Práctica: Hilos

José Luis Quiroz Fabián

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 un multiprocesador o procesador multi-núcleo. 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**.

2.1 Solución secuencial

La suma del contenido de un arreglo es muy simple de forma secuencial, es suficiente con tener una estructura de repetición que obtenga cada dato del arreglo y acumule estos en una variable (líneas 17-30).

```
1  package mx.uam.pc.simple;
2
3  import java.util.Arrays;
4  import java.util.Random;
5
6  public class SumaArreglo{
7
8      static int[] arreglo = null;
9
10
11     SumaArreglo(int n){
12
13         arreglo= new int[n];
14
15     }
16
17     public int  sumar(){
18
19         int i,suma=0;
```

```
20
21          for(i=0;i<arreglo.length;i++){
22
23              suma = suma + arreglo[i];
24          }
25
26          System.out.println("Thread id: "+Thread.currentThread().getId()
        +" Suma Local:"+suma);
27
28          return suma;
29
30      }
31
32
33      void inicializarArreglo(int[] arreglo,int n){
34
35          Random azar = new Random();
36          int i;
37          for(i=0;i<arreglo.length;i++){
38              arreglo[i]= azar.nextInt(n);
39          }
40
41      }
42
43       void imprimir(int[] arreglo){
44          System.out.println(Arrays.toString(arreglo));
45      }
46
47      public static void main(String[] args){
48
49          int n=0,resultado;
50
51          try{
52
53              n = Integer.parseInt(args[0]);
54
55          }catch(NumberFormatException e){
56
57              System.out.println("Error: No es posible convertir a entero
        ");
58              System.exit(0);
59          }
60
61          SumaArreglo sm = new SumaArreglo(n);
62
63          sm.inicializarArreglo(arreglo,100);
64
65          sm.imprimir(arreglo);
66
67          resultado=sm.sumar();
68
69
70          System.out.println("Suma:"+resultado);
71
72      }
73  }
```

practicas/hilos/simple/SumaArreglo.java

2.2 Creación de hilos mediante Thread

Una forma de crear los hilos es utilizar herencia sobre la clase **Thread** (línea 6). Al crear una instancia de la clase **Thread** es posible ejecutar el método **start()** (línea 91), el cual realiza la creación del hilo. Al crearse un hilo éste ejecuta el método **run()** (línea 20) de la instancia. Los datos compartidos por los hilos son generalmente **static**.

En particular, para el problema de la suma del contenido de un arreglo al tener un grupo de hilos, el arreglo se particiona entre el total de hilos y el resto se divide solo entre un sub-conjunto de ellos (líneas 20-42).

```
1  package mx.uam.pc.thread;
2
3  import java.util.Arrays;
4  import java.util.Random;
5
6  public class SumaArreglo extends Thread{
7
8      static int hilos = 1;
9      static int[] resultado=null;
10     static int[] arreglo = null;
11     private int id=-1;
12
13
14     SumaArreglo(int id){
15
16         this.id=id;
17
18     }
19
20     public void run(){
21
22         int i,suma=0;
23         int tam=arreglo.length/hilos;
24         int resto = (arreglo.length%hilos);
25         int ini = (id*tam);
26         int fin = ini+tam;
27
28
29         for(i=ini;i<fin;i++){
30
31             suma = suma + arreglo[i];
32         }
33         if(resto>id){
34
35             suma = suma + arreglo[(arreglo.length-1)-id];
36         }
37
38         System.out.println("Thread local id "+id+" Thread Id: "+this.
       getId()+" Suma Local:"+suma);
39
```

```
40          resultado[id] = suma;
41
42      }
43
44
45      static void inicializarArreglo(int[] arreglo,int n){
46
47          Random azar = new Random();
48          int i;
49          for(i=0;i<arreglo.length;i++){
50              arreglo[i]= azar.nextInt(n);
51          }
52
53      }
54
55      static void imprimir(int[] arreglo){
56          System.out.println(Arrays.toString(arreglo));
57      }
58
59      public static void main(String[] args){
60
61          int i,n=0;
62          Thread[] trabajadores = null;
63          int sumaFinal=0;
64          try{
65
66              System.out.println(args.length);
67
68              n = Integer.parseInt(args[0]);
69              hilos =  Integer.parseInt(args[1]);
70
71          }catch(NumberFormatException e){
72
73              System.out.println("Error: No es posible convertir a entero
    ");
74              System.exit(0);
75          }
76
77          arreglo =  new int[n];
78
79          trabajadores = new Thread[hilos];
80
81          resultado = new int[hilos];
82
83          inicializarArreglo(arreglo,100);
84
85          imprimir(arreglo);
86
87
88          for(i=0;i<hilos;i++){
89
90              trabajadores[i] = new SumaArreglo(i);
91              trabajadores[i].start();
92
93          }
94
95          for(i=0;i<hilos;i++){
```

```
96
97              try {
98                  trabajadores[i].join();
99
100                 sumaFinal = sumaFinal + resultado[i];
101
102             } catch (InterruptedException e) {
103
104                 System.out.println("Error: en la espera del hilo");
105             }
106
107
108         }
109         System.out.println("Suma Thread:"+sumaFinal);
110
111     }
112 }
```

