El interface Java Runnable es uno de los interfaces más utilizados en el lenguaje ya que se encarga de ejecutar una tarea de forma concurrente. Normalmente cuando utilizamos el interface Runnable lo implementamos a través del uso de una clase. Sin embargo a partir de Java 8 podemos realizar las operaciones de una forma más sencilla .Sin tener tanto lio vamos a ver un par de ejemplos:

```
package com.arquitecturajava.complemntario;

public class Tarea implements Runnable {

	@Override
	public void run() {
		for (int i=0;i<5; i++) {
			try {
				Thread.sleep(200);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			System.out.println("tarea A" +i);
		}
	}

}
```

Java Runnable y sus problemas

Aquí tenemos una sencilla clase Java que se encarga de implementar Runnable. Runnable es la interface de Java que dispone de un método run y nos permite ejecutar una Tarea en paralelo desde un programa main usando la clase Thread.

```
package com.arquitecturajava.complemntario;
```

```
public class Principal {

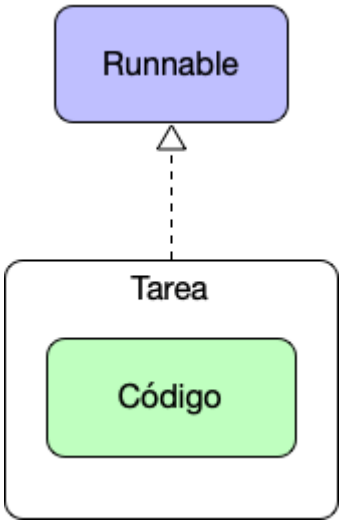
	public static void main(String[] args) {
		Thread t= new Thread(new Tarea());
		t.start();
	}

}
```

Es cuestión de crear la Tarea y ejecutar el método start(). Eso arrancará el nuevo hilo de ejecución y veremos cómo se imprimen los mensajes de la Tarea por la Consola usando Java Runnable.

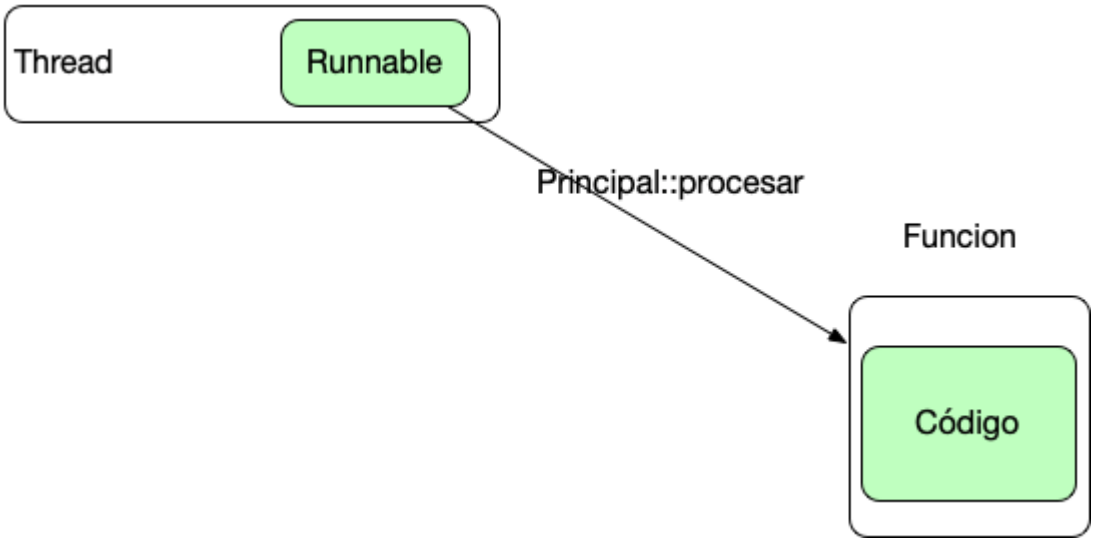
```
tarea A0
tarea A1
tarea A2
tarea A3
tarea A4
```

Esto siempre ha sido un poco laborioso ya que para ejecutar una funcionalidad concreta en concurrente nos vemos obligados a implementar una nueva clase cuando simplemente queremos ejecutar un pequeño código en concurrente.



Java 8 Lambdas y simplificación

Java 8 nos puede ayudar a simplificar como trabajamos con programación concurrente. En estos momentos necesitamos crear una clase nueva que implemente el interface Runnable para poder pasarla como parámetro al Thread. Sin embargo es posible simplificarlo ya que el interface Runnable es una interface Funcional y solo necesita un método para ejecutar. Por lo tanto podemos usar un método de referencia y apuntar al método que nos interese procesar de forma concurrente a través de un Thread.



Vamos a ver en código cómo nos podemos librar de la clase Tarea y procesar directamente

el flujo concurrente.

```
package com.arquitecturajava.complemntario;

public class Principal {

	public static void main(String[] args) {
		Thread t2= new Thread (Principal::procesar);
		t2.start();

	}
	public static void procesar() {
		for (int i =0;i<5;i++) {
			try {
				Thread.sleep(200);
			} catch (InterruptedException e) {
				// TODO Auto-generated catch block
				e.printStackTrace();
			}
			System.out.println("proceso A"+i);
		}
	}

}
```

Otros artículos relacionados

- Java Thread concurrencia y Runnables
- Java Executor Service y Threading
- Java wait notify y threads
- Java 8 Lambda Expressions (I)

- ¿Qué es un Java Lambda?
- Oracle Runnable