

## UD2. Boletín 5. Hilos

1. Hacer un programa que calcule el factorial<sup>1</sup> y un número de la sucesión de Fibonacci<sup>2</sup>. Cada uno de los cálculos tiene que ser realizado por un hilo independiente. Las clases que implementen cada hilo tienen que llamarse Factorial y Fibonacci. Implementar los procesos de 2 maneras diferentes:

- a) Heredando de la clase Thread
- b) Implementando la interface Runnable

Los primeros términos de la serie de Fibonacci son:

$n =$	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	...
$x_n =$	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	...

Posible salida:

```
<terminated> Ej1Main [Java Application] G:\eclipse\plugins\org.  
El factorial de : 5 es 120  
El resultado de Fibonacci del número: 7 es 13
```

```
Factorial.java
1 package ejercicio01;
2
3 public class Factorial implements Runnable{
4
5     private int factorial = 1;
6     int numero;
7
8     public Factorial(int numero) {
9         super();
10        this.numero = numero;
11    }
12
13    public void run() {
14        int inicial = numero;
15        while(numero != 0) {
16            factorial = factorial * numero;
17            numero--;
18        }
19        System.out.println("El factorial de " + inicial +
20                           " es: " + factorial);
21    }
22 }
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

Fibonacci.java
1 package ejercicio01;
2
3 public class Fibonacci implements Runnable{
4
5     int numero, num1 = 0, num2 = 1, suma = 1;
6
7     public Fibonacci(int numero) {
8         super();
9         this.numero = numero;
10    }
11
12    public void run() {
13        for (int i = 1; i < numero; i++) {
14            suma = num1 + num2;
15            num1 = num2;
16            num2 = suma;
17        }
18        System.out.println("El resultado de Fibonacci del numero:"
19                           + " " + numero + " es: " + suma);
20    }
21 }
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100

HiloMainEjercicio01.java
1 package ejercicio01;
2
3 public class HiloMainEjercicio01 {
4
5     public static void main(String[] args) {
6
7         Factorial hilo1 = new Factorial(5);
8         Fibonacci hilo2 = new Fibonacci(7);
9
10        Thread hiloA = new Thread(hilo1);
11        Thread hiloB = new Thread(hilo2);
12
13        // hilo1.start();
14        // hilo2.start();
15
16        hiloA.start();
17        hiloB.start();
18    }
19 }
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

```
<terminated> HiloMainEjercicio01 (1) [Java Application] C:\Pro  
El factorial de 5 es: 120  
El resultado de Fibonacci del numero: 7 es: 13
```

- 1 Recuerda que factorial(0)=1 y para todo  $n > 0$ , factorial( $n$ )= $n$ \*factorial( $n-1$ ).
- 2 Recuerda que fib(0)=0, fib(1)=1 y para todo  $n > 1$ , fib( $n$ )=fib( $n-1$ )+fib( $n-2$ ).

2. Hacer un programa que calcule los factoriales del 5 al 14 modificando la clase Factorial del ejercicio anterior, para que se visualicen los mensajes:

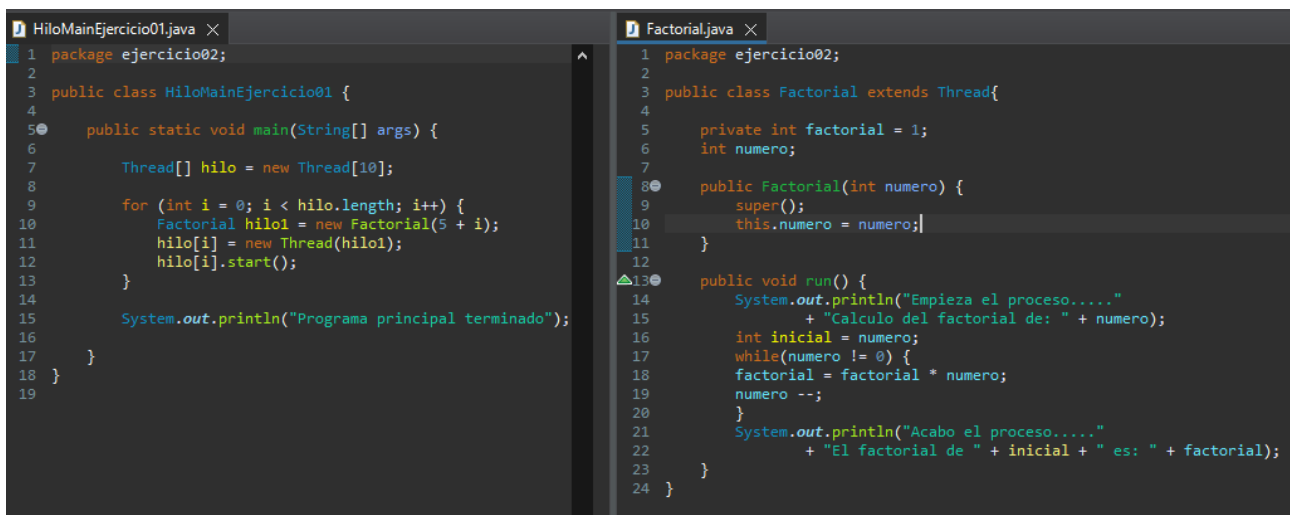
*Empieza el proceso ... cálculo del factorial de: XX*

*Acabó el proceso ..... el factorial de: XX es XXXX*

En el método main declarar un vector o array de 10 hilos y lanzarlos a ejecución.

Possible salida:

