Aplicaciones para Ambientes Distribuidos

Laboratorio # 1 – Prof. Regis Rivera

Ernesto Crespo – 8-929-1657

Objetivo: Crear aplicaciones sencillas tipo consola en Java para aplicar primeros conceptos de concurrencia

Nota: Los nombres de archivos de cada aplicación están dados en el icono 

Laboratorio 1.1: Crear una salida de N números aplicando concurrencia

lab11.java

```
import java.lang.Math ;

class EjemploThread extends Thread {
int numero;

EjemploThread (int n) {
    numero = n;
}

public void run() {
try {
    while (true) {
        System.out.println (numero);
        sleep((long)(1000*Math.random()));
    }
}
    catch (InterruptedException e) { return; } // acaba este thread
}
    public static void main (String args[]) {
        for (int i=0; i<10; i++)
            new EjemploThread(i).start();
    }
}
```

Aspectos interesantes:

- Explique la salida del programa
  El programa imprimirá un numero entre 0 y 9, en un tiempo aleatorio de 0 y 1 segundo
- Mencione cuantos hilos se ejecutaron
  Se ejecutaron 10 hilos ya que se crean 10 instancias de EjemploThread
- ¿Qué haría si se quisiera agregar un segundo hilo al programa? Explique
  Hago otra instancia nueva, seguida del llamado a new EjemploThread(i).start()

Laboratorio 1.2: Ejecutar 3 tareas concurrentes

lab12.java

```java
import java.util.concurrent.*;

public class EjemploConcurrenciaJava {
    public static void main(String[] args) {
        // Crear un ExecutorService para administrar hilos
        ExecutorService executor = Executors.newFixedThreadPool(3);

        // Crear tareas (Runnable) que se ejecutaran en paralelo
        Runnable tarea1 = () -> {
            for (int i = 0; i < 5; i++) {
                System.out.println("Tarea 1: " + i);
            }
        };
        Runnable tarea2 = () -> {
            for (int i = 0; i < 5; i++) {
                System.out.println("Tarea 2: " + i);
            }
        };
        Runnable tarea3 = () -> {
            for (int i = 0; i < 5; i++) {
                System.out.println("Tarea 3: " + i);
            }
        };
        // Ejecutar las tareas en hilos separados
        executor.execute(tarea1);
        executor.execute(tarea2);
        executor.execute(tarea3);

        // Cerrar el ExecutorService cuando ya no se necesite
        executor.shutdown();
    }
}
```

Aspectos interesantes:

- Explique la salida del programa

  Se muestral las 3 tareas realizadas en desorden porque se ejecutan en hilos separados simultáneamente. Por eso es muy probable que cada vez que se ejecuta el código se obtenga una salida distinta con orden de tareas diferentes a la anterior.
- ¿Cuántos hilos se ejecutan? ¿Por qué?

  Este programa utiliza 3 hilos para ser ejecutados, los mismos son administrados por el ExectuorService y este mismo fija el ThreadPool en 3, cada tarea es ejecutada en su propio hilo simultáneamente.
- ¿Qué haría si se quisiera agregar un hilo adicional al programa? Explique

  Crearía una nueva tarea Runnable y la ejecutaría con el ExecutorSerivce

lab13.java

```java
public class NoSincronizada extends Thread {
    static int n = 1;

    public void run() {
        for (int i = 0; i < 10; i++) {
        System.out.println(n);
        n++;
        }
    }

    public static void main(String args[]) {
        Thread thr1 = new NoSincronizada();
        Thread thr2 = new NoSincronizada();

        thr1.start();
        thr2.start();
    }
}
```

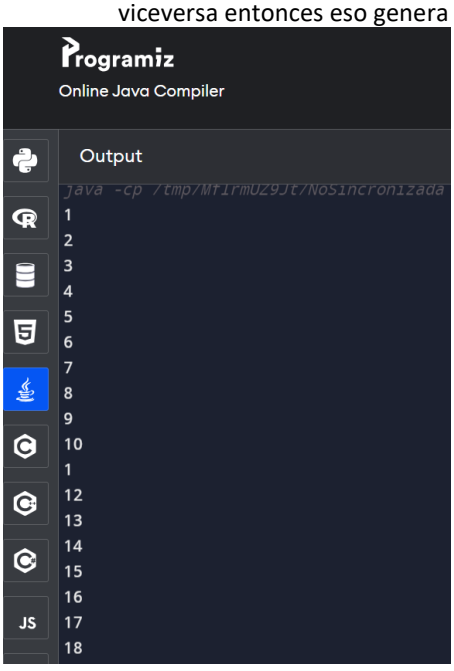
Aspectos interesantes:

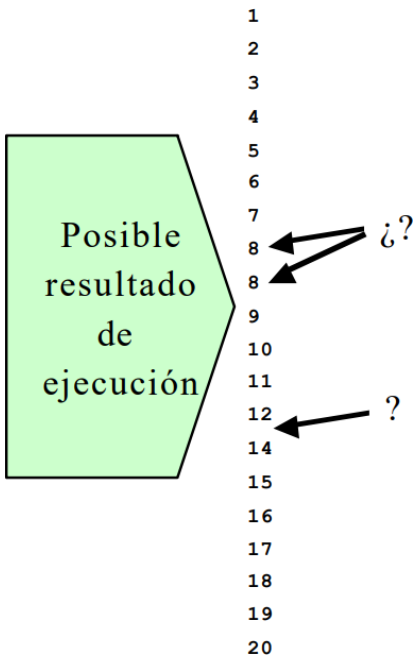
- Explique la salida del programa

  Este programa crea 2 hilos que ejecutaran la misma tarea de imprimir el valor de n, 10 veces cada hilo, dando como resultado la impresión de 20 números finales

- ¿Cuál de las opciones de la imagen aparecen en su ejecución? ¿Por qué?

  En la imagen que adjunto me muestra un 1 en lugar de 11 en la sucesión de impresión, esto se da debido que tenemos 2 hilos diferentes accediendo a la misma variable y haciendo el mismo proceso de manera intercalada y no sincronizada, el hilo 1 no sabe que el hilo 2 accedió a la variable o viceversa entonces eso genera esta diferencia.




Posible resultado de ejecución
1
2
3
4
5
6
7
8
8
9
10
11
12
14
15
16
17
18
19
20
¿?
?