(3 puntos) 7. A continuación mostramos una modificación de la especificación formal de un recurso gestor de *lectores/escritores* para evitar el riesgo de inanición de escritores. Se ha dividido la operación *inicioEscribir* en dos: una primera que declara la intención de escribir por parte de un proceso escritor (*intencionEscribir*) y una segunda que realmente solicita el acceso (*permisoEscribir*). La primera incrementa el contador de *escritores en espera*, de modo que si este contador es distinto de 0, no dejamos que entren más lectores.

C-TAD GestorLE_2

OPERACIONES
ACCIÓN intencionEscribir:
ACCIÓN permisoEscribir:
ACCIÓN finEscribir:
ACCIÓN inicioLeer:
ACCIÓN finLeer:

SEMÁNTICA
DOMINIO:
TIPO: *GestorLE_2* = (*leyendo* : $\mathbb{N}$ × *escribiendo* : $\mathbb{N}$ × *esc_esperando* : $\mathbb{N}$)
INICIAL: self $= (0, 0, 0)$
INVARIANTE: self.*leyendo* · self.*escribiendo* $= 0 \wedge$ self.*escribiendo* ≤ 1

CPRE: Cierto
intencionEscribir()
POST: self$^{pre} = (l, e, w) \wedge$ self $= (l, e, w + 1)$

CPRE: self.*leyendo* $= 0 \wedge$ self.*escribiendo* $= 0$
permisoEscribir()
POST: self$^{pre} = (l, e, w) \wedge$ self $= (0, e + 1, w - 1)$

CPRE: Cierto
finEscribir()
POST: self$^{pre} = (l, e, w) \wedge$ self $= (0, e - 1, w)$

CPRE: self.*escribiendo* $= 0 \wedge$ self.*esc_esperando* $= 0$
inicioLeer()
POST: self$^{pre} = (l, e, w) \wedge$ self $= (l + 1, 0, 0)$

CPRE: Cierto
finLeer()
POST: self$^{pre} = (l, e, w) \wedge$ self $= (l - 1, 0, w)$

Se pide: Completar la implementación de este recurso mediante monitores que aparece en la página siguiente. En cuanto al código de desbloqueos podéis optar tanto por un método de desbloqueo genérico como por tener código de desbloqueo especializado en los distintos métodos. Si optáis por la segunda posibilidad dejad en blanco el cuerpo del método `desbloqueoSimple`.

Apellidos: **Nombre:** **Matrícula:**

```
class GestorLE2Mon implements GestorLE2 {
  // estado del recurso
```

```
  // Declaramos monitores y colas condition
```

```
  public GestorLE2Mon() {
```

```
  }

  public void intencionEscribir() {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

```
    // codigo de desbloqueo y salida de la seccion critica
```

```
  }

  public void permisoEscribir() {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

```
    // codigo de desbloqueo y salida de la seccion critica
```

```
  }

 public void finEscribir() {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

```
    // codigo de desbloqueo y salida de la seccion critica
```

```
    }

  public void inicioLeer() {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

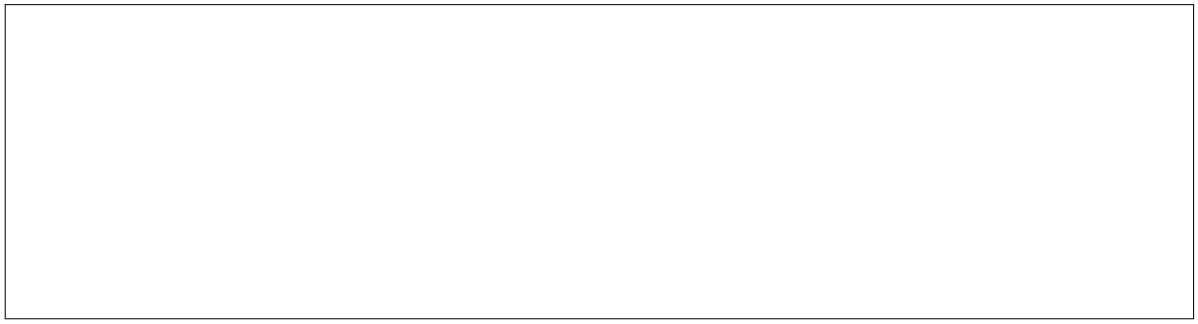
```
    // codigo de desbloqueo y salida de la seccion critica
```

```
  }

  public void finLeer() {
    // acceso a la seccion critica y codigo de bloqueo
```

```
    // codigo de la operacion
```

```
    // codigo de desbloqueo y salida de la seccion critica
```

```
  }

  private void desbloqueoSimple() {
```



```
  }
}
```

```
import es.upm.babel.cclib.Monitor;

// C-TAD GestorLE2
public class GestorLE2Mon
{
    // OPERACIONES
    // ACCION IntencionEscribir:
    // ACCION PermisoEscribir:
    // ACCION FinEscribir:
    // ACCION InicioLeer:
    // ACCION FinLeer:

    // SEMANTICA
    // DOMINIO:
    // TIPO: GestorL2E = (NLect:N x NEsc:N x NEscEsp:N)
    private int nLect;
    private int nEsc;
    private int nEscEsp;
    // todos a 0 inicialmente

    // TODO: declaracion de atributos extra necesarios
    private Monitor mutex = new Monitor();
    private Monitor.Cond esperandoEsc = mutex.newCond();
    private Monitor.Cond esperandoLeer = mutex.newCond();

    // INICIAL:
    //  -self.Esc /\ self.NLect = 0
    public Gestor_LE()
    {
        // no queda nada por inicializar en Java
    }

    // CPRE: true
    // IntencionEscribir()
    // POST: self