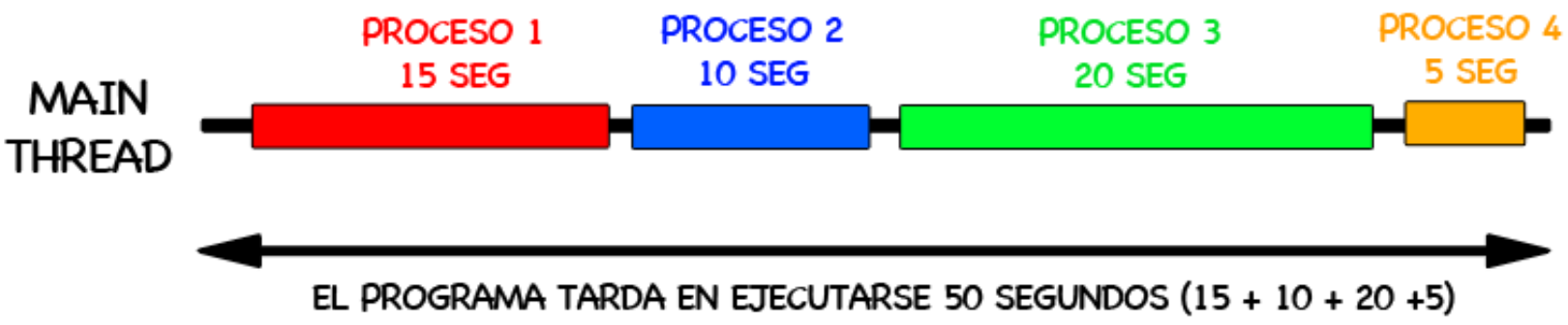
Multitarea e Hilos en Java con ejemplos (Thread & Runnable)

El proyecto de este post lo puedes descargar pulsando AQUI (https://github.com/jarroba/ThreadsJarroba).

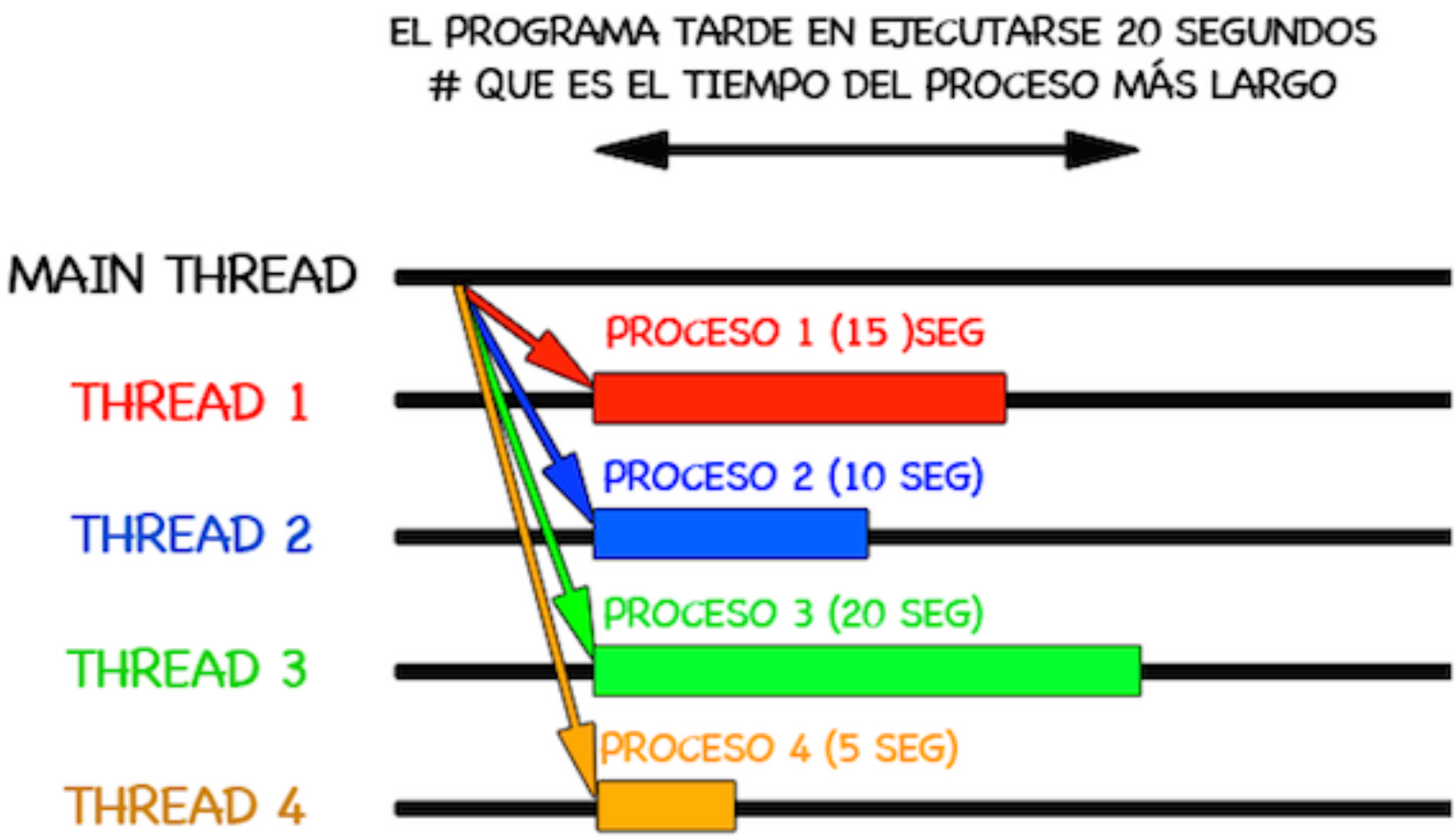
En esta entrada vamos a ver las diferentes maneras de como trabajar con Threads en Java (o hilos en español). Sino tienes muy claro el concepto de la multitarea te recomendamos que te leas primero la entrada de Multitaréa e Hilos, fácil y muchas ventajas (http://jarroba.com/multitarea-e-hilos-facil-y-muchas-ventajas/) aunque en esta entrada también veremos (en menor detalle) los conceptos y las ventajas de la multitarea.

