Programación multihilo

La programación multihilo es el tipo de programación que “aparentemente” nos permite correr varias secuencias de código a la vez en un mismo programa.
Se usa aparentemente porque como los procesos comparten una misma PC, en caso de que este solo cuente con un núcleo de procesamiento, lo que sucede es que las secuencias van alternando el PC para correr sus líneas de código, este proceso se realiza demasiado rápido y es imperceptible para el ser humano, por lo que parece que ambos procesos se ejecutan a la vez.[1]
Para entender mejor la diferencia entre un programa que use la programación multihilo y la que no veamos las siguientes imágenes:

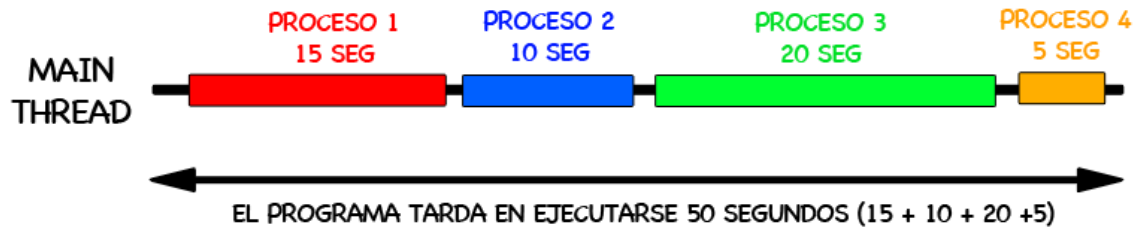


Imagen 1. Programa sin programación multihilo[2]

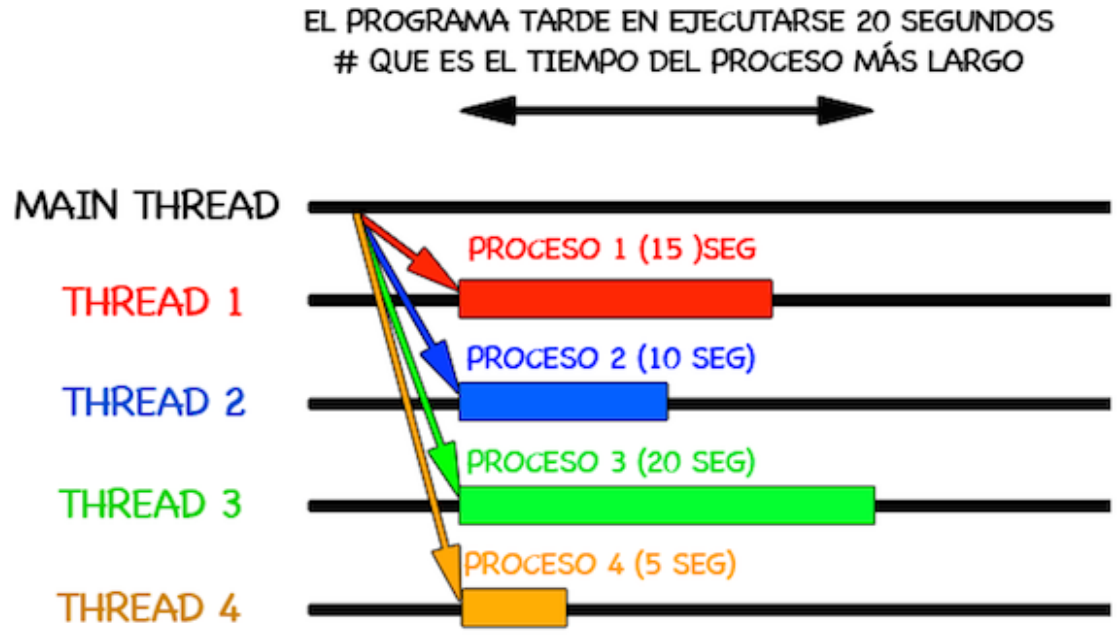


Imagen 2. Programa con programación multihilo[2]

Ejemplo:

Para el ejemplo de la programación multihilo tenemos el caso donde tenemos un proceso que se ejecutara por x segundos, durante su ejecución mostrará por consola cuantos segundos han pasado desde que se inició el programa hasta llegue a su tiempo límite.

Existen dos manera de usar la programación multihilo en Java:

- **Implementando la interfaz' Runnable':**

Clase 'Proceso'

Implementación de la interfaz Runnable

```
public class Proceso implements Runnable{

    private int id;
    private int segundos;

    public Proceso(int id, int segundos) {
        this.id = id;
        this.segundos = segundos;
    }

    public void run(){
        System.out.println("El proceso "+ this.id +" se ejecutara por "+ this.segundos +" segundos");

        for(int i=1; i<= this.segundos; i++) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Proceso: "+ this.id +"  Segundo: "+ i);
        }
        System.out.println("El proceso "+ this.id+ " termino");
    }

}
```

Todo lo que se encuentre contenido en la función run es lo que se ejecutara simultaneamente en el programa

Le decimos al programa que espere un segundo antes de que ejecute la proxima instrucción

