

UD2. Boletín 7. Sincronización Básicos.

1. A continuación te muestro el código de un sencillo programa Java en el que cuatro objetos (hilos) trabajan con un objeto de datos compartido. Este objeto común imprime un mensaje de inicio y un mensaje de fin y para entre ambos haciendo una pequeña pausa. Realmente, el objeto debería iniciarse, realizar su tarea y terminar antes de que otro hilo inicie una nueva tarea, pero no lo hace ya que no existe sincronización alguna entre ellos, todos los hilos pelean por el procesador.

SIN SINCRONIZACION

```
package SincroBasicos;

public class mainSincronizada {
    public static void main(String args[]) {
        objetoComun compartido = new objetoComun();

        //Creo los 4 hilos que usarán el objeto compartido anterior
        sincronizada dibujo1 = new sincronizada("Caperucita", compartido);
        sincronizada dibujo2 = new sincronizada("Blancanieves",
        compartido); sincronizada dibujo3 = new sincronizada("Pinocho",
        compartido); sincronizada dibujo4 = new sincronizada("Pulgarcito",
        compartido);

        //Lanzo los hilos
        dibujo1.start();
        dibujo2.start();
        dibujo3.start();
        dibujo4.start();

        try {
            Thread.sleep(10000);
        } catch (InterruptedException ex) {
            System.out.printf("Interrupcion");
        }

        System.out.println("Hemos terminado amiguitos");
    }
}

package SincroBasicos;

class sincronizada extends Thread
{ objetoComun comparti;
  String nom;

  //Constructor
  sincronizada (String nombre, objetoComun compartido)
      { comparti=compartido;
        nom=nombre;
      }

  public void run() {
      comparti.Mostrar(nom);
  }
}
```

```

package SincroBasicos;

public class objetoComun {

    public void Mostrar (String mensaje) {
        System.out.println("¡En marcha...! " +
            mensaje);

        try {
            Thread.sleep(2000);

        } catch (InterruptedException ex) {
            System.out.printf("Interrupcion");
        }
    }
}

```

La salida obtenida:

```

run:
¡En marcha...! Pulgarcito
¡En marcha...! Pinocho
¡En marcha...! Caperucita
¡En marcha...! Blancanieves
¡He llegado! Pinocho
¡He llegado! Caperucita
¡He llegado! Blancanieves
¡He llegado! Pulgarcito
Hemos terminado amiguitos
BUILD SUCCESSFUL (total time: 1

```

lo correcto hubiera sido

```

run:
¡En marcha...! Caperucita
¡He llegado! Caperucita
¡En marcha...! Pulgarcito
¡He llegado! Pulgarcito
¡En marcha...! Pinocho
¡He llegado! Pinocho
¡En marcha...! Blancanieves
¡He llegado! Blancanieves
Hemos terminado amiguitos
BUILD SUCCESSFUL (total time: 1

```

es decir, el hilo se pone en marcha, duerme unos 2 segundos y llega a la meta.

CON SINCRONIZACION

La clave para la sincronización en Java es el concepto de *monitor*, que controla el acceso a un objeto. Un monitor funciona al implementar el concepto de *bloqueo*. Cuando un objeto está bloqueado por un subproceso, ningún subproceso puede tener acceso a dicho objeto. Todos los objetos de Java tienen un monitor. Esta característica se encuentra integrada en el propio lenguaje de Java. La sincronización es soportada por la palabra clave **synchronized**.

Ahora quiero que me soluciones el problema, es decir, tengo que bloquear el acceso al objeto común cuando haya un hilo que lo esté utilizando, y cuando ese hilo haya terminado con él lo liberará y podrá hacer uso otro hilo.

Hay dos formas de sincronizar la ejecución del código de hilos: la sincronización de bloques de código y los métodos sincronizados. Prueba a sincronizar el programa de ambas formas.

SINCRONIZAR UN OBJETO: utilizando la palabra reservada *synchronized* indica el objeto al que quieres restringirle el acceso, es decir, cuando se está ejecutando el hilo y se le indica que se mostrará el nombre.

SINCRONIZACION DE MÉTODOS: ahora sincronizarás el método al que acceden todos los hilos.

```

package ejercicio01;

public class MainSincronizada {

    public static void main(String args[]) {
        ObjetoComun compartido = new ObjetoComun();

        //Creo los 4 hilos que usarán el objeto compartido anterior
        Sincronizada dibujo1 = new Sincronizada("Caperucita", compartido);
        Sincronizada dibujo2 = new Sincronizada("Blancanieves", compartido);
        Sincronizada dibujo3 = new Sincronizada("Pinocho", compartido);
        Sincronizada dibujo4 = new Sincronizada("Pulgarcito", compartido);

        //Lanzo los hilos
        dibujo1.start();
        dibujo2.start();
        dibujo3.start();
        dibujo4.start();

        try {
            Thread.sleep(10000);
        } catch (InterruptedException ex) {
            System.out.printf("Interrupcion");
        }

        System.out.println("Hemos terminado amiguitos");
    }
}