```
Empieza el proceso.....Calculo del factorial de: 5
Empieza el proceso.....Calculo del factorial de: 14
Empieza el proceso.....Calculo del factorial de: 13
Programa principal terminado
Empieza el proceso.....Calculo del factorial de: 12
Empieza el proceso.....Calculo del factorial de: 11
Empieza el proceso.....Calculo del factorial de: 10
Empieza el proceso.....Calculo del factorial de: 9
Empieza el proceso.....Calculo del factorial de: 8
Empieza el proceso.....Calculo del factorial de: 7
Empieza el proceso.....Calculo del factorial de: 6
Acabo el proceso.....El factorial de : 7 es: 5040
Acabo el proceso.....El factorial de : 8 es: 40320
Acabo el proceso.....El factorial de : 9 es: 362880
Acabo el proceso.....El factorial de : 10 es: 3628800
Acabo el proceso.....El factorial de : 11 es: 39916800
Acabo el proceso.....El factorial de : 12 es: 479001600
Acabo el proceso.....El factorial de : 13 es: 1932053504
Acabo el proceso.....El factorial de : 14 es: 1278945280
Acabo el proceso.....El factorial de : 5 es: 120
Acabo el proceso.....El factorial de : 6 es: 720
```



```
HiloMainEjercicio01.java
1 package ejercicio02;
2
3 public class HiloMainEjercicio01 {
4
5     public static void main(String[] args) {
6
7         Thread[] hilo = new Thread[10];
8
9         for (int i = 0; i < hilo.length; i++) {
10             Factorial hilo1 = new Factorial(5 + i);
11             hilo[i] = new Thread(hilo1);
12             hilo[i].start();
13         }
14
15         System.out.println("Programa principal terminado");
16
17     }
18 }
19

Factorial.java
1 package ejercicio02;
2
3 public class Factorial extends Thread{
4
5     private int factorial = 1;
6     int numero;
7
8     public Factorial(int numero) {
9         super();
10        this.numero = numero;
11    }
12
13    public void run() {
14        System.out.println("Empieza el proceso....."
15            + "Calculo del factorial de: " + numero);
16        int inicial = numero;
17        while(numero != 0) {
18            factorial = factorial * numero;
19            numero --;
20        }
21        System.out.println("Acabo el proceso....."
22            + "El factorial de " + inicial + " es: " + factorial);
23    }
24 }
```

```
<terminated> HiloMainEjercicio01 (1) [Java Application] C:\Prog
El factorial de 5 es: 120
El resultado de Fibonacci del numero: 7 es: 13
```

3. Escribe un programa que conste de las dos clases que se describen a continuación:  
Una de las clases, de nombre **ContarThread**, que se va a ejecutar en paralelo, tiene:

- un atributo entero de carácter privado de nombre `maxContar`
- un constructor que recibe un entero y lo asigna al atributo `maxContar`

Esta clase repite un número de veces igual a `maxContar` lo siguiente:

- muestra en que repetición va → "Repeticion: xx"
- muestra el nombre del Thread
- ejecuta el método `sleep` durante 2000 milisegundos

La clase **ContarApp** tiene el método `main`, que hará lo siguiente:

- crea un objeto de la clase `ContarThread` y ordénale que se ejecute
- Detén el método 3000 milisegundos.
- Crea otro objeto de la clase `ContarThread` y ordénale que se ejecute
- Mientras no hayan terminado los dos hilos anteriores muestra por pantalla "Sigo contando..." y ejecuta `sleep` pasándole 1000 como parámetro sucesivamente hasta que deje de cumplirse la condición.

Posible salida:

