Seminario 6. Programación Concurrente.

Este documento deberá llevarse impreso la sexta clase de seminario.

Ejercicio 1

Analice el siguiente código entendiendo su funcionamiento. Aunque compila sin ningún error, trate de encontrar algún posible error de ejecución (el código está disponible para su descarga en el campus virtual, en la sección de *Seminario 6*). Si hubiere algún error, corríjalo.

```
using System.Threading;
namespace TPP.Seminarios.Concurrente.Seminario6 {
    /// <summary>
    /// Representa un recurso a adquirir por un hilo
    /// </summary>
    public class Recurso {

        public string Nombre { get; private set; }

        public Recurso(string nombre) {
            this.Nombre = nombre;
        }

        public void Procesar() {
            Thread.Sleep(100); // simula procesamiento...
        }
    }
}


namespace TPP.Seminarios.Concurrente.Seminario6 {
    /// <summary>
    /// Encapsula un hilo
    /// </summary>
    public class Hilo {

        public Recurso Recurso1 { get; private set; }
        public Recurso Recurso2 { get; private set; }

        public Hilo(Recurso recurso1, Recurso recurso2) {
            this.Recurso1 = recurso1;
            this.Recurso2 = recurso2;
        }

        /// <summary>
        /// Este método será llamado concurrentemente en la ejecución del hilo
        /// </summary>
        public void Ejecutar() {
            lock (this.Recurso1) {
                Recurso1.Procesar();
                lock (this.Recurso2) {
                    Recurso2.Procesar();
                }
            }
        }
    }
}
```

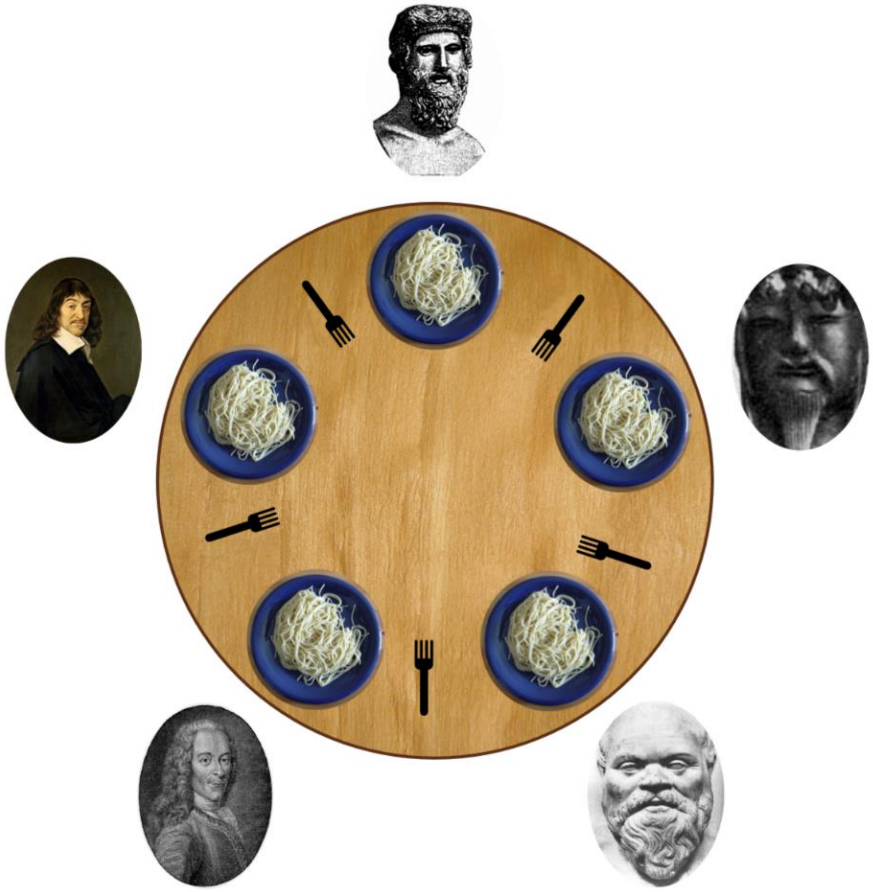
```
using System.Threading;

namespace TPP.Seminarios.Concurrente.Seminario6 {
    class Program {
        public static void Main() {
            Recurso recurso1 = new Recurso("Recurso 1"),
                    recurso2 = new Recurso("Recurso 2");
            Hilo hilo1 = new Hilo(recurso1, recurso2),
                 hilo2 = new Hilo(recurso2, recurso1);
            new Thread(hilo1.Ejecutar).Start();
            new Thread(hilo2.Ejecutar).Start();
        }
    }
}
```