```

```

package ejercicio01;

public class ObjetoComun {

    //La segunda forma se le añade la palabra synchronized al metodo
    public synchronized void Mostrar(String mensaje) {
        System.out.println("¡En marcha...! " + mensaje);

        try {
            Thread.sleep(2000);
            System.out.println("!He llegado¡ " + mensaje);

        } catch (InterruptedException ex) {
            System.out.printf("Interrupcion");
        }
    }
}

```

```

package ejercicio01;

public class Sincronizada extends Thread {
    private ObjetoComun comparti;
    private String nom;

    // Constructores
    public Sincronizada() {
    }

    public Sincronizada(String nombre, ObjetoComun compartido) {
        comparti = compartido;
        nom = nombre;
    }

    //La primera forma es añadir el syncriniced y meter la accion que se
    //quiera sincronizar dentro
    public void run() {
        synchronized (comparti) {
            comparti.Mostrar(nom);
        }
    }
}

```

```

<terminated> MainSincronizada [Java
¡En marcha...! Caperucita
!He llegado¡ Caperucita
¡En marcha...! Pulgarcito
!He llegado¡ Pulgarcito
¡En marcha...! Pinocho
!He llegado¡ Pinocho
¡En marcha...! Blancanieves
!He llegado¡ Blancanieves
Hemos terminado amiguitos

```

2. A partir del código siguiente y teniendo en cuenta la salida que obtengo tras su ejecución, haz los cambios que consideres necesarios para que el resultado de la ejecución de 2 threads en paralelo sea la secuencia de 1 a 7 sin repeticiones ni saltos de números,

```
package SincroBasicos;

public class MainEj2NoSincro {
    public static void main(String args[]) {
        Listado lista = new Listado();

        //Creo los objeto hilo
        Thread primero = new Ej2NoSincro();
        Thread segundo = new
        Ej2NoSincro();

        Ya termine.

        //Los lanzo
        primero.start();
        segundo.start();

        Esta es mi salida

        Esta es mi salida
        0 0 1 2 3 1 4 2 5 3 6 4 7 5 6 7

        try {
            Thread.sleep(4000);
        } catch (InterruptedException ex) {
            System.out.printf("Interrupcion");
        }
        System.out.println("\nYa termine.");
    }
}

package SincroBasicos;

public class Ej2NoSincro extends Thread
{ Listado salida;

    //constructor
    Ej2NoSincro(Listado sali) {
        salida=sali;

    public void run() {
        salida.Mostrar();
    }
}

package SincroBasicos;

public class Listado {
    public void Mostrar() {
        System.out.println("\nEsta es mi salida ");
        for (int i=0; i<8; i++) {
            System.out.print(i+" ");
        }
    }
}
```

es decir, utilizando la sincronización quiero obtener algo similar a:

run:

```
Esta es mi salida
0 1 2 3 4 5 6 7
Esta es mi salida
0 1 2 3 4 5 6 7
Ya termine.
```

RECUERDA:

- Si es invocado un método **synchronized** únicamente el Thread que lo invoca tiene acceso a la instancia del Objeto, y cualquier otro Thread que intente acceder a esta instancia tendrá que esperar hasta que sea terminada la ejecución del método synchronized.
- Se sincronizan Objetos o métodos.

SOLUCION:

```
package ejercicio02;

public class MainEj2NoSincro {
    public static void main(String args[]) {
        Listado lista = new Listado();

        // Creo los objeto hilo
        Thread primero = new Ej2NoSincro(lista);
        Thread segundo = new Ej2NoSincro(lista);

        // Los lanzo
        primero.start();
        segundo.start();

        try {
            Thread.sleep(4000);
        } catch (InterruptedException ex) {
            System.out.printf("Interrupcion");
        }
        System.out.println("\nYa termine.");
    }
}

package ejercicio02;

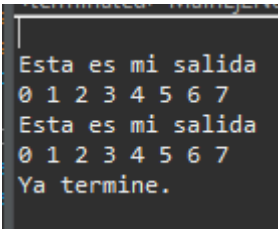
public class Listado {
    public synchronized void Mostrar() {
        System.out.println("\nEsta es mi salida ");
        for (int i = 0; i < 8; i++) {
            System.out.print(i + " ");
        }
    }
}
```

```
package ejercicio02;

public class Ej2NoSincro extends Thread {
    Listado salida;

    //constructor
    Ej2NoSincro(Listado sali) {
        salida = sali;
    }

    public void run() {
        salida.Mostrar();
    }
}
```



```
terminated - Main.java
Esta es mi salida
0 1 2 3 4 5 6 7
Esta es mi salida
0 1 2 3 4 5 6 7
Ya termine.
```