Sincronización en C# y Java.

Ejemplos

Sistemas Distribuidos
Grado en Ingeniería Informática

ETSI ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN

Universidad de Huelva

Ejemplo 1. Ejemplos con hilos en C#.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;
using System.Runtime.CompilerServices;

namespace ejemploHilos
{
    class Program
    {
        static void Main(string[] args)
        {
            recurso r = new recurso();
            sumador osumador = new sumador(r);
            Console.WriteLine("El recurso vale {0}", r.Contador);
            Thread h1 = new Thread(new ThreadStart(osumador.incrementa));
            Thread h2 = new Thread(new ThreadStart(osumador.incrementa));
            h1.Start();
            h2.Start();
            h1.Join();
            h2.Join();
            Console.WriteLine("El recurso vale {0}", r.Contador);
            Console.ReadLine();

        }
    }
```

```
class sumador
{
    recurso r;
    public sumador(recurso r)
    {
        this.r = r;
    }

    public void incrementa()
    {
        for (int i = 0; i < 10000; i++)
            r.Suma(1);
    }
}

class recurso
{
    private int contador = 0;
```

```
//[MethodImpl(MethodImplOptions.Synchronized)]
        public void Suma(int n)
        {
            lock (this)             {
                contador = contador + n;
            }
            /*
            Monitor.Enter(this);
            contador = contador + n;
            Monitor.Exit(this);
            */
        }
        //Ejemplo de propiedad de solo lectura
        public int Contador
        {
            get
            {
                return contador;
            }
        }
    }
}
```

Ejemplo 2. Consumidor y productor con buffer en Java utilizando monitores

Fichero Buffer.java

```
package ProductorConsumidor;

public class Buffer {
    private int[] Datos;
    private int Tamanio, Cabeza, Cola, NElementos;
    public Buffer(int Tamanio) {
          Datos=new int[Tamanio];
          this.Tamanio=Tamanio;
          Cabeza=Cola=NElementos=0;
    }

    public synchronized void Poner(int Dato) {
        while (NElementos==Tamanio){
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("Buffer.Poner ha sido interrumpida");
            }
        }
        Datos[Cola]=Dato;
        Cola=(Cola+1)%Tamanio;
        NElementos++;
        System.out.println ("Buffer. Numeros de elementos "+NElementos);
        notify();
    }
```

```
    public synchronized int Sacar() {
        while (NElementos==0) {
            try {
                wait();
            } catch (InterruptedException e) {
                System.out.println("Buffer.Sacar ha sido interrumpida");
            }
        }
        int Elemento=Datos[Cabeza];
        Cabeza=(Cabeza+1)%Tamanio;
        NElementos--;
        System.out.println ("Bbuffer. Numeros de elementos "+NElementos);
        notify();
        return Elemento;
    }
}
```

Fichero Consumidor.java

```
package ProductorConsumidor;

import java.util.Random;

public class Consumidor extends  Thread{
       private int idConsumidor;
    private int nveces;
    static boolean Comenzar=false;
    private Buffer buf;
    private int Pausa;
```

```
public Consumidor (int id, int veces, Buffer buf,int Pausa) {
    idConsumidor=id;
    nveces=veces;
    this.buf=buf;
    this.Pausa=Pausa;
}

public void run()
{
    try {
       Random v=new Random(idConsumidor);
       int valor;

       while (Comenzar==false)
       {
            System.out.println("Soy el Consumidor ("+idConsumidor+") y estoy esperando");
            Thread.sleep(v.nextInt(Pausa));
       }

       for (int i=0; i<nveces; i++)
       {
          valor=buf.Sacar();
          System.out.println("Soy el Consumidor ("+idConsumidor+") y Leido el valor "+valor);

          Thread.sleep(v.nextInt(Pausa));
       }
       System.out.println("Soy el Consumidor ("+idConsumidor+") y TERMINO");
       Thread.sleep(2000);
    }
```

```
            catch (InterruptedException ex)
            {
                System.out.println("Error en la ejecución de la hebra Consumidor"+idConsumidor);
            }
        }

}
```

Fichero Productor.java

```
package ProductorConsumidor;

import java.util.Random;

public class Productor extends Thread{
    private int idProductor;
    private int nveces;
    static boolean Comenzar=false;
    private Buffer buf;
    private int Pausa;

    public Productor (int id, int veces, Buffer buf,int Pausa) {
        idProductor=id;
        nveces=veces;
        this.buf=buf;
        this.Pausa=Pausa;
    }
```

