HILOS ESTÁTICOS, DINÁMICOS SINCRONIZADOS

MARIO JIMÉNEZ MARSET

ÍNDICE

1. ENUNCIADO – OBJETIVOS

En esta práctica se pedía crear, a partir del programa dado, cuatro variaciones del mismo según se llame al método: si este es estático y no sincronizado, estático y sincronizado, dinámico y no sincronizado o dinámico y sincronizado.

2. DESARROLLO

En primer lugar, se muestra el código de la variación del método estático no sincronizado. Se acompaña la explicación del mismo. Consta de tres clases:

```
package hilos1;

class sumArray{
   private static int sum;

   static int sumArray(int nums[]){
      sum=0;
      for (int i=0; i<nums.length;i++){
         sum+=nums[i];
         System.out.println("Total acumulado de "+Thread.currentThread().getName()+"
es "+sum);
      try {
         Thread.sleep(10);//permitir el cambio de tarea
      }catch (InterruptedException exc){
         System.out.println("Hilo interrumpido");
      }
      }
      return sum;
   }
}
```

Esta clase muestra el funcionamiento del sumatorio del array. El método se hace estático y se quita el synchronized. Esto implica que la variable local deba hacerse static para poder ser utilizada dentro del método estático.
Al hacer static el método, hacemos que este y lo que hay dentro de él sea compartido por los dos hilos que, posteriormente, serán ejecutados. Esto hará que la suma de cada hilo sea la misma.
Esta clase hace un for para recorrer el array pasado por parámetro y sumarle a la variable static la posición del array. Se imprime por pantalla el número cada vez que se ejecute un ciclo del for, con un pequeño Thread.sleep para esperar, con el objetivo de permitir el cambio de tarea.

```
class MiHilo implements Runnable{
    Thread hilo;
    //sumArray sumarray= new sumArray();
    int a[];
    int resp;

    //Construye un nuevo hilo.
    MiHilo(String nombre, int nums[]){
        hilo= new Thread(this,nombre);
        a=nums;
    }

    //Un método que crea e inicia un hilo
    public static MiHilo creaEInicia (String nombre,int nums[]){
        MiHilo miHilo=new MiHilo(nombre,nums);

        miHilo.hilo.start(); //Inicia el hilo
        return miHilo;
    }
    //Punto de entrada del hilo
    public void run(){
        int sum;
        System.out.println(hilo.getName()+ " iniciando.");

        resp=sumArray.sumArray(a);
        System.out.println("Suma para "+hilo.getName()+ " es "+resp);
        System.out.println(hilo.getName()+ " terminado.");
    }
}
```

Esta clase construye un hilo a través del constructor, pasando por parámetros un String nombre y un array de enteros, creando un hilo al que pasarle este nombre, además de un array igualado al que se acaba de pasar por parámetro.
Se crea un método que crea e inicia el hilo, pasándole por parámetros el nombre y el array anteriores. Se crea un objeto de la clase MiHilo y se llama al método start, retornando el método este hilo.
Se crea el método a ejecutar en la última clase: este consiste en imprimir el nombre del hilo en cuestión y llamar, en una variable int, a la clase sumArray y al método static establecido antes, pasando por parámetro el entero igualado en el constructor de la clase MiHilo. Finalmente, se imprime el nombre del hilo y si ha terminado.

```
public class Sincronizacion {
   public static void main(String[] args) {
      int a[]={1,2,3,4,5};
      MiHilo mh1 = MiHilo.creaEInicia("#1",a);
      MiHilo mh2 = MiHilo.creaEInicia("#2",a);

      try {
         mh1.hilo.join();
         mh2.hilo.join();
      }catch (InterruptedException exc){
         System.out.println("Hilo principal interrumpido.");
      }
   }
}
```

Sencillamente, esta es la clase que llama al método de la clase MiHilo. Se ha creado un array con valores aleatorios (para probar), y finalmente, dentro de un try catch, se llama al método join, el cual espera a que el Thread en cuestión muera.

Resultado:

```
#2 iniciando.
#1 iniciando.
Total acumulado de #1 es 1
Total acumulado de #2 es 1
Total acumulado de #1 es 5
Total acumulado de #2 es 5
Total acumulado de #1 es 8
Total acumulado de #2 es 11
Total acumulado de #2 es 19
Total acumulado de #1 es 19
Total acumulado de #2 es 24
Total acumulado de #1 es 29
Suma para #1 es 29
Suma para #2 es 29
#2 terminado.
#1 terminado.
```

Cada vez que se ejecute, saldrá un resultado diferente, puesto que, al no ser sincronizado, se llama a los dos hilos indistintamente (no se espera a que un hilo termine completamente su ejecución).

