

programacion de servicios y procesos

Actividad 4 Tema 2



Carles Ricart

# Ejercicio 1: Lee atentamente las explicaciones y teniendo en cuenta lo visto en clase, indica la salida de este programa y comenta el código:

**package** ejercicio4;

**class** Callme {

**synchronized** **void** call (String msg) {

System.*out*.print("[" + msg);

**try** {

Thread.*sleep*(1000);

}

**catch** (Exception e) {};

System.*out*.println("]");

}

}

**class** Caller **implements** Runnable { //Creamos una clase que implementa la //interfaz runnable

String msg;

Callme target;

**public** Caller(Callme t, String s) {

target = t;

msg = s;

**new** Thread(**this**).start();

}

**public** **void** run() {

**synchronized**(target) {

target.call(msg);

}

}

}

**class** Synch {

**public** **static** **void** main(String args[]) {

Callme target = **new** Callme();

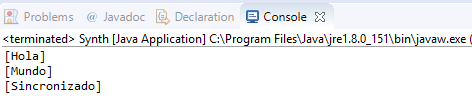
**new** Caller(target, "Hola");

**new** Caller(target, "Mundo");

**new** Caller(target, "Sincronizado");

}

}



# Ejercicio 2: A continuación, tenemos dos códigos, que a priori, realizan la misma operación Indica qué hacen dichos programas, cuál será su salida por pantalla, cual es la diferencia entre las soluciones propuestas y, por último, cuál sería la solución correcta.

class Q {

int n;

synchronized int get() {

System.out.println("Obtengo: " + n);

return n;

}

synchronized void put(int n) {

this.n = n;

System.out.println("Pongo: " + n);

}

}

class Producer implements Runnable {

Q q;

Producer(Q q) {

this.q = q;

new Thread(this, "Productor").start();

}

public void run() {

int i = 0;

while(true) {

q.put(i++);

}

}

}

class Consumer implements Runnable {

Q q;

Consumer(Q q) {

this.q = q;

new Thread(this, "Consumidor").start();

}

public void run() {

while(true) {

q.get();

}

}

}

class PC { //La clase PC es la clase principal que lanza los hilos //Productor y Consumidor

public static void main(String args[]) {

Q q = new Q();

new Producer(q);

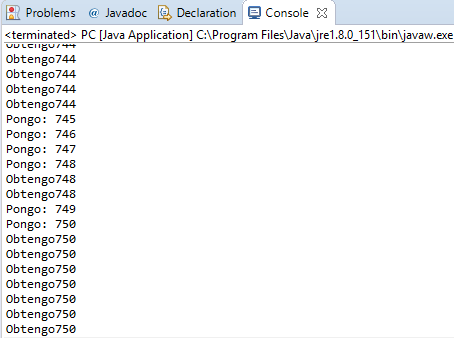
new Consumer(q);

System.out.println("Pulsa Control-C para parar.");

}

}

## Salida ejer 2



# COMPARA EL CÓDIGO ANTERIOR CON EL

# SIGUIENTE:

class Q {

int n;

boolean valueSet = false;

synchronized int get() {

if(!valueSet)

try {

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException capturada");

}

System.out.println("Obtengo: " + n);

valueSet = false;

notify();

return n;

}

synchronized void put(int n) {

if(valueSet)

try {

wait();

} catch(InterruptedException e) {

System.out.println("InterruptedException capturada");

}

this.n = n;

valueSet = true;

System.out.println("Pongo: " + n);

notify();

}

}

class Producer implements Runnable {

Q q;

Producer(Q q) {

this.q = q;

new Thread(this, "Productor").start();

}

public void run() {

int i = 0;

while(true) {

q.put(i++);

}

}

}

class Consumer implements Runnable {

Q q;

Consumer(Q q) {

this.q = q;

new Thread(this, "Consumidor").start();

}

public void run() {

while(true) {

q.get();

}

}

}

class PCFixed {

public static void main(String args[]) {

Q q = new Q();

new Producer(q);

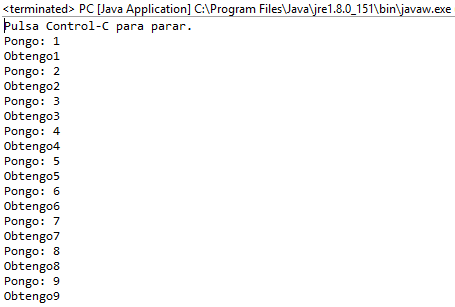
new Consumer(q);

System.out.println("Pulsa Control-C para parar.");

}

}

## Salida ejer 3



# Ejercicio 3: Crea un proceso en el cual se generen 5 hilos, a dichos hilos les asignaremos una prioridad diferente y lo ejecutaremos. Cuando hayamos comprobado que funciona, variaremos dichas prioridades para comprobar como el resultado varía. Recuerda forzar la máquina a sólo un núcleo.