```
    public void run() {
        try {
           Random v=new Random(idProductor);
           int valor;
           while (Comenzar==false)
           {
                System.out.println("Soy el Productor ("+idProductor+") y estoy esperando");
                Thread.sleep(v.nextInt(Pausa));
           }

           for (int i=0; i<nveces; i++)
           {
              valor=v.nextInt(100);
              System.out.println("Soy el Productor ("+idProductor+") y Escribo el valor "+valor);
              buf.Poner((valor));
              Thread.sleep(v.nextInt(Pausa));
           }
           System.out.println("Soy el Productor ("+idProductor+") y TERMINO");
           Thread.sleep(2000);
        }
        catch (InterruptedException ex)
         {
             System.out.println("Error en la ejecución de la hebra Productor"+idProductor);
         }
    }

}
```

Fichero Productor.java

```
package ProductorConsumidor;
import java.io.*;

public class ProductorConsumidor {
    @SuppressWarnings("empty-statement")
     public static void main(String[] args) {

        InputStreamReader Entrada=new InputStreamReader(System.in);
        BufferedReader Teclado=new BufferedReader(Entrada);
        int TamaBuffer=0;
        int NHebrasPC=0;
        int NVeces=0;
        int PausaProductor=0,PausaConsumidor=0;

        try {
            System.out.print("Introduce el tamaño del buffer: ");
            TamaBuffer= Integer.parseInt(Teclado.readLine());
            System.out.print("Introduce el nº de Hebras por Productor y Consumidor: ");
            NHebrasPC= Integer.parseInt(Teclado.readLine());
            System.out.print("Introduce el nº de veces a consumir o a producir: ");
            NVeces= Integer.parseInt(Teclado.readLine());
            System.out.print("Introduce la pausa en ms de los productores: ");
            PausaProductor= Integer.parseInt(Teclado.readLine());
            System.out.print("Introduce la pausa en ms de los consumidores: ");
            PausaConsumidor= Integer.parseInt(Teclado.readLine());
       } catch (IOException ex) {
           System.out.println("Error en la entrada por teclado....");
       }
```

```
        Buffer buf=new Buffer(TamaBuffer);
        for (int i=0; i<NHebrasPC; i++)
        {
            Productor p=new Productor(i,NVeces,buf,PausaProductor);
            p.start();
        }
        for (int i=0; i<NHebrasPC; i++)
        {
            Consumidor c=new Consumidor(i,NVeces,buf,PausaConsumidor);
            c.start();
        }

        Consumidor.Comenzar=true;
        Productor.Comenzar=true;