En esencia la multitarea nos permite ejecutar varios procesos a la vez; es decir, de forma concurrente y por tanto eso nos permite hacer programas que se ejecuten en menor tiempo y sean más eficientes. Evidentemente no podemos ejecutar infinitos procesos de forma concurrente ya que el hardware tiene sus limitaciones, pero raro es a día de hoy los ordenadores que no tengan más de un núcleo por tanto en un procesador con dos núcleos se podrían ejecutar dos procesos a la vez y así nuestro programa utilizaría al máximo los recursos hardware. Para que veáis la diferencia en un par de imágenes, supongamos que tenemos un programa secuencial en el que se han de ejecutar 4 procesos; uno detrás de otro, y estos tardan unos segundos:



(http://jarroba.com/wp-content/uploads/2014/05/Threads_Un_unico_hilo.png)

Si en vez de hacerlo de forma secuencial, lo hiciésemos con 4 hilos, el programa tardaría en ejecutarse solo 20 segundos, es decir el tiempo que tardaría en ejecutarse el proceso más largo. Esto evidentemente sería lo ideal, pero la realidad es que no todo se puede paralelizar y hay que saber el número de procesos en paralelo que podemos lanzar de forma eficiente. En principio en esta entrada no vamos a hablar sobre ello ya que el objetivo de la misma es ver como se utilizan los hilos en java con un ejemplo relativamente sencillo y didáctico.



(http://jarroba.com/wp-content/uploads/2014/05/Threads_Varios_hilos.png)

En Java para utilizar la multitarea debemos de usar la clase ***Thread*** (es decir que la clase que implementemos debe heredar (http://jarroba.com/herencia-en-la-programacion-orientada-a-objetos-ejemplo-en-java/) de la clase Thread) y la clase Thread implementa la Interface (http://jarroba.com/polimorfismo-en-java-interface-parte-ii-con-ejemplos/) ***Runnable***. En el siguiente diagrama de clase mostramos la Interface Runnable y la clase Thread con sus principales métodos:

(http://jarroba.com/wp-content/uploads/2014/05/MultiHilos_Jarroba_Diagrama_de_clases.jpg)

En esta entrada no vamos a ver como utilizar todos los métodos de la clase Thread, pero os los mostramos para que sepáis que existen y a parte por su nombre podéis intuir su funcionalidad.

En esta entrada vamos a poner un ejemplo para que veáis las ventajas de la multitarea, viendo como se ejecutaría un programa sin utilizar la multitarea y otro utilizándola.

En este ejemplo vamos a simular el proceso de cobro de un supermercado; es decir, unos clientes van con un carro lleno de productos y una cajera les cobra los productos, pasándolos uno a uno por el escaner de la caja registradora. En este caso la cajera debe de procesar la compra cliente a cliente, es decir que primero le cobra al cliente 1, luego al cliente 2 y así sucesivamente. Para ello vamos a definir una clase “Cajera” y una clase “Cliente” el cual tendrá un “array de enteros” que representaran los productos que ha comprado y el tiempo que la cajera tardará en pasar el producto por el escaner; es decir, que si tenemos un array con [1,3,5] significará que el cliente ha comprado 3 productos y que la cajera tardara en procesar el producto 1 ‘1 segundo’, el producto 2 ‘3 segundos’ y el producto 3 en ‘5 segundos’, con lo cual tardara en cobrar al cliente toda su compra ‘9 segundos’.