```
Repeticion: 1
Nombre del hilo: Thread-0
Repeticion: 2
Nombre del hilo: Thread-0
Repeticion: 1
Nombre del hilo: Thread-1
Sigo Contando....
Repeticion: 3
Nombre del hilo: Thread-0
Sigo Contando....
Repeticion: 2
Nombre del hilo: Thread-1
Sigo Contando....
Repeticion: 4
Nombre del hilo: Thread-0
Repeticion: 3
Nombre del hilo: Thread-1
Sigo Contando....
Sigo Contando....
Repeticion: 5
Nombre del hilo: Thread-0
Repeticion: 4
Nombre del hilo: Thread-1
Sigo Contando....
Sigo Contando....
Sigo Contando....
```

```
ContarThread.java X
1 package ejercicio03;
2
3 public class ContarThread extends Thread{
4
5     private int maxContar;
6
7     public ContarThread(int maxContar) {
8         super();
9         this.maxContar = maxContar;
10    }
11
12    public void run() {
13        for (int i = 0; i < maxContar; i++) {
14            System.out.println("Repeticion " + (i+1));
15            System.out.println("Nombre del hilo: " + getName());
16
17            try {
18                sleep(2000);
19            } catch (InterruptedException e) {
20                // TODO Auto-generated catch block
21                e.printStackTrace();
22            }
23        }
24    }
25
26 }
27

ContarApp.java X
1 package ejercicio03;
2
3 public class ContarApp {
4
5     public static void main(String[] args) {
6
7         ContarThread hilo = new ContarThread(5);
8         ContarThread hilo1 = new ContarThread(4);
9
10        hilo.start();
11        try {
12            hilo.sleep(3000);
13        } catch (InterruptedException e) {}
14        // TODO Auto-generated catch block
15        e.printStackTrace();
16
17        hilo1.start();
18
19        try {
20            while (hilo.isAlive() || hilo1.isAlive()) {
21                System.out.println("Sigo Contando...");
22                Thread.sleep(1000);
23            }
24        } catch (InterruptedException e) {
25            // TODO Auto-generated catch block
26            e.printStackTrace();
27        }
28    }
29 }
```

```
<terminated> ContarApp [Java Applica
Repeticion 1
Nombre del hilo: Thread-0
Repeticion 2
Nombre del hilo: Thread-0
Sigo Contando....
Repeticion 1
Nombre del hilo: Thread-1
Sigo Contando....
Repeticion 3
Nombre del hilo: Thread-0
Repeticion 2
Nombre del hilo: Thread-1
Sigo Contando....
Sigo Contando....
Repeticion 4
Nombre del hilo: Thread-0
Repeticion 3
Nombre del hilo: Thread-1
Sigo Contando....
Sigo Contando....
Repeticion 5
Nombre del hilo: Thread-0
Repeticion 4
Nombre del hilo: Thread-1
Sigo Contando....
Sigo Contando....
```

4. Escribe un programa que conste de las dos clases que se describen a continuación:  
La clase **ThreadBasico** (implementala con la interface Runnable), se va a ejecutar en paralelo, tiene
- dos atributos privados: frase, que almacena una cadena de caracteres; y aleatorio que es de tipo Random (Random es una clase que se encuentra en el paquete java.util.Random).
  - un constructor que recibe una frase y la almacena en el atributo anterior, también crea un elemento Random que se le asigna al atributo Random anterior.
  - Esta clase mostrará por pantalla el contenido de su frase, luego espera un tiempo aleatorio, **repitiendo estas instrucciones de manera infinita**. Para indicar el tiempo aleatorio usaremos esta fórmula:

$(\text{long}) (\text{Math.abs}(\text{aleatorio.nextInt()}) \% 1000)$  convierte a entero largo el valor absoluto del siguiente entero del número aleatorio obtenido antes.

Como será un número muy grande, solo se quiere el resto de dividirlo entre 1000.

La clase **ThreadBasicoMain** contiene el método principal. El método main crea y ejecuta dos objetos de ThreadBasico, pasándole al constructor de cada uno de ellos el parámetro de la frase.

Posible salida:

```
Buenos días
Hasta luego, buenas noches
Buenos días
Hasta luego, buenas noches
Hasta luego, buenas noches
Buenos días
Hasta luego, buenas noches
Hasta luego, buenas noches
Buenos días
Hasta luego, buenas noches
Buenos días
Hasta luego, buenas noches
Buenos días
Hasta luego, buenas noches
Hasta luego, buenas noches
Buenos días
Buenos días
Buenos días
Hasta luego, buenas noches
Buenos días
```