Las demás variaciones cuentan con el mismo código, exceptuando cuando hay que tocar si es static o no y synchronized o no.

Se muestra la segunda variación (estático y sincronizado):

```
package hilos2;

class sumArray{
    private static int sum;

    //sumArray está sincronizado
    static synchronized int sumArray(int nums[]){
        sum=0;
        for (int i=0; i<nums.length;i++){
            sum+=nums[i];
            System.out.println("Total acumulado de "+Thread.currentThread().getName()+"
es "+sum);
        try {
            Thread.sleep(10);//permitir el cambio de tarea
        }catch (InterruptedException exc){
            System.out.println("Hilo interrumpido");
        }
        }
        return sum;
    }
}
```

El único cambio respecto a la variación es la inclusión de synchronized: esto hace que, cada hilo haga su ejecución, independientemente del otro hilo.

```
class MiHilo implements Runnable{
    Thread hilo;
    //sumArray sumarray= new sumArray();
    int a[];
    int resp;

    //Construye un nuevo hilo.
    MiHilo(String nombre, int nums[]){
        hilo= new Thread(this,nombre);
        a=nums;
    }

    //Un método que crea e inicia un hilo
    public static MiHilo creaEInicia (String nombre,int nums[]){
        MiHilo miHilo=new MiHilo(nombre,nums);

        miHilo.hilo.start(); //Inicia el hilo
        return miHilo;
    }
    //Punto de entrada del hilo
    public void run(){
        int sum;
        System.out.println(hilo.getName()+ " iniciando.");

        resp=sumArray.sumArray(a);
```

```
        System.out.println("Suma para "+hilo.getName()+ " es "+resp);
        System.out.println(hilo.getName()+ " terminado.");
    }
}
```

Sin cambios con respecto a la primera variación.

```
public class Sincronizacion {
    public static void main(String[] args) {
        int a[]={1,2,3,4,5};
        MiHilo mh1 = MiHilo.creaEInicia("#1",a);
        MiHilo mh2 = MiHilo.creaEInicia("#2",a);

        try {
            mh1.hilo.join();
            mh2.hilo.join();
        }catch (InterruptedException exc){
            System.out.println("Hilo principal interrumpido.");
        }
    }
}
```

Sin cambios con respecto a la primera variación.

Resultado:

```
#1 iniciando.
#2 iniciando.
Total acumulado de #2 es 1
Total acumulado de #2 es 3
Total acumulado de #2 es 6
Total acumulado de #2 es 10
Total acumulado de #2 es 15
Total acumulado de #1 es 1
Suma para #2 es 15
#2 terminado.
Total acumulado de #1 es 3
Total acumulado de #1 es 6
Total acumulado de #1 es 10
Total acumulado de #1 es 15
Suma para #1 es 15
#1 terminado.
```

Sale el mismo resultado porque sigue siendo static el método sumArray.

Se muestra la tercera variación, en la cual se elimina el static y se pone el método no sincronizado.

```
package hilos3;

class sumArray{
   private int sum;

   int sumArray(int nums[]){
      sum=0;
      for (int i=0; i<nums.length;i++){
         sum+=nums[i];
         System.out.println("Total acumulado de "+Thread.currentThread().getName()+"
es "+sum);
      try {
         Thread.sleep(10);//permitir el cambio de tarea
      }catch (InterruptedException exc){
         System.out.println("Hilo interrumpido");
      }
      }
      return sum;
   }
}
```

Al ser no sincronizado, se ejecutan los dos hilos indistintamente; al ser dinámico y no static, no se comparten ni el método ni las variables, por lo que cada suma debería ser diferente. Sin embargo, al ser el mismo array el que se ejecute en la clase main, saldrá el mismo resultado.

```
class MiHilo implements Runnable{
    Thread hilo;
    sumArray sumarray= new sumArray();
    int a[];
    int resp;

    //Construye un nuevo hilo.
    MiHilo(String nombre, int nums[]){
        hilo= new Thread(this,nombre);
        a=nums;
    }

    //Un método que crea e inicia un hilo
    public static MiHilo creaEInicia (String nombre,int nums[]){
        MiHilo miHilo=new MiHilo(nombre,nums);

        miHilo.hilo.start(); //Inicia el hilo
        return miHilo;
    }
    //Punto de entrada del hilo
    public void run(){
        int sum;
        System.out.println(hilo.getName()+ " iniciando.");

        resp=sumarray.sumArray(a);
        System.out.println("Suma para "+hilo.getName()+ " es "+resp);
        System.out.println(hilo.getName()+ " terminado.");
    }
}
```

