Entregable Lab. 1

1. .

1.1

class MiHebra extends Thread{

int miId;

public MiHebra(int miId){

this.miId = miId;

}

public void run(){

for(int i = 0; i<1000; i++)

System.out.println(miId);

}

}

class EjemploCreacion4a{

public static void main( String args[]){

new MiHebra(0).start();

new MiHebra(1).start();

}

}

1.2

class MiRun implements Runnable{

int miId;

public MiRun(int miId){

this.miId = miId;

}

public void run(){

for(int i = 0; i<1000; i++)

System.out.println(miId);

}

}

class EjemploCreacion4b{

public static void main( String args[]){

new Thread(new miRun(0)).start();

new Thread(new miRun(1)).start();

}

}

* Si, en el primer caso se intercalan sin orden y en el segundo también pero con tras varias iteraciones
* El hilo principal es quien se encarga de ejecutar el código de la hebra
* Deja de haber concurrencia por lo mencionado arriba

1. .

class MiHebra extends Thread{

int miId;

int ini;

int fin;

public MiHebra(int miId, int ini, int fin){

this.miId = miId;

this.ini = ini;

this.fin = fin;

}

public void run(){

int suma = 0;

for(int i = ini+1; i<fin; i++)

suma+=i;

System.out.println("Soy la hebra "+miId+" y la suma de los números comprendidos entre "+ini+" y "+fin+" es "+suma);

}

}

class EjemploCreacion5{

public static void main( String args[]){

System.out.println("Vamos a arrancar las 2 hebras")

new Thread(new MiHebra(0, 7,15)).start();

new Thread(new MiHebra(1,4,10)).start();

System.out.println("Trabajo terminado");

}

}

* Si, ha finalizado antes el programa principal.
* No se altera, la hebra principal acaba y se acabará la ejecución

class EjemploCreacion5{

public static void main( String args[]) throws InterruptedException {

System.out.println("Vamos a arrancar las 2 hebras");

MiHebra t = new MiHebra(0, 1,1000000);

MiHebra t2 = new MiHebra(1,1,1000000);

new Thread(t).setDaemon(true);

new Thread(t2).setDaemon(true);

t.start();

t2.start();

t.join();

t2.join();

System.out.println("Trabajo terminado");

}

}

* La hebra principal espera a las otras dos hebras.

No, que sean Daemon o no, no influye.

1. .

class MiHebra3 extends Thread {

int miId;

CuentaIncrementos1a obj;

public MiHebra3(int miId, CuentaIncrementos1a obj) {

this.miId = miId;

this.obj = obj;

}  
  
 public void run() {  
 for (int i = 1; i < 1000000; i++) {  
 System.out.println("Soy la hebra " + miId + " y voy a realizar un incremento");  
 obj.incrementaContador();  
 System.out.println("Incremento realizado, valor actual: "+obj.dameContador());  
 }  
  
 }  
}

3.2.2 - CuentaIncrementos1a contador = new CuentaIncrementos1a();

3.2.3 - System.out.println("El valor inicial del contador es: "+contador.dameContador());

3.2.4

for (int i = 0; i < numHebras; i++) {

vectorHebras[i] = new MiHebra3(i, contador);

vectorHebras[i].start();

}

3.2.5 – Añadir debajo después del for vectorHebras[i].join();

3.2.6 - System.out.println("El valor final del contador es: " + contador.dameContador());

3.3 Hay concurrencia porque las hebras se alternan y al final el resultado es el esperado.

3.4 Debería escribir 4.000.000. El ordenador escribe 3.999.995

3.5 Supongo que el error viene por la compartición de datos.

4.

4.1 No, la interfaz se bloquea.

4.2

class MiHebra4 extends Thread {

long num;

public MiHebra4(long num) {

this.num = num;

}

public void run() {

boolean primo = esPrimo(num);

if (primo) {

System.out.println("El numero " + num + " SI es primo.");

} else {

System.out.println("El numero " + num + " NO es primo.");

}

}

}

try {  
 long numero = Long.parseLong(txfNumero.getText().trim());  
 System.out.println("Examinando numero: " + numero);  
 hebraEsPrimo = new MiHebra4(numero);  
 hebraEsPrimo.start();  
}

4.3 Si, ahora funciona la interfaz mientras averigua si es primo

4.4 No hace falta, sólo sería necesario si mientras calcula quisieras cerrar la ventana, ya que si no es del tipo Daemon no finalizará la ejecución