Strings. La última que se añada, desde consola, será la recuperada justo después de pulsar el intro. Al finalizar la aplicación, el proceso principal, mostrará todas las cadenas introducidas, y evidentemente no quedarán procesos en ejecución. Se dejará de pedir datos al usuario, cuando éste inserte la cadena "fin". Debemos tener en cuenta, que un mismo proceso podría recuperar varias cadenas de texto.

7. EJERCICIO . WAIT Y NOTIFY 2.

Cada 3 segundos (aproximado, simplemente con un `Thread.sleep(3000)`), se van a generar 5 números aleatorios , una vez generados despertaremos cinco procesos que cogerán cada uno de esos números aleatorios mostrando por pantalla qué número han cogido (lógicamente no podrán coger el mismo). Optimiza la aplicación para que no haya procesos que estén ociosos con tiempo de CPU. el proceso padre cuando finalice de generar los cinco números aleatorios mostrará un string por consola informando de que se han generado. Finalizar la aplicación después de haber generado tres veces cinco números aleatorios.