Explicado este ejemplo vamos a ver como hemos definido estas clases:

Clase “***Cajera.java***“:

```
public class Cajera {

	private String nombre;

	// Constructor, getter y setter

	public void procesarCompra(Cliente cliente, long timeStamp) {

		System.out.println("La cajera " + this.nombre +
					" COMIENZA A PROCESAR LA COMPRA DEL CLIENTE "
+ cliente.getNombre() +
					" EN EL TIEMPO: " + (System.currentTimeMillis
() - timeStamp) / 1000  +
					"seg");

		for (int i = 0; i < cliente.getCarroCompra().length; i++) {
					this.esperarXsegundos(cliente.getCarroCompra(
)[i]);
					System.out.println("Procesado el producto " +
(i + 1) +
					" ->Tiempo: " + (System.currentTimeMillis() -
timeStamp) / 1000 +
					"seg");
		}

		System.out.println("La cajera " + this.nombre + " HA TERMINAD
O DE PROCESAR " +
					cliente.getNombre() + " EN EL TIEMPO: " +
					(System.currentTimeMillis() - timeStamp) / 10
00 + "seg");

	}


	private void esperarXsegundos(int segundos) {
		try {
			Thread.sleep(segundos * 1000);
		} catch (InterruptedException ex) {
			Thread.currentThread().interrupt();
		}
	}

}
```

Clase “***Cliente.java***“:

```
public class Cliente {

	private String nombre;
	private int[] carroCompra;

	// Constructor, getter y setter

}
```

Si ejecutásemos este programa propuesto con dos Clientes y con un solo proceso (que es lo que se suele hacer normalmente), se procesaría primero la compra del Cliente 1 y después la del Cliente 2, con lo cual se tardará el tiempo del Cliente 1 + Cliente 2. A continuación vamos a ver como programamos el método Main para lanzar el programa. ***CUIDADO***: *Aunque hayamos puesto dos objetos de la clase Cajera (cajera1 y cajera2) no significa que tengamos dos cajeras independientes, lo que estamos diciendo es que dentro del mismo hilo se ejecute primero los métodos de la cajera1 y después los métodos de la cajera2, por tanto a nivel de procesamiento es como si tuviésemos una sola cajera*:

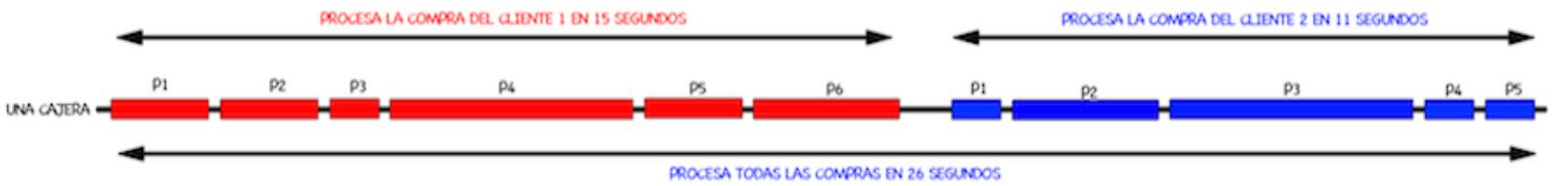
Clase "***Main.java***":

```
public class Main {

	public static void main(String[] args) {

		Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2,
1, 5, 2, 3 });
		Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3,
5, 1, 1 });

		Cajera cajera1 = new Cajera("Cajera 1");
		Cajera cajera2 = new Cajera("Cajera 2");

		// Tiempo inicial de referencia
		long initialTime = System.currentTimeMillis();

		cajera1.procesarCompra(cliente1, initialTime);
		cajera2.procesarCompra(cliente2, initialTime);
	}
}
```

Si ejecutamos este código tendremos lo siguiente:

```
La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIE
MPO: 0seg
Procesado el producto 1 ->Tiempo: 2seg
Procesado el producto 2 ->Tiempo: 4seg
Procesado el producto 3 ->Tiempo: 5seg
Procesado el producto 4 ->Tiempo: 10seg
Procesado el producto 5 ->Tiempo: 12seg
Procesado el producto 6 ->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg
La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIE
MPO: 15seg
Procesado el producto 1 ->Tiempo: 16seg
Procesado el producto 2 ->Tiempo: 19seg
Procesado el producto 3 ->Tiempo: 24seg
Procesado el producto 4 ->Tiempo: 25seg
Procesado el producto 5 ->Tiempo: 26seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 26seg
```

Como vemos se procesa primero la compra del cliente 1 y después la compra del cliente 2 tardando en procesar ambas compras un tiempo de 26 segundos.



(http://jarroba.com/wp-content/uploads/2014/05/Compra_1_cajera_1_hilo.png)

¿Y si en vez de procesar primero un cliente y después otro, procesásemos los dos a la vez?, ¿Cuanto tardaría el programa en ejecutarse?. Pues bien si en vez de haber solo una Cajera (es decir un solo hilo), hubiese dos Cajeras (es decir dos hilos o threads) podríamos procesar los dos clientes a la vez y tardar menos tiempo en ejecutarse el programa. Para ello debemos de modificar la clase "***Cajera.java***" y hacer que esta clase herede de la clase ***Thread*** para heredar y sobre-escribir algunos de sus métodos. Primero vamos a ver como codificamos esta nueva clase "***CajeraThread.java***" y después explicamos sus caracteristicas.

```
public class CajeraThread extends Thread {

	private String nombre;

	private Cliente cliente;

	private long initialTime;

	// Constructor, getter & setter

	@Override
	public void run() {

		System.out.println("La cajera " + this.nombre + " COMIENZA A
PROCESAR LA COMPRA DEL CLIENTE "
					+ this.cliente.getNombre() + " EN EL
TIEMPO: "
					+ (System.currentTimeMillis() - this.
initialTime) / 1000
					+ "seg");

		for (int i = 0; i < this.cliente.getCarroCompra().length; i++
) {
			this.esperarXsegundos(cliente.getCarroCompra()[i]);
			System.out.println("Procesado el producto " + (i + 1)
			+ " del cliente " + this.cliente.getNombre() + "->Tie
mpo: "
			+ (System.currentTimeMillis() - this.initialTime) / 1
000
			+ "seg");
		}

		System.out.println("La cajera " + this.nombre + " HA TERMINAD
O DE PROCESAR "
					+ this.cliente.getNombre() +
" EN EL TIEMPO: "
					+ (System.currentTimeMillis()
- this.initialTime) / 1000
					+ "seg");
	}

	private void esperarXsegundos(int segundos) {
		try {
			Thread.sleep(segundos * 1000);
		} catch (InterruptedException ex) {
			Thread.currentThread().interrupt();
		}
	}

}
```