Ejercicio 2. Problema de los Filósofos Cenando.

Es un problema propuesto por Dijkstra en 1965 para representar un problema de sincronización en aplicaciones concurrentes. El problema dice así:

Cinco filósofos se sientan alrededor de una mesa y pasan su vida cenando y pensando. Cada filósofo tiene un plato de espagueti y un tenedor a la izquierda de su plato. Para comer los fideos son necesarios dos tenedores y cada filósofo sólo puede tomar los que están a su izquierda y derecha. Todos los filósofos deberían comer y pensar cada cierto tiempo (no sería válido que sólo realizasen una acción).



Una posible implementación es la siguiente (código disponible para su descarga en el campus virtual). Analice el código y, si hubiere algún error, corríjalo.

```
namespace TPP.Seminarios.Concurrente.Seminario6 {
    /// <summary>
    /// Cada uno de los tenedores disponibles
    /// </summary>
    class Tenedor {
        /// <summary>
        /// Número de tenedor
        /// </summary>
        private int numero;

        public Tenedor(int numero) {
            this.numero = numero;
        }
    }
}
```

```
using System;
using System.Threading;

namespace TPP.Seminarios.Concurrente.Seminario6 {

    class Filosofo {
        /// <summary>
        /// ID del filósofo
        /// </summary>
        private int numeroFilosofo;

        /// <summary>
        /// El tiempo que tarda pensando
        /// </summary>
        private int milisPensar;

        /// <summary>
        /// El tiempo que tarda comiendo
        /// </summary>
        private int milisComer;

        /// <summary>
        /// Los tenedores izquierdos y derechos
        /// </summary>
        private Tenedor tenedorIzquierdo, tenedorDerecho;

        public Filosofo(int numeroFilosofo, int milisPensar, int milisComer,
                        Tenedor tenedorIzquierdo, Tenedor tenedorDerecho) {
            this.numeroFilosofo = numeroFilosofo;
            this.milisPensar = milisPensar; this.milisComer = milisComer;
            this.tenedorIzquierdo = tenedorIzquierdo;
            this.tenedorDerecho = tenedorDerecho;

            new Thread(new ThreadStart(ComerYPensar)).Start();
        }
```

```
        private void ComerYPensar() {
            for (;;) {
                lock (this.tenedorIzquierdo) {
                    lock (this.tenedorDerecho) {
                        Console.WriteLine("El filósofo " + numeroFilosofo +
                                          " está comiendo...");
                        Thread.Sleep(milisComer);
                    }
                }
                Console.WriteLine("El filósofo " + numeroFilosofo +
                                  " está pensando...");
                Thread.Sleep(milisPensar);
            }
        }
    }
}


namespace TPP.Seminarios.Concurrente.Seminario6 {
    class Program {
        static void Main(string[] args) {
            const int milisPensar = 0, milisComer = 0;
            Tenedor[] tenedor = new Tenedor[5];
            for (int i = 0; i < tenedor.Length; i++)
                tenedor[i] = new Tenedor(i);

            new Filosofo(0, milisPensar, milisComer, tenedor[0], tenedor[1]);
            new Filosofo(1, milisPensar, milisComer, tenedor[1], tenedor[2]);
            new Filosofo(2, milisPensar, milisComer, tenedor[2], tenedor[3]);
            new Filosofo(3, milisPensar, milisComer, tenedor[3], tenedor[4]);
            new Filosofo(4, milisPensar, milisComer, tenedor[4], tenedor[0]);
        }
    }
}
```