practicas/hilos/thread/SumaArreglo.java

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.

2.3 Creación de hilos mediante Runnable

La creación de hilos mediante Runnable es similar a Threah, solo que en lugar de utiliza se utiliza una interfaz, la misma **Runnable**. La única diferencia que se puede resaltar es que para la creación se requiere usar el constructor de **Thread** que recibe como parámetro un **Runnable** (línea 89).

```
1  package mx.uam.pc.runnable;
2
3  import java.util.Arrays;
4  import java.util.Random;
5
6  public class SumaArreglo implements Runnable{
7
8      static int hilos = 1;
9      static int[] resultado=null;
10     static int[] arreglo = null;
11     private int id=-1;
12
13
14     SumaArreglo(int id){
15
16         this.id=id;
17
18     }
19
20     public void run(){
21
22         int i,suma=0;
23         int tam=arreglo.length/hilos;
24         int resto = (arreglo.length%hilos);
25         int ini = (id*tam);
```

```
26          int fin = ini+tam;
27
28
29          for(i=ini;i<fin;i++){
30
31              suma = suma + arreglo[i];
32          }
33          if(resto>id){
34
35              suma = suma + arreglo[(arreglo.length-1)-id];
36          }
37
38          System.out.println("Thread local id "+id+" Thread Id: "+Thread.
    currentThread().getId()+" Suma Local:"+suma);
39
40          resultado[id] = suma;
41
42      }
43
44
45      static void inicializarArreglo(int[] arreglo,int n){
46
47          Random azar = new Random();
48          int i;
49          for(i=0;i<arreglo.length;i++){
50              arreglo[i]= azar.nextInt(n);
51          }
52
53      }
54
55      static void imprimir(int[] arreglo){
56          System.out.println(Arrays.toString(arreglo));
57      }
58
59      public static void main(String[] args){
60
61          int i,n=0;
62          Thread[] trabajadores = null;
63          int sumaFinal=0;
64          try{
65
66
67              n = Integer.parseInt(args[0]);
68              hilos =  Integer.parseInt(args[1]);
69
70          }catch(NumberFormatException e){
71
72              System.out.println("Error: No es posible convertir a entero
    ");
73              System.exit(0);
74          }
75
76          arreglo =  new int[n];
77
78          trabajadores = new Thread[hilos];
79
80          resultado = new int[hilos];
```

```
81
82          inicializarArreglo(arreglo,100);
83
84          imprimir(arreglo);
85
86
87          for(i=0;i<hilos;i++){
88
89              trabajadores[i] = new Thread(new SumaArreglo(i));
90              trabajadores[i].start();
91
92          }
93
94          for(i=0;i<hilos;i++){
95
96              try {
97                  trabajadores[i].join();
98
99                  sumaFinal = sumaFinal + resultado[i];
100
101             } catch (InterruptedException e) {
102
103                 System.out.println("Error: en la espera del hilo");
104             }
105
106
107         }
108         System.out.println("Suma Runnable:"+sumaFinal);
109
110     }
111 }
```

practicas/hilos/runnable/SumaArreglo.java

El caso de Callable es muy interesante, en este modelo se crea un Pool de Hilos (línea 85) y se crea un pool de tareas (línea 88). Al someter una tarea los hilos pueden tomarla y ejecutarla (línea 89). Eventualmente cuando termine una tarea se pude recuperar su resultado (línea 96).