Lo primero que vemos y que ya hemos comentado es que la clase “CajeraThread” debe de heredar de la clase Thread: “***extendsThread***“.

Otra cosa importante que vemos es que hemos sobre-escrito el método "***run()***" (de ahi la etiqueta *@Override*) . Este método es imprescindibles sobre-escribirlo (ya que es un método que esta en la clase Runnable y la clase Thread Implementa esa Interface) porque en él se va a codificar la funcionalidad que se ha de ejecutar en un hilo; es decir, que lo que se programe en el método "***run()***" se va a ejecutar de forma secuencial en un hilo. En esta clase "CajeraThread" se pueden sobre-escribir más métodos para que hagan acciones sobre el hilo o thread como por ejemplo, parar el thread, ponerlo en reposos, etc. A continuación vamos a ver como programamos el método Main para que procese a los clientes de forma paralela y ver como se tarda menos en procesar todo. El método Main esta en la clase "***MainThread.java***" que tiene el siguiente contenido:

```
public class MainThread {

	public static void main(String[] args) {

		Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2,
1, 5, 2, 3 });
		Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3,
5, 1, 1 });

		// Tiempo inicial de referencia
		long initialTime = System.currentTimeMillis();
		CajeraThread cajera1 = new CajeraThread("Cajera 1", cliente1,
initialTime);
		CajeraThread cajera2 = new CajeraThread("Cajera 2", cliente2,
initialTime);

		cajera1.start();
		cajera2.start();
	}
}
```

Ahora vamos a ver cual sería el resultado de esta ejecución y vamos a comprobar como efectivamente el programa se ejecuta de forma paralela y tarda solo 15 segundos en terminar su ejecución:

```
La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIE
MPO: 0seg
La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIE
MPO: 0seg
Procesado el producto 1 del cliente Cliente 2->Tiempo: 1seg
Procesado el producto 1 del cliente Cliente 1->Tiempo: 2seg
Procesado el producto 2 del cliente Cliente 2->Tiempo: 4seg
Procesado el producto 2 del cliente Cliente 1->Tiempo: 4seg
Procesado el producto 3 del cliente Cliente 1->Tiempo: 5seg
Procesado el producto 3 del cliente Cliente 2->Tiempo: 9seg
Procesado el producto 4 del cliente Cliente 2->Tiempo: 10seg
Procesado el producto 4 del cliente Cliente 1->Tiempo: 10seg
Procesado el producto 5 del cliente Cliente 2->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg
Procesado el producto 5 del cliente Cliente 1->Tiempo: 12seg
Procesado el producto 6 del cliente Cliente 1->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg
```

En este ejemplo vemos como el efecto es como si dos cajeras procesasen la compra de los clientes de forma paralela sin que el resultado de la aplicación sufra ninguna variación en su resultado final, que es el de procesar todas las compras de los clientes de forma independiente. De forma gráfica vemos que el programa ha realizado lo siguiente en dos hilos distintos:

(http://jarroba.com/wp-content/uploads/2014/05/Compra_2_hilos.png)

Otra forma de hacer lo mismo pero sin heredar de la clase “Thread” es implementar la Interface “Runnable”. En este caso no dispondremos ni podremos sobre-escribir los métodos de la clase Thread ya que no la vamos a utilizar y solo vamos a tener que sobre-escribir el método “***run()***“. En este caso solo será necesario implementar el método “***run()***” para que los procesos implementados en ese método se ejecuten en un hilo diferente. Vamos a ver un ejemplo de como utilizando objetos de las clases “***Cliente.java***” y “***Cajera.java***” podemos implementar la multitarea en la misma clase donde se llama al método Main de la aplicación. A continuación vemos la codificación en la clase “***MainRunnable.java***“:

```
public class MainRunnable implements Runnable{

	private Cliente cliente;
	private Cajera cajera;
	private long initialTime;

	public MainRunnable (Cliente cliente, Cajera cajera, long initialTime
){
		this.cajera = cajera;
		this.cliente = cliente;
		this.initialTime = initialTime;
	}

	public static void main(String[] args) {

		Cliente cliente1 = new Cliente("Cliente 1", new int[] { 2, 2,
1, 5, 2, 3 });
		Cliente cliente2 = new Cliente("Cliente 2", new int[] { 1, 3,
5, 1, 1 });

		Cajera cajera1 = new Cajera("Cajera 1");
		Cajera cajera2 = new Cajera("Cajera 2");

		// Tiempo inicial de referencia
		long initialTime = System.currentTimeMillis();

		Runnable proceso1 = new MainRunnable(cliente1, cajera1,
initialTime);
		Runnable proceso2 = new MainRunnable(cliente2, cajera2,
initialTime);

		new Thread(proceso1).start();
		new Thread(proceso2).start();

	}

	@Override
	public void run() {
		this.cajera.procesarCompra(this.cliente, this.initialTime);
	}

}
```

