Nombre de la práctica:	PROGRAMACIÓN CONCURRENTE (JAVA)
No. de práctica:	2
Alumna:	LAURA KAREN NAVA GÁLVEZ

INTRODUCCIÓN

Un Lenguaje de Programación será concurrente si posee las estructuras necesarias para definir y manejar diferentes tareas (hilos de ejecución) dentro de un programa.

La concurrencia es, en esencia, el poder realizar múltiples cosas en el mismo tiempo, pero, no específicamente en paralelo.

Una de las formas más sencillas de comprender la concurrencia es imaginar a una persona la cual trabaja en múltiples tareas al mismo tiempo, y que rápidamente cambia de una tarea a otra.

La principal diferencia entre concurrencia y paralelismo recae en la forma en que se realizan las tareas. Cuando se ejecutan tareas de forma concurrente a estas se les asigna un x periodo de tiempo antes de cambiar de tarea, será en ese periodo en el cual se inicie, continúe, o se complete la tarea, por otro lado, si ejecutamos tareas en paralelo, las tareas se realizarán de forma simultánea, comenzarán y finalizarán sin interrupciones.

Si lo que se quiere es implementar concurrencia en los programas una muy buena idea será utilizar Threads, por otro lado, si lo que se quiere es implementar paralelismos es mejor utilizar procesos.

OBJETIVO

• Comprender la programación concurrente, a través de algunos ejemplos prácticos con el lenguaje de programación de Java

• Codificar ejemplos de programación con hilos en el lenguaje de programación de Java.

• Entender el concepto de concurrencia por medio de ejemplos prácticos en el lenguaje de programación de Java

DESARROLLO DE LA ACTIVIDAD PRÁCTICA

1.

```
package principal;
import java.util.Random;
/**
 *
 * @author karen
 */
public class Principal extends Thread{
  private  static double[] vec= new double[0000000];
  private int inicio, fin;

public Principal (int inicio, int fin) {
  this.inicio = inicio;
  this.fin = fin;
}

  public static void main(String[] args) {
    iniciavec();
    vec_NOconcurrente();
    vec concurrente();
  }
  private static void iniciavec() {
    Random rand = new Random (System.nanoTime());
    for(int i=0;i<vec.length;i++){
      vec [i]= rand.nextInt();
    }
  }
  //INSTRUCCIONES SECUENCIALES
  private static void vec_NOconcurrente() {
    double tiempo= System.nanoTime();
    for(int i=0;i<vec.length;i++){
      vec [i]/=10;
      vec [i]*=10;
```

```
    for(int i=0;i<vec.length;i++){
      vec [i]/=10;
      vec [i]*=10;
      vec [i]/=10;
    }
    System.out.println("Version NO concurrente: "+((System.nanoTime()-tiempo)/1000000)+ " milisegundos");
  }
  public void run(){
    for(int i=inicio;i<fin;i++){
      vec [i]/=10;
      vec [i]*=10;
      vec [i]/=10;
    }
  }
    //INSTRUCCIONES CONCURRENTES
    private static void vec_concurrente() {
      int nproc= Runtime.getRuntime().availableProcessors();
      int inicio = 0, fin = 0;
      Principal[] prin = new Principal[nproc];
      double tiempo = System.nanoTime();

      for(int i=0;i<nproc;i++){
        inicio=fin;
        fin+=vec.length/nproc;
        prin[i]= new Principal (inicio, fin);
        prin[i].start();
      }
      for(int i=0;i<nproc;i++){
        try{
          prin[i].join();
        }catch(Exception e){}
      }
      System.out.println("Versión concurrente: "+((System.nanoTime() - tiempo) / 1000000)+  " milisegundo");
```

2.

```java
package threadejemplo;
/**
 * @author karen
 */
class ThreadEjemplo extends Thread {

  public ThreadEjemplo(String str){
    super(str);
  }
  public void run(){
    for(int i=0;i<10;i++){
      System.out.println(i+""+getName());
      System.out.println("Termina thread"+getName());
    }
  }

  public static void main(String[] args) {
    new ThreadEjemplo("Lenguajes").start();
    new ThreadEjemplo("Concurrentes").start();
    System.out.println("Termina thread main");
  }
}
```

3.

```java
package threadejemploo;
/**
 * @author karen
 */
public class ThreadEjemploo implements Runnable {

    public void run(){
        for (int i=0;i<5; i++)
        System.out.println(i +" "+
        Thread.currentThread().getName());
        System.out.println("Termina thread " +
        Thread.currentThread().getName());
    }

    public static void main (String [] args){
        new Thread (new ThreadEjemploo(), "Lenguajes").start();
        new Thread (new ThreadEjemploo(), "Concurrentes").start();
        System.out.println("Termina thread main");
    }

}
```

RESULTADO (Producto de aprendizaje)

1.

```
44          //INSTRUCCIONES CONCURRENTES
```

Output - principal (run)