**public** **class** Hilo **implements** Runnable {

**void** Hilo() {

}

**public** **void** run() {

**try** {

**for** (**int** i = 0; i < 15; i++) {

Thread.*sleep*(400);

System.***out***.println(Thread.*currentThread*().getName());

}

} **catch** (InterruptedException e) {

}

System.***out***.println("Termina el hilo: "+Thread.*currentThread*().getName());

}

}

**public** **class** Lanzame {

**public** **static** **void** main(String[] args) {

// **TODO** Auto-generated method stub

Hilo h1 = **new** Hilo();

Hilo h2 = **new** Hilo();

Hilo h3 = **new** Hilo();

Hilo h4 = **new** Hilo();

Hilo h5 = **new** Hilo();

Thread t1 = **new** Thread(h1);

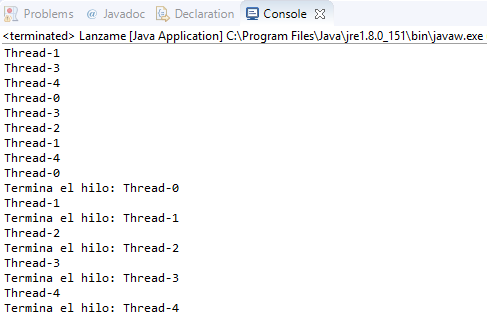
Thread t2 = **new** Thread(h2);

Thread t3 = **new** Thread(h3);

Thread t4 = **new** Thread(h4);

Thread t5 = **new** Thread(h5);

t1.setPriority(Thread.***MAX\_PRIORITY***);

 t2.setPriority(Thread.***NORM\_PRIORITY***);

t3.setPriority(Thread.***NORM\_PRIORITY***);

t4.setPriority(Thread.***NORM\_PRIORITY***);

t5.setPriority(Thread.***MIN\_PRIORITY***);

t1.start();

t2.start();

t3.start();

t4.start();

t5.start();

}

}

# Ejercicio 4: Crea un proceso en el cual se generen 2 hilos, uno que muestre el mensaje “Amaneciendo”, otro que nos muestre el mensaje “Buenas Noches, a descansar”, y tras 5 iteraciones, muestre el mensaje principal “otra semana de clases acabadas”, obviamente esta vez, te pido que sea un proceso cuyos hilos estén sincronizados.

public class Dia implements Runnable {

Saluda s;

public Dia(Saluda s)

this.s = s;

}

public void run()

for (int i=0;i<5;i++) {

s.buenosdias();

}

}

}

public class Noche implements Runnable {

Saluda s;

public Noche(Saluda s) {

this.s = s;

}

public void run() {

for (int i=0;i<5;i++) {

s.buenasnoches();

}

}

}

public class Lanzame {

public static void main(String[] args) {

// TODO Auto-generated method stub

Saluda s = new Saluda(); //Creamos el objeto Saluda

Dia d = new Dia(s); //Creamos el objeto dia al que le pasamos por parametro s

Noche n = new Noche(s); //Creamos el objeto noche al que le pasamos por parametro s

Thread t1 = new Thread(d); //Instanciamos los hilos

Thread t2 = new Thread(n);

t1.start(); //Ejecutamos los hilos

t2.start();

try {

t2.join();

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

//Hasta que no haya terminado el ultimo hilo no sale por pantalla el mensaje

System.out.print("Otra semana de clases acabadas");

}

}

public class Saluda {

boolean amanecio = false;

int dia,noche=0;

public synchronized int buenosdias() { //Creamos el metodo buenosdias que esta sincronizado

if (!amanecio) { //Si no ha amanecido ejecutara el codigo imprimiendo que ha amanecido

try {

Thread.sleep(400);

System.out.println(dia+": Amaneciendo");

amanecio=true;

dia++;

notify(); //Cuando ya ha amanecido notificamos al segundo hilo que puede acceder al objeto

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

} else {

try {

wait(); //Si no ha amanecido el hilo quedará suspendido

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return dia;

}

public synchronized int buenasnoches(){ //Creamos el metodo buenasnoches que esta sincronizado

if (amanecio) { //Si ha amanecido ejecutara el codigo imprimiendo Buenas noches

try {

Thread.sleep(400);

System.out.println(noche+": Buenas noches, a descansar");

noche++;

amanecio = false;

notify(); //Cuando ejecuta el codigo notificamos al primer hilo que puede acceder al objeto

} catch (InterruptedException e) {

e.printStackTrace();

}

} else {

try {

wait(); //Si no ha anochecido esperara

} catch (InterruptedException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

}

return noche;

}

}