En este caso implementamos el método "***run()***" dentro de la misma clase donde se encuentra el método Main, y en el llamamos al método de "procesarCompra()" de la clase Cajera. Dentro del método Main, nos creamos dos objetos de la misma clase en la que estamos ("new MainRunnable") y nos creamos dos objetos de la clase Thread para lanzar los proceso y que se ejecuten estos en paralelo. El resultado de esta ejecución es el mismo que en el caso anterior:

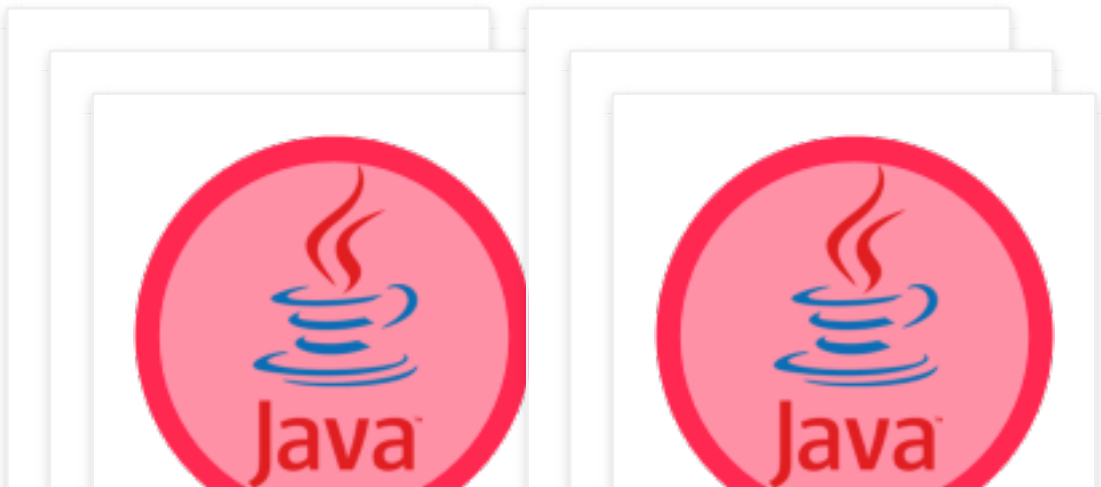
```
La cajera Cajera 2 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 2 EN EL TIE
MPO: 0seg
La cajera Cajera 1 COMIENZA A PROCESAR LA COMPRA DEL CLIENTE Cliente 1 EN EL TIE
MPO: 0seg
Procesado el producto 1 del cliente Cliente 2->Tiempo: 1seg
Procesado el producto 1 del cliente Cliente 1->Tiempo: 2seg
Procesado el producto 2 del cliente Cliente 2->Tiempo: 4seg
Procesado el producto 2 del cliente Cliente 1->Tiempo: 4seg
Procesado el producto 3 del cliente Cliente 1->Tiempo: 5seg
Procesado el producto 3 del cliente Cliente 2->Tiempo: 9seg
Procesado el producto 4 del cliente Cliente 2->Tiempo: 10seg
Procesado el producto 4 del cliente Cliente 1->Tiempo: 10seg
Procesado el producto 5 del cliente Cliente 2->Tiempo: 11seg
La cajera Cajera 2 HA TERMINADO DE PROCESAR Cliente 2 EN EL TIEMPO: 11seg
Procesado el producto 5 del cliente Cliente 1->Tiempo: 12seg
Procesado el producto 6 del cliente Cliente 1->Tiempo: 15seg
La cajera Cajera 1 HA TERMINADO DE PROCESAR Cliente 1 EN EL TIEMPO: 15seg
```

CONCLUSIONES Y ACLARACIONES:

El concepto de multitarea o multiprocesamiento es bastante sencillo de entender ya que solo consiste en hacer varias cosas a la vez sin que se vea alterado el resultado final. Como ya se ha dicho en la entrada no todo se puede paralelizar y en muchas ocasiones suele ser complicado encontrar la manera de paralelizar procesos dentro de una aplicación sin que esta afecte al resultado de la misma, por tanto aunque el concepto sea fácil de entender el aplicarlo a un caso práctico puede ser complicado para que el resultado de la aplicación no se vea afectado.