El único cambio con respecto a la primera variación es, que en el método run, en vez de llamar al método sumArray con el nombre de la clase (por ser método static) se llama con un objeto creado de la misma clase (el objeto se crea al principio de la clase MiHilo).

```
public class Sincronizacion {
    public static void main(String[] args) {
        int a[]={1,2,3,4,5};
        MiHilo mh1 = MiHilo.creaEInicia("#1",a);
        MiHilo mh2 = MiHilo.creaEInicia("#2",a);

        try {
            mh1.hilo.join();
            mh2.hilo.join();
        }catch (InterruptedException exc){
            System.out.println("Hilo principal interrumpido.");
        }
    }
}
```

Sin cambios respecto las demás variaciones.

Resultado:

```
#1 iniciando.
#2 iniciando.
Total acumulado de #1 es 1
Total acumulado de #2 es 1
Total acumulado de #1 es 3
Total acumulado de #2 es 3
Total acumulado de #1 es 6
Total acumulado de #2 es 6
Total acumulado de #1 es 10
Total acumulado de #2 es 10
Total acumulado de #1 es 15
Total acumulado de #2 es 15
Suma para #2 es 15
#2 terminado.
Suma para #1 es 15
#1 terminado.
```

Sale el mismo resultado por ser el mismo array el que ejecutan los dos hilos (nada que ver en este caso con static). No Synchronized, por lo que cada hilo se ejecuta indistintamente.

Por último, la cuarta variación consiste en llamar al método dinámico y sincronizado.

```
package hilos4;

class sumArray{
    private int sum;

    //sumArray está sincronizado
    synchronized int sumArray(int nums[]){
        sum=0;
        for (int i=0; i<nums.length;i++){
            sum+=nums[i];
            System.out.println("Total acumulado de "+Thread.currentThread().getName()+"
es "+sum);
        try {
            Thread.sleep(10);//permitir el cambio de tarea
        }catch (InterruptedException exc){
            System.out.println("Hilo interrumpido");
        }
        }
        return sum;
    }
}
```

El único cambio con respecto a la variación anterior es que el método es synchronized, lo cual hace que, primero se ejecute un hilo y, habiendo finalizado este, que se ejecute el siguiente.

```
class MiHilo implements Runnable{
    Thread hilo;
    static sumArray sumarray= new sumArray();
    int a[];
    int resp;

    //Construye un nuevo hilo.
    MiHilo(String nombre, int nums[]){
        hilo= new Thread(this,nombre);
        a=nums;
    }

    //Un método que crea e inicia un hilo
    public static MiHilo creaEInicia (String nombre,int nums[]){
        MiHilo miHilo=new MiHilo(nombre,nums);

        miHilo.hilo.start(); //Inicia el hilo
        return miHilo;
    }
    //Punto de entrada del hilo
    public void run(){
        int sum;
        System.out.println(hilo.getName()+ " iniciando.");

        resp=sumarray.sumArray(a);
        System.out.println("Suma para "+hilo.getName()+ " es "+resp);
        System.out.println(hilo.getName()+ " terminado.");
    }
}
```

Sin cambios con respecto a la tercera variación.

```
public class Sincronizacion {
    public static void main(String[] args) {
        int a[]={1,2,3,4,5};
        MiHilo mh1 = MiHilo.creaEInicia("#1",a);
        MiHilo mh2 = MiHilo.creaEInicia("#2",a);

        try {
            mh1.hilo.join();
            mh2.hilo.join();
        }catch (InterruptedException exc){
            System.out.println("Hilo principal interrumpido.");
        }
    }
}
```

Sin cambios con respecto a la tercera variación.

Resultado:

```
#2 iniciando.
#1 iniciando.
Total acumulado de #2 es 1
Total acumulado de #2 es 3
Total acumulado de #2 es 6
Total acumulado de #2 es 10
Total acumulado de #2 es 15
Total acumulado de #1 es 1
Suma para #2 es 15
#2 terminado.
Total acumulado de #1 es 3
Total acumulado de #1 es 6
Total acumulado de #1 es 10
Total acumulado de #1 es 15
Suma para #1 es 15
#1 terminado.
```

Al ser synchronized, se ejecuta primero un hilo y luego otro. Tienen el mismo resultado debido a que el array es el mismo.