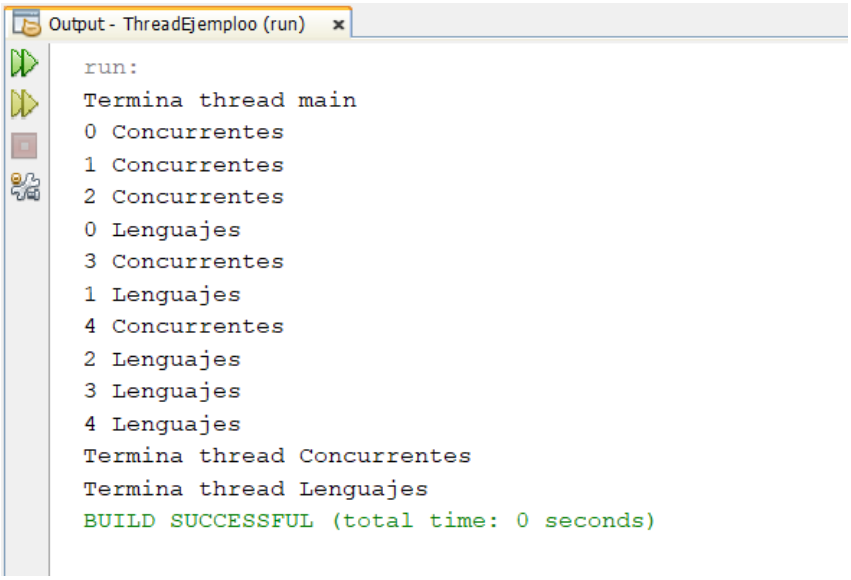
```
run:
Version NO concurrente: 0.0049 milisegundos
Versión concurrente: 2.0545 milisegundo
BUILD SUCCESSFUL (total time: 0 seconds)
```

2.

Output - ThreadEjemplo (run)

```
run:
Termina thread main
0Concurrentes
0Lenguajes
Termina threadConcurrentes
1Concurrentes
Termina threadConcurrentes
Termina threadLenguajes
2Concurrentes
Termina threadConcurrentes
1Lenguajes
3Concurrentes
Termina threadConcurrentes
Termina threadLenguajes
2Lenguajes
4Concurrentes
Termina threadLenguajes
Termina threadConcurrentes
3Lenguajes
Termina threadLenguajes
5Concurrentes
4Lenguajes
Termina threadConcurrentes
Termina threadLenguajes
6Concurrentes
5Lenguajes
Termina threadConcurrentes
Termina threadLenguajes
7Concurrentes
Termina threadConcurrentes
8Concurrentes
Termina threadConcurrentes
6Lenguajes
9Concurrentes
Termina threadLenguajes
7Lenguajes
Termina threadConcurrentes
Termina threadLenguajes
8Lenguajes
Termina threadLenguajes
9Lenguajes
Termina threadLenguajes
BUILD SUCCESSFUL (total time: 0 seconds)
```

3.


Output - ThreadEjemploo (run)
run:
Termina thread main
0 Concurrentes
1 Concurrentes
2 Concurrentes
0 Lenguajes
3 Concurrentes
1 Lenguajes
4 Concurrentes
2 Lenguajes
3 Lenguajes
4 Lenguajes
Termina thread Concurrentes
Termina thread Lenguajes
BUILD SUCCESSFUL (total time: 0 seconds)

DISCUSIÓN

La programación concurrente se ocupa del estudio y desarrollo de programas que puedan realizar varias tareas “al mismo tiempo”.

La programación concurrente funciona bajo el concepto de que los procesos, si bien se ejecutan al mismo tiempo, no ocurre de manera paralela, es decir todos a la vez. En este tipo de procesamiento, se pueden procesar múltiples datos en un mismo procesador, pero indicando un tiempo máximo de límite de ejecución.

Básicamente, la concurrencia es una forma de estructurar una solución que puede ser paralelizadle, aunque esto no siempre tenga que ser de esta manera.

CUESTIONARIO

1.- ¿Cuáles son las ventajas de utilizar la programación concurrente? Su implementación permite hacer una mejor utilización de los procesadores incluidos en ordenador, pues estos cada vez traen mayor cantidad de núcleos. Mejor utilización de los recursos del microprocesador.

2.- ¿Cuáles son las desventajas de utilizar la programación concurrente? Cambio de contexto y velocidad de ejecución. Desconocer cuál es el orden en que se ejecutarán los programas.

3.- ¿Son iguales la programación paralela y la concurrente? Explica ¿por qué? No, ya que la concurrencia tiene que ver con el diseño del software, mientras que el paralelismo tiene que ver con la ejecución.

4.- ¿Cuáles son las diferencias entre la programación secuencial y la programación concurrente? La forma en que se realizan las tareas. La programación concurrente logra que múltiples instrucciones se ejecuten a la vez, logrando así mayor velocidad en la ejecución de nuestros programas.La programación secuencial hace que una instrucción siga a otra instrucción después de que ésta haya finalizado.

REFERENCIAS

Moya Ricardo (2014). Multitarea e hilos en Java, disponible en: https://jarroba.com/multitarea-e-hilos-en-java-con-ejemplos-thread-runnable/

Vallejo D., González C. y Albusac J. (2016) Programación concurrente en tiempo real, 3ra Edición, ISBN:978-1518609261. Disponible en: http://190.57.147.202:90/xmlui/bitstream/handle/123456789/445/Programacion-Concurrente-y-Tiempo-Real.pdf?sequence=1&isAllowed=y