    }
}
```

Ejemplo 3. Consumidor y productor con buffer en c# utilizando monitores

//El código comentado y en color verde indican una mejor manera de utilizar los hilos ya que al mantener las referencias en vectores
//podríamos determinar en todo momento las hebras que están activas o no.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading;

namespace ejemploHilos
{
    class Buffer
    {
        int[] Datos;
        int Tamanio, Cabeza, Cola, NElementos;
        public Buffer(int Tamanio) {
            Datos = new int[Tamanio];
            this.Tamanio = Tamanio;
            Cabeza = Cola = NElementos = 0;
        }

        public void Poner(int Dato)  {
            Monitor.Enter(this);
            try {
                while (NElementos == Tamanio) {
                    Monitor.Wait(this);
                }
                Datos[Cola] = Dato;
                Cola = (Cola + 1) % Tamanio;
```

```
            NElementos++;
            Console.WriteLine("Numero de elementos en el Buffer" + NElementos);
            Monitor.Pulse(this);
        }
        Finally {
            Monitor.Exit(this);
        }
    }

    public int Sacar() {
        int Elemento = -1;
        Monitor.Enter(this);
        Try   {
            while (NElementos == 0) {
                Monitor.Wait(this);
            }
            Elemento = Datos[Cabeza];
            Cabeza = (Cabeza + 1) % Tamanio;
            NElementos--;
            Console.WriteLine("Numero de elementos en el Buffer" + NElementos);
            Monitor.Pulse(this);
        }
        Finally {
            Monitor.Exit(this);
        }
        return Elemento;
    }
}
```

```
class Consumidor
{
    int idConsumidor;
    int nveces;
    Buffer buf;
    int Pausa;
    public static bool Comenzar = false;

    public Consumidor(int id, int veces, Buffer buf, int Pausa)    {
        idConsumidor = id;
        nveces = veces;
        this.buf = buf;
        this.Pausa = Pausa;

    }

    public void Consume() {
        try  {
            Random v = new Random(idConsumidor);
            int valor;
            while (Comenzar == false)  {
                Console.WriteLine("Soy el Consumidor (" + idConsumidor + ") y estoy esperando");
                Thread.Sleep(v.Next(Pausa));
            }

            for (int i = 0; i < nveces; i++) {
                valor = buf.Sacar();
                Console.WriteLine("Soy el Consumidor (" + idConsumidor + ") y Leido el valor " +
                                  valor);
                Thread.Sleep(v.Next(Pausa));
            }
```

```
            Console.WriteLine("Soy el Consumidor (" + idConsumidor + ") y TERMINO");
            Thread.Sleep(v.Next(Pausa));
        }
        catch (ThreadAbortException ex) {
            Console.WriteLine("Error en la ejecución de la hebra Consumidor" + idConsumidor);
        }
    }
}

class Productor
{
    int idProductor;
    int nveces;
    Buffer buf;
    int Pausa;
    public static bool Comenzar = false;

    public Productor(int id, int veces, Buffer buf, int Pausa) {
        idProductor = id;
        nveces = veces;
        this.buf = buf;
        this.Pausa = Pausa;
    }

    public void Produce() {
        try  {
            Random v = new Random(idProductor);
            int valor;
```

```
            while (Comenzar == false) {
                Console.WriteLine("Soy el Productor ("+idProductor+ ") y estoy esperando");
                Thread.Sleep(v.Next(Pausa));
            }

            for (int i = 0; i < nveces; i++) {
               valor=v.Next(100);
               Console.WriteLine("Soy el Productor ("+idProductor+") y Escribo el valor "+valor);
               buf.Poner(valor);
               Thread.Sleep(v.Next(Pausa));
            }
            Console.WriteLine("Soy el Productor ("+idProductor+ ") y TERMINO");
            Thread.Sleep(v.Next(Pausa));
        }
        catch (ThreadAbortException ex)  {
            Console.WriteLine("Error en la ejecución de la hebra Consumidor" + idProductor);
        }
    }
}


class Program
{
    static void Main(string[] args) {

        int TamaBuffer = 0;
        int NHebrasPC = 0;
        int NVeces = 0;

        int PausaProductor = 0, PausaConsumidor = 0;
        Console.Write("Introduce el tamaño del buffer: ");
```

```
int.TryParse(Console.ReadLine(),out TamaBuffer);
Console.Write("Introduce  el nº de Hebras por Productor y Consumidor: ");
int.TryParse(Console.ReadLine(),out NHebrasPC);
Console.Write("Introduce el nº de veces a consumir o a producir: ");
int.TryParse(Console.ReadLine(), out NVeces);
Console.Write("Introduce la pausa en ms de los productores: ");
int.TryParse(Console.ReadLine(), out PausaProductor);
Console.Write("Introduce la pausa en ms de los consumidores: ");
int.TryParse(Console.ReadLine(), out PausaConsumidor);

Buffer buf=new Buffer(TamaBuffer);
/*
Thread[] HilosProductores = new Thread[NHebrasPC];
Thread[] HilosConsumidores = new Thread[NHebrasPC];
Productor[] Productores = new Productor[NHebrasPC];
Consumidor[] Consumidores = new Consumidor[NHebrasPC];
*/

for (int i=0; i<NHebrasPC; i++)  {
    /*
    Productores[i]=new Productor(i,NVeces,buf,PausaProductor);
    HilosProductores[i]=new Thread(new ThreadStart(Productores[i].Produce));
    HilosProductores[i].Start();
    */
    Productor p= new Productor(i, NVeces, buf, PausaProductor);
    Thread th = new Thread(new ThreadStart(p.Produce));
    th.Start();
};
```

```
            for (int i = 0; i < NHebrasPC; i++)   {
                /*
                Consumidores[i] = new Consumidor(i, NVeces, buf, PausaConsumidor);
                HilosConsumidores[i] = new Thread(new ThreadStart(Consumidores[i].Consume));
                HilosConsumidores[i].Start();
                */
                Consumidor c = new Consumidor(i, NVeces, buf, PausaConsumidor);
                Thread th = new Thread(new ThreadStart(c.Consume));
                th.Start();
            };

            Consumidor.Comenzar = true;
            Productor.Comenzar=true;
            /*
            for (int i = 0; i < NHebrasPC; i++) {
                HilosProductores[i].Join();
                HilosConsumidores[i].Join();
            }
            Thread.Sleep(3000);*/

        }
    }
}
```