Practica 4: Hilos con Java

Marzo 2017

1 Introducción

Un hilo es una unidad de ejecución carece de zona de memoria propia reservada (usa la del proceso principal) y tiene como finalidad de realizar una tarea adicional al flujo del programa principal o la de aprovechar el hardware disponible, por ejemplo multiprocesador o multicore, para para decrementar el tiempo de ejecución . Los hilos se crean a través de un hilo principal, el cual se corresponde unívocamente con el proceso principal. Existen diferentes tecnologías/herramientas para crear hilos. En esta práctica se utilizarán los hilos en la tecnología Java.

2 Creación de hilos

Para explicar como crear hilos con Java utilizaremos el siguiente ejemplo: **sumar los elementos de un arreglo**. Considere la siguiente clase:

```
import java.util.Arrays;
import java.util.Random;

public class OperArreglos {


    static void inicializarArreglo(int[] arreglo,int n){

        Random azar = new Random();
        int i;
        for(i=0;i<arreglo.length;i++){
            arreglo[i]= azar.nextInt(n);
        }

    }

    static void imprimir(int[] arreglo){
        System.out.println(Arrays.toString(arreglo));
    }

}
```

2.1 Creación de hilos mediante Thread

Se puede utilizar la clase Thread para la creación de hilos.

```
public class SumarArreglo extends Thread{

    static int N=1;
    static int HILOS = 1;
    static int[] arreglo = null;
    private int id=-1;


    SumarArreglo(int id){

        this.id=id;

    }

    public void run(){

        int i,suma=0;
        int tam=arreglo.length/HILOS;
        int resto = (arreglo.length%HILOS);
        int ini = (id*tam);
        int fin = ini+tam;


        for(i=ini;i<fin;i++){

            suma = suma + arreglo[i];
        }
        if(resto>id){

            suma = suma + arreglo[(arreglo.length-1)-id];
        }

        System.out.println("Hilo "+id+" PID: "+this.getId()+" Suma Local:"+suma);

    }



    public static void main(String[] arg){

        int i;
        Thread[] trabajadores = null;

        try{

            N = Integer.parseInt(arg[0]);
            HILOS = Integer.parseInt(arg[1]);

        }catch(NumberFormatException e){

            System.out.println("Error: No es posible convertir a entero");
            System.exit(0);
```

```
        }

        arreglo = new int[N];

        trabajadores = new Thread[HILOS];

        OperArreglos.inicializarArreglo(arreglo,100);

        OperArreglos.imprimir(arreglo);


        for(i=0;i<HILOS;i++){

            trabajadores[i] = new SumarArreglo(i);
            trabajadores[i].start();

        }

        for(i=0;i<HILOS;i++){

            try {
                trabajadores[i].join();
            } catch (InterruptedException e) {

                System.out.println("Error: en la espera del hilo");
            }

        }

    }
}
```

Para obtener la suma total, tendríamos que recuperar la suma parcial de cada hilo. Lo anterior lo podríamos hacer usando un arreglo donde en cada entrada un hilo guarde su resultado.

2.2 Creación de hilos mediante Runnable

```
public class SumarArregloInterface implements Runnable {

    static int N=1;
    static int HILOS = 1;
    static int[] arreglo = null;
    static int[] resultados=null;
    private int id=-1;


    SumarArregloInterface(int id){

        this.id=id;

    }

    @Override
    public void run() {
```

```
    int i,suma=0;
    int tam=arreglo.length/HILOS;
    int resto = (arreglo.length%HILOS);
    int ini = (id*tam);
    int fin = ini+tam;


    for(i=ini;i<fin;i++){

        suma = suma + arreglo[i];
    }
    if(resto>id){

        suma = suma + arreglo[(arreglo.length-1)-id];
    }

    resultados[id] = suma;

}


public static void main(String[] arg){

    int i;
    int sum=0;
    Thread[] trabajadores = null;

    try{

        N = Integer.parseInt(arg[0]);
        HILOS = Integer.parseInt(arg[1]);

    }catch(NumberFormatException e){

        System.out.println("Error: No es posible convertir a entero");
        System.exit(0);
    }

    arreglo = new int[N];
    trabajadores = new Thread[HILOS];
    resultados = new int[HILOS];

    OperArreglos.inicializarArreglo(arreglo,100);

    OperArreglos.imprimir(arreglo);

    for(i=0;i<HILOS;i++){

        trabajadores[i] = new Thread(new SumarArregloInterface(i));
        trabajadores[i].start();

    }

    for(i=0;i<HILOS;i++){
```

```
            try {
                trabajadores[i].join();
                sum = sum + resultados[i];
            } catch (InterruptedException e) {

                System.out.println("Error: en la espera del hilo");
            }

        }
        System.out.printf("Runnable: La suma total es:"+ sum);

    }

}
```

Sería muy práctico que en lugar de que cada hilo ejecutara un método void, pudiera regresar el resultado se calculo local.

2.3 Creación de hilos mediante Callable

```
import java.util.LinkedList;
import java.util.concurrent.Callable;
import java.util.concurrent.ExecutionException;
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.Future;

public class SumaArregloICall implements Callable<Integer> {

    static int N=1;
    static int HILOS = 1;
    static int[] arreglo = null;
    private int id=-1;


    SumaArregloICall(int id){

        this.id=id;

    }
    @Override
    public Integer call() throws Exception {
        int i,suma=0;
        int tam=arreglo.length/HILOS;
        int resto = (arreglo.length%HILOS);
        int ini = (id*tam);
        int fin = ini+tam;


        for(i=ini;i<fin;i++){

            suma = suma + arreglo[i];
        }
        if(resto>id){
```

```
            suma = suma + arreglo[(arreglo.length-1)-id];
        }
        System.out.println("PARCIAL:"+suma);
        return new Integer(suma);

    }


    public static void main(String[] arg){

        int sum = 0;


        try{

            N = Integer.parseInt(arg[0]);
            HILOS = Integer.parseInt(arg[1]);

        }catch(NumberFormatException e){

            System.out.println("Error: No es posible convertir a entero");
            System.exit(0);
        }

        arreglo = new int[N];

        OperArreglos.inicializarArreglo(arreglo,100);

        OperArreglos.imprimir(arreglo);


        LinkedList<Future<Integer>> valores = new LinkedList<Future<Integer>>();

        ExecutorService pool = Executors.newFixedThreadPool(HILOS);

        for (int i=0;i<HILOS;i++) {
            Callable<Integer> callable = new SumaArregloICall(i);
            Future<Integer> future = pool.submit(callable);
            valores.add(future);

        }

        for (Future<Integer> future : valores) {
            try {
                sum += future.get();

            } catch (InterruptedException | ExecutionException e) {
                System.out.println("Error: Al obtener el dato del hilo");
            }
        }

        System.out.printf("La suma total es:"+ sum);
        pool.shutdown();
    }

}
```

3 Ejercicios

Realizar un script para resolver los siguientes problemas.

1. Investigar como realizar los semáforos y barreras en Java.
2. Realizar la multiplicación de matrices mediante Thread.
3. Encontrar todas las soluciones del problemas de las N-Reinas mediante Callable.