Por otro lado para los que empecéis a ver estos temas de la concurrencia, multitarea y demás, no so preocupéis al principio si os cuesta programar problemas de este tipo ya que a parte de la multitarea se mezclan cosas como la herencia y las Interfaces que al principio son cosas que cuestan de asimilar, así que ir poco a poco pero tener muy claro que la multitarea es muy util y se ha de aplicar para hacer las aplicaciones más eficientes y que den mejor rendimiento.

Recomendados



(http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-ii-runnable-executors/)
Multitarea e Hilos en Java con ejemplos II (Runnable & Executors)
(http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-ii-runnable-executors/)

(http://jarroba.com/excepciones-exception-en-java-con-ejemplos/)
Excepciones (Exception) en Java, con ejemplos
(http://jarroba.com/excepciones-exception-en-java-con-ejemplos/)

(http://jarroba.com/map-en-java-con-ejemplos/)
Map en Java, con ejemplos
(http://jarroba.com/map-en-java-con-ejemplos/)

(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)
Enum (Enumerados) en Java, con ejemplos
(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)

Enum (Enumerados) en Java, con ejemplos

(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/map-en-java-con-ejemplos/)

Map en Java, con ejemplos

(http://jarroba.com/map-en-java-con-ejemplos/)

(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)
Enum (Enumerados) en Java, con ejemplos
(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)

(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)
Enum (Enumerados) en Java, con ejemplos
(http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/enum-

enumerados-en-java-con-ejemplos/)

Enum (Enumerados) en Java, con ejemplos (http://jarroba.com/enum-enumerados-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/map-en-java-con-ejemplos/)

Map en Java, con ejemplos (http://jarroba.com/map-en-java-con-ejemplos/)

(http://jarroba.com/map-en-java-con-ejemplos/)

Map en Java, con ejemplos (http://jarroba.com/map-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/map-en-java-con-ejemplos/)

Map en Java, con ejemplos

(http://jarroba.com/map-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/map-en-java-con-ejemplos/)

Map en Java, con ejemplos

(http://jarroba.com/map-en-java-con-ejemplos/)

(http://jarroba.com/map-en-

java-con-ejemplos/)

Map en Java, con ejemplos (http://jarroba.com/map-en-java-con-ejemplos/)

Load more posts

(http://jarroba.com/map-en-java-con-ejemplos/)

Map en Java, con ejemplos (http://jarroba.com/map-en-java-con-ejemplos/)

Load more posts

Load more posts

Comparte esta entrada en:

31 comentarios sobre “Multitarea e Hilos en Java con ejemplos (Thread & Runnable)”

Víctor dice:

28/08/2015 a las 21:00 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-22841)

Que buen artículo, se agradece siempre el compartir los conocimientos 🙂

Tengo una duda que quisiera ver si me puedes explicar, soy novato en Java…

Si tuviera un archivo de tipo txt que quisiera leer X cantidad de veces de forma paralela (por ejemplo 5 hilos, leer de forma paralela 5 veces el archivo)… ¿Cómo tendría que ser la rutina para esto?…

Gracias de antemano!

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=22841#respond)

Ramón [Admin Jarroba] dice:

29/08/2015 a las 11:21 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-22871)

No he probado a abrir varias veces el mismo fichero con varios hilos, pero seguramente cuando lo abras con un hilo lo bloquearás y no te permitirá abrirlo con más. Simplemente obtén el contenido del fichero en una variable y multiplícalo en cada hilo que quieras realizar la lectura.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=22871#respond)

Ismael Venegas Castelló (https://github.com/Ismael-VC) dice:

24/06/2015 a las 07:33 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-19446)

Hola Javeros! Me gustaría saber, que opinan de este por a Julia: http://bit.ly/julia_async (http://bit.ly/julia_async) ?

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=19446#respond)

irene501 dice:

25/05/2015 a las 22:11 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-17287)

hola,

si quisiera que un proceso corra durante unos minutos, (suponiendo que se generan ls clientes aleatoriamente) cómo hago para dejarlo corriendo en un ciclo mientras con la condicion de que no hallan pasado los dos minutos?? se puede hacer eso???

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=17287#respond)

Ramón [Admin Jarroba] dice:

27/05/2015 a las 18:44 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-17362)

Hola Irene. Puedes hacer lo fácil (aunque a mí no me gusta mucho y no te asegura que justo a los 2 minutos se termine) que es “System.currentTimeMillis()” para capturar el tiempo actual en milisegundos, y sumarle dos minutos para obtener el momento de parada, solo te quedaría ir preguntando a cada iteración si se ha sobrepasado el tiempo. Otra es utilizar las clases Timer de Java, tienes ejemplos en https://docs.oracle.com/javase/tutorial/uiswing/misc/timer.html

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=17362#respond)

altairkurai dice:

25/03/2015 a las 17:23 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-13886)

muy buen articulo, tienen alguna información de git y github no me ha quedado del todo claro su manejo
gracias

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=13886#respond)

Adolfo dice:

18/03/2015 a las 03:43 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-13140)

Hola,

El proyecto de este post no esta disponible, podrian ponerlo.

Muy bueno este y demas post de POO.

Gracias

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=13140#respond)

Richard [Admin Jarroba] dice:

18/03/2015 a las 08:46 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-13169)

Hola Adolfo.