Clase 'Main'

```
import java.util.Random;

public class Main{

    public static void main(String[] args)  {
        Random numAleatorio = new Random();
        int n = numAleatorio.nextInt(10);
        int n2 = numAleatorio.nextInt(10);

        Proceso proceso1 = new Proceso(1, n);
        Proceso proceso2 = new Proceso(2,n2);

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

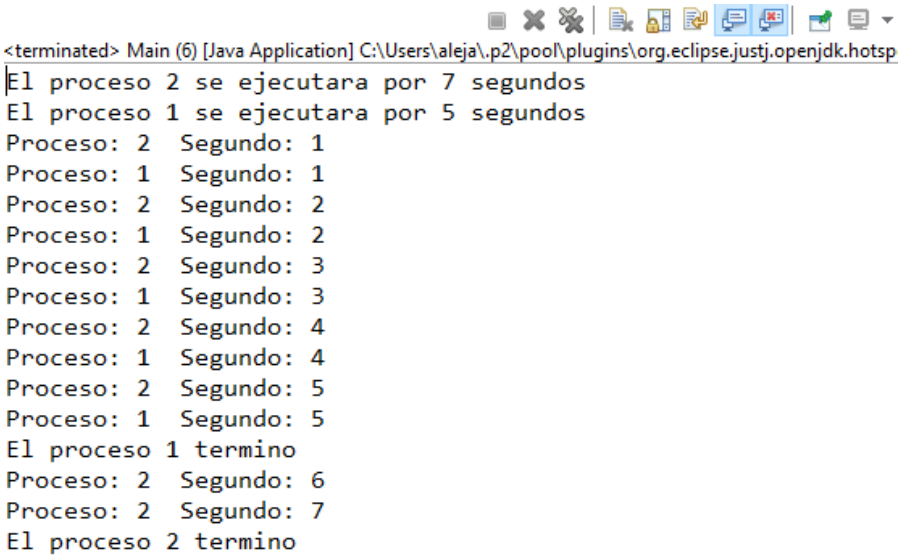
Generamos un número random para cada proceso

Se instancian los procesos con su respectivo id y tiempo maximo de ejecución

Le decimos al programa que cada proceso es un hilo

Llamamos el método run de cada proceso

Resultado en consola



- **Heredando la clase 'Threads'**
  Funciona de manera muy similar que con el Runnable, su diferencia radica que como cada objeto de la clase queda instanciado como un hilo, en el main solo es necesario llamar la función start().

Clase 'Proceso'

Herencia de la clase Thread

```
public class Proceso extends Thread{

    private int id;
    private int segundos;

    public Proceso(int id, int segundos) {
        this.id = id;
        this.segundos = segundos;
    }

    public void run(){
        System.out.println("El proceso "+ this.id +" se ejecutara por "+ this.segundos +" segundos");

        for(int i=1; i<= this.segundos; i++) {
            try {
                Thread.sleep(1000);
            } catch (InterruptedException e) {
                e.printStackTrace();
            }
            System.out.println("Proceso: "+ this.id +"  Segundo: "+ i);
        }
        System.out.println("El proceso "+ this.id+ " termino");
    }

}
```

Todo lo que se encuentre contenido en la función run es lo que se ejecutara simultaneamente en el programa

Le decimos al programa que espere un segundo antes de que ejecute la proxima instrucción

Clase ‘Main’

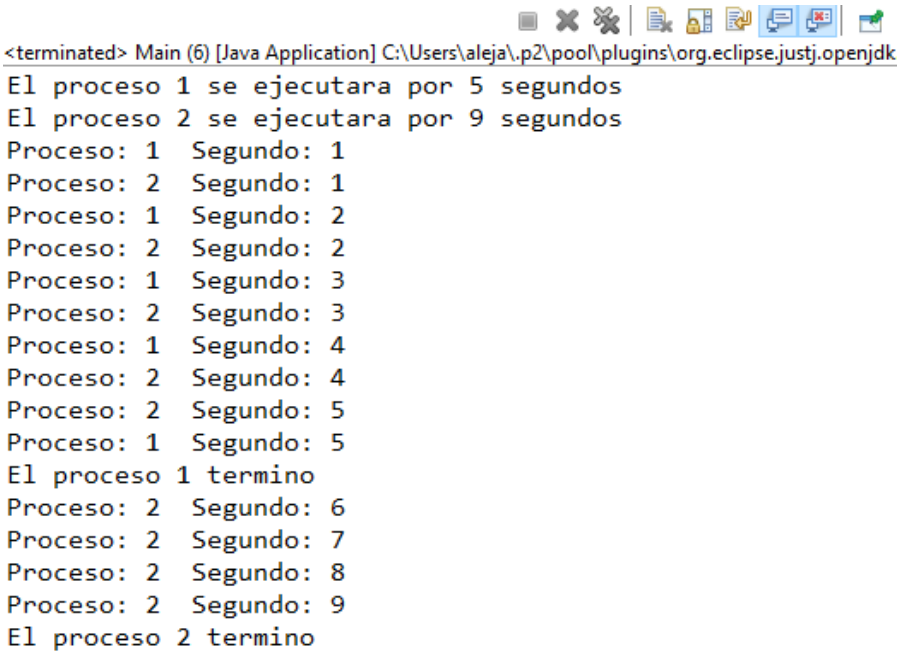
```
import java.util.Random;

public class Main{

    public static void main(String[] args)  {
        Random numAleatorio = new Random();
        int n = numAleatorio.nextInt(10);
        int n2 = numAleatorio.nextInt(10);

        Proceso proceso1 = new Proceso(1, n);
        Proceso proceso2 = new Proceso(2,n2);

        proceso1.start();
        proceso2.start();
    }
}
```

Generamos un número random para cada proceso

Se instancian los procesos con su respectivo id y tiempo maximo de ejecución

iniciamos el proceso de cada hilo ejecutando el método run()

Resultado en consola



Cabe mencionar que tanto la clase ‘Threads’ como la interfaz ‘Runnable’ se encuentran predefinidas por Java.

Referencias

1. Tutorial de Java - Hilos y Multihilo. (2021). Dis.um.es. http://dis.um.es/~bmoros/Tutorial/parte10/cap10-1.html
2. https://www.facebook.com/richard.mgr. (2014, May 23). Multitarea e Hilos en Java con ejemplos (Thread & Runnable) - Jarroba. Jarroba. https://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/