```
ThreadBasicoMain.java ×
1 package ejercicio04;
2
3 public class ThreadBasicoMain {
4
5     public static void main(String[] args) {
6         String frase = "Buenos dias";
7         String frase2 = "Hasta luego, buenas noches";
8
9         ThreadBasico hilo = new ThreadBasico(frase);
10        ThreadBasico hilo2 = new ThreadBasico(frase2);
11
12        Thread hiloA = new Thread(hilo);
13        Thread hiloB = new Thread(hilo2);
14
15        hiloA.start();
16        hiloB.start();
17    }
18 }
19
20
21

ThreadBasico.java ×
1 package ejercicio04;
2
3 import java.util.Random;
4
5 public class ThreadBasico implements Runnable {
6
7     private String frase;
8     private Random aleatorio;
9
10    public ThreadBasico(String frase) {
11        super();
12        this.frase = frase;
13        this.aleatorio = new Random();
14    }
15
16    @Override
17    public void run() {
18        long numero = (long) (Math.random()* 20 +1);
19        while (numero > 0) {
20            System.out.println(frase);
21            numero --;
22        }
23    }
24 }
```

```
<terminated> ThreadBasicoMain [Jav
Buenos dias
Buenos dias
Buenos dias
Buenos dias
Hasta luego, buenas noches
Hasta luego, buenas noches
Hasta luego, buenas noches
Hasta luego, buenas noches
Hasta luego, buenas noches
Buenos dias
Buenos dias
Buenos dias
Buenos dias|
Buenos dias
Buenos dias
Buenos dias
Buenos dias
Buenos dias
```

5. Explica, sin probarlo, que crees que hace el siguiente programa

```
1 package Boletin5;
2
3 import java.util.Random;
4
5 public class CHilo implements Runnable{
6     private String Caracter;
7     CHilo (String Carac ){
8         Caracter=Carac;
9     }
10
11     public void run() {
12         try {
13             Thread.sleep((long)(Math.abs(new Random().nextInt())%1000));
14             System.out.print(Caracter);
15         }catch (InterruptedException e) {}
16     }
17 }
18
19 class LetrasHilos{
20
21     LetrasHilos(String Frase){
22
23         Thread[] Hilo = new Thread[Frase.length()];
24
25         for (int i=0; i!=Frase.length();i++){
26             Hilo[i]= new Thread(new CHilo(Frase.substring(i,i+1)));
27
28             /*La sentencia de arriba es equivalente a estas:
29             CHilo objeto=CHilo(Frase.substring(i,i+1));
30             Thread Hi=new Thread(objeto);
31             Hilo[i]=Hi;
32             */
33             Hilo[i].start();
34         }
35     }
36 }
37
38 }
39 }
```

```
1 package Boletin5;
2
3 public class PruebaLetrasHilos {
4
5     public static void main(String[] args) throws InterruptedException{
6         LetrasHilos Instancia = new LetrasHilos("esternocleidomastoideo");
7
8     }
9
10 }
```

La clase CHilo implementa la clase Runnable, lo cual nos hará implementar su metodo de run(). Creamos un hilo que se dormira un tiempo aleatorio y pintara un caracter que se le pasara a traves de un constructor.

Por otro lado tenemos la clase LetrasHilos que tendra un String que contendra una frase. Se crear un array de hilos de misma longitud que tamaño tenga la frase introducida. A continuacion hacemos un bucle "for" para recorrer cada uno de los caracteres de la frase y asignarselo a cada hilo.

Por ultimo tenemos una clase main que ejecutara lo anterior. Su ejecucion sera una salida de los

caracteres contenidos en la frase escogida y iran apareciendo de forma desordenada y aleatoria debido a que el tiempo que se duerme cada proceso es diferente ya que es aleatorio. Por lo tanto si la frase sale ordenada simplemente sera mera coincidencia.

si modifiko el main de esta forma, ¿cuál es ahora el nuevo resultado?

```
1 package Boletin5;
2
3 public class PruebaLetrasHilos2 {
4
5     public static void main(String[] args) {
6         if (args.length!=1) {
7             System.out.println("Hay que introducir un argumento");
8             System.exit(0);
9         }
10        LetrasHilos Instancia = new LetrasHilos (args[0]);
11    }
12
13 }
```

Hay que introducir un argumento antes de ejecutar el programa. Si no se le pasa nada, el programa mostrara el mensaje de "Hay que introducir un argumento". Si se le pasa mostrara la palabra o frase como en el ejemplo anterior.