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

2.4 Creación de hilos mediante Callable

```
1  package mx.uam.pc.callable;
2
3  import java.util.LinkedList;
4  import java.util.concurrent.Callable;
5  import java.util.concurrent.ExecutionException;
6  import java.util.concurrent.ExecutorService;
7  import java.util.concurrent.Executors;
8  import java.util.concurrent.Future;
9  import java.util.Arrays;
10 import java.util.Random;
11
12 public class SumaArreglo implements Callable<Integer> {
```

```
13
14      static int hilos = 1;
15      static int[] arreglo = null;
16      private int id=-1;
17
18
19      SumaArreglo(int id){
20
21          this.id=id;
22
23      }
24      @Override
25      public Integer call() throws Exception {
26          int i,suma=0;
27          int tam=arreglo.length/hilos;
28          int resto = (arreglo.length%hilos);
29          int ini = (id*tam);
30          int fin = ini+tam;
31
32
33          for(i=ini;i<fin;i++){
34
35              suma = suma + arreglo[i];
36          }
37          if(resto>id){
38
39              suma = suma + arreglo[(arreglo.length-1)-id];
40          }
41          System.out.println("Thread local id "+id+" Thread Id: "+Thread.
        currentThread().getId()+" Suma Local:"+suma);
42          return suma;
43
44      }
45
46      static void inicializarArreglo(int[] arreglo,int n){
47
48          Random azar = new Random();
49          int i;
50          for(i=0;i<arreglo.length;i++){
51              arreglo[i]= azar.nextInt(n);
52          }
53
54      }
55
56      static void imprimir(int[] arreglo){
57          System.out.println(Arrays.toString(arreglo));
58      }
59
60      public static void main(String[] arg){
61
62          int sumaFinal = 0;
63          int n=0;
64
65          try{
66
67              n = Integer.parseInt(arg[0]);
68              hilos =  Integer.parseInt(arg[1]);
```

```
69
70          }catch(NumberFormatException e){
71
72              System.out.println("Error: No es posible convertir a entero
    ");
73              System.exit(0);
74          }
75
76          arreglo =  new int[n];
77
78          inicializarArreglo(arreglo,100);
79
80          imprimir(arreglo);
81
82
83          LinkedList<Future<Integer>> valores  = new  LinkedList<Future<
    Integer>>();
84
85          ExecutorService pool = Executors.newFixedThreadPool(hilos);
86
87          for (int i=0;i<hilos;i++) {
88              Callable<Integer> callable = new SumaArreglo(i);
89              Future<Integer> future = pool.submit(callable);
90              valores.add(future);
91
92          }
93
94          for (Future<Integer> future : valores) {
95              try {
96                  sumaFinal += future.get();
97
98              } catch (InterruptedException | ExecutionException e) {
99                  System.out.println("Error: Al obtener el dato del hilo"
    );
100             }
101         }
102
103         System.out.println("La suma callable es:"+ sumaFinal);
104         pool.shutdown();
105     }
106
107 }
```

practicas/hilos/callable/SumaArreglo.java

3 Ejercicios

- Realizar la suma de matrices (NxN) concurrente considerando N hilos y H hilos con H<N.
- Resolver el problema de las N-Reinas con hilos. Se debe conocer el total de soluciones para un tablero de NxN. N debe ser ingresado por argumentos de la función **main**.

```java
import java.util.Arrays;
public class NReinas {
    public static final int N=15;

    public static boolean esComida(int[][] tablero, int r, int c){

    int auxR,auxC;

    auxR=r-1;
    auxC=c;
    while(auxR>=0){
      if(tablero[auxR][auxC]==1){
        return true;
      }
      auxR--;
    }
    auxR=r-1;
    auxC=c+1;
    while(auxR>=0 && auxC< tablero.length){
      if(tablero[auxR][auxC]==1){
        return true;
      }
      auxR--;
      auxC++;
    }

    auxR=r-1;
    auxC=c-1;
    while(auxR>=0 && auxC>=0){
      if(tablero[auxR][auxC]==1){
        return true;
      }
      auxR--;
      auxC--;
    }
    return false;

  }


  public static boolean backtrackReinas(int[][] tablero, int r){
      int c;

      if(r==tablero.length)
        return true;
        for(c=0;c<tablero.length;c++){

          if(!esComida(tablero,r, c)){

            tablero[r][c]=1;
            if(backtrackReinas(tablero,r+1))
              return true;
            tablero[r][c]=0;
          }

        }
        return false;
```

```
58    }
59
60    public static void main(String[] arg){
61
62      int[][] tablero = new int[N][N];
63
64      if(backtrackReinas(tablero,0)){
65
66        for(int c=0;c<tablero.length;c++){
67          System.out.println(Arrays.toString(tablero[c]));
68
69        }
70
71      }
72    }
73  }
```

practicas/hilos/ejercicio/NReinas.java