Ya tienes el proyecto compartido en Git, que por algún motivo debimos borrar el repositorio sin querrer. Gracias por avisarnos.

SL2

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=13169#respond)

Adolfo dice:

20/03/2015 a las 06:34 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-13389)

Gracias por compartir éstas entradas.

Excelente Web…..

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=13389#respond)

Jorge dice:

14/03/2015 a las 22:10 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-12826)

Hola,

Como puedo ejecutar el código desde la línea de comando en Linux. Muchas gracias.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=12826#respond)

Richard [Admin Jarroba] dice:

16/03/2015 a las 14:17 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-12966)

Hola Jorge.

Simplemente te debes de ir a la carpeta donde tengas es fichero java (NombreFichero.java) y poner lo siguiente: "java NombreFichero.java" y ya se ejecutará tu programa, eso siempre y cuando tengas instalada la maquina virtual java.

SL2

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=12966#respond)

Jorge dice:

14/03/2015 a las 22:07 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-12825)

Hola,
El link para descargar el proyecto al principio de esta pagina no funciona, podrían arreglarlo. Muchas gracias por todo el contenido de la pagina.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=12825#respond)

Sarahi dice:

22/02/2015 a las 06:10 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-10680)

!Hola! Esta muy completo lo que han explicado sobre los hilos, sin embargo tengo una duda se puede implementar creando una interfaz y como que herramientas puedo utilizar de ser así?

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=10680#respond)

Ramón [Admin Jarroba] dice:

24/02/2015 a las 16:00 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-10768)

¡Hola! una interfaz la puedes crear en cualquier momento. Herramientas como tal es aplicar interfaces:
http://docs.oracle.com/javase/tutorial/java/concepts/interface.html (http://docs.oracle.com/javase/tutorial/java/concepts/interface.html)

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=10768#respond)

Jose Antonio dice:

27/11/2014 a las 19:50 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-

thread-runnable/#comment-1392)

Muy buena la explicacion de hilos. Aunque tengo una duda. Utilizando Thread si el numero de personas es mayor que el número de cajeras como lo resolvemos?

Gracias!!

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1392#respond)

Richard [Admin Jarroba] (http://www.jarroba.com) dice:

29/11/2014 a las 11:37 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1393)

Hola Jose Antonio.
Muy buena pregunta la que planteas. Para hacer eso requiere un poco mas de control sobre los threads y aunque no es muy complejo, requiere una explicación bastante detallada como para explicarlo en un comentario. En breve publicaremos un tutorial de como hacer eso que planteas ya que estamos viendo que las entradas que hacemos sobre multiproceso están teniendo bastante exito. Gracias por tu comentario y por motivarnos ha hacer un tutorial más sobre el tema que planteas.
En breve te daremos la solución en una nueva entrada.
SL2

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1393#respond)

jose dice:

12/11/2014 a las 02:53 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1391)

Los felicito, ejemplo completo y entendible con código acorde a la POO, tomare este y algunos otros artículos como referencias para las clases que dicto.

Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1391#respond)

Pedro May dice:

02/11/2014 a las 15:45 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-

thread-runnable/#comment-1390)

De lo mejor que he encontrado en la red sobre el tema de los hilos. Felicitaciones.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1390#respond)

Enrique dice:

29/10/2014 a las 00:13 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1388)

Saludos! Esta muy bien explicado el tema, sigan asi, muchas gracias. Me podrias decir que IDE utilizaste? esque me gustó porque esta muy legible el código, gracias!

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1388#respond)

Ramón [Admin Jarroba] (http://www.jarroba.com) dice:

29/10/2014 a las 00:34 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1389)

Hola Enrique,

El IDE que solemos utilizar para Java es Eclipse.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1389#respond)

alejandro puerto dice:

23/10/2014 a las 03:04 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1385)

Hola gracias por la informacion, sin embargo lo que estoy buscando es el consumo de un web service por medio de multithread, me explico tengo un web service con cuatro operaciones basicas suma, resta, multiplicacion y division. la idea es que apartir de una aplicacion de escritorio se consuma el web service haciendo uso de los hilos podrias ayudarme con ello?

yo lo he intentado de varias maneras pero siempre me ocurre que cuando llamo el metodo .start me sale un error que dice que es indefinido

este es el codigo de invocacion

package WebServiceMatematicas.hilos;
import WebServiceMatematicas.ImplOpMatematicas;
import WebServiceMatematicas.ImplOpMatematicasProxy;
public class Proceso {
public static void main(String[] args) {
ImplOpMatematicas hilo1 = new
ImplOpMatematicasProxy("http://localhost:8080/WebServicedistri/services/ImplOpMatematicas");
hilo1.start(); //Es aca donde me aparece el error
}
}

de antemano muchas gracias

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1385#respond)

Paco dice:

23/10/2014 a las 02:44 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1383)

Gran artículo!! Es un placer leer estos temas sobre java porque ayudáis a muchas personas. Tenéis mucho mérito con todo lo que hacéis. Esta web es buenísima. Enhorabuena!

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1383#respond)

Paco dice:

23/10/2014 a las 02:50 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1384)

Una pregunta, ¿Que diferencia hay entre crear hilos a través de la herencia de la clase Thread y crearlos a través de la implementación de la interfaz Runnable? ¿Alguna forma es la más usada por los programadores? ¿Existe alguna ventaja?

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1384#respond)

Ramón [Admin Jarroba] (http://www.jarroba.com) dice:

23/10/2014 a las 21:26 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1387)

Hay mucha discusión al respecto. Personalmente te recomiendo hacer el implements de Runnable, ya que dejas libre la herencia. Por el resto, heredar de Thread, este hereda de Runnable.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1387#respond)

Ramón [Admin Jarroba] (http://www.jarroba.com) dice:

23/10/2014 a las 21:14 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1386)

Gracias Paco, seguiremos esforzándonos por contenido de calidad 😀

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1386#respond)

Daniel dice:

12/09/2014 a las 14:57 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1382)

Saludos.

Soy nuevo en el área de la programación java. Y queria saber acerca de un ejemplo sencillo de como puedo manejar esto en programacion objeto. Por ejemplo como un semaforo.

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1382#respond)

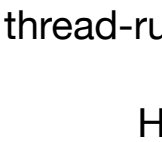
Nícolas (http://www.3universo.com) dice:

14/08/2014 a las 23:31 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-

thread-runnable/#comment-1380)

Hola, me parece interesante tu tutorial y la forma como explicas sobre multitarea, pero tengo una consulta, lo que haz mostrado es referente a la misma función cajera, osea hacer dos tareas simultáneas usando la misma función cajera, pero como se haría si fueran dos funciones distintas?
Por ejemplo quiero enviar y recibir datos por socket, mi cliente en android y mi servidor en java, por el lado de mi servidor quiero que haga una multitarea, por un lado recibiendo el dato del cliente y por el otro lado en todo momento leer si ingreso un dato por teclado y enviarlo al cliente. El problema es que no se como hacer que me lea el teclado en todo momento. Tengo en un while(true) la ejecución de recepción de datos del cliente, pero imagino que tendría que usar multitarea para que me lea en todo momento el teclado y enviarlo al cliente.
Agradecería tus comentarios!!!
Nícolas

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1380#respond)



Ramón [Admin Jarroba] (http://www.jarroba.com) dice:

16/08/2014 a las 12:32 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1381)

Hola Nícolas,

se haría de manera similar pero con dos clases diferentes. Aquí la complicación radicaría en la comunicación entre las clases que extienden de Thread, para no saltarte la seguridad de hilos (por ejemplo, evitando el interbloqueo).
De cualquier manera, para Android no necesitas preocuparte por los hilos, ya que los eventos te solucionan esta parte. Para leer los cambios (introducción por teclado) de un EditText puedes utilizar:

```
EditText et = (EditText) findViewById(R.id.miViewEditText);


      et.addTextChangedListener(new TextWatcher() {
public void afterTextChanged(Editable s) {
// Hacer algo después de cambiar el texto
}
```

```
      public void beforeTextChanged(CharSequence s, int start, int
count, int after) {
// Hacer algo antes de cambiar el texto
}



      public void onTextChanged(CharSequence s, int start, int
before, int count) {
// Hacer algo al momento de cambiar el texto
}
});
}
```

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1381#respond)

Andres dice:

16/07/2014 a las 16:23 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1377)

Hola amigos, excelente tutorial la verdad.
Podrian hacer un tutorial acerca de la sincronizacion de Hilos?
He estado buscando un poco, pero no encuentro nada asi de bien explicado como lo hacen ustedes.
Muchas gracias!

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1377#respond)

Ramón [Admin Jarroba] (http://www.jarroba.com) dice:

16/07/2014 a las 19:18 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1378)

Hola Andrés,

tenemos muchos proyectos entre los que se encuentran más tutoriales sobre hilos. Apuntamos tu sugerencia y la daremos prioridad; avisar que no lo publicaremos pronto, nos gusta hacer los artículos lo más perfectos y fáciles de comprender posible eso requiere tiempo 😉

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1378#respond)

Andres dice:

17/07/2014 a las 03:22 (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/#comment-1379)

Genial, a mi me gusta mucho Java y siempre ando aprendiendo algo nuevo.
Como les dije antes su pagina es muy ilustrativa y ayuda mucho a entender de mejor forma este lenguaje.
Gracias!

💬 Responder (http://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/?replytocom=1379#respond)

Deja un comentario

Tu dirección de correo electrónico no será publicada. Los campos necesarios están marcados *

Nombre *

Correo electrónico *

Web

Publicar comentario

(/)

Siguenos en:

(http://www.youtube.com/user/Jarrobaweb) (https://twitter.com/JarrobaWeb)

(https://www.facebook.com/jarrobaWeb) (https://github.com/jarroba?tab=repositories)

(https://plus.google.com/103352032205227460108/)

Contacto: jarrobaweb@gmail.com (mailto:jarrobaweb@gmail.com)

FAQ (HTTP://JARROBA.COM/FAQ/)

VISITAS (HTTP://JARROBA.COM/VISITAS/)

GASTOS

SITE-MAP (HTTP://JARROBA.COM/SITEMAP_INDEX.XML)

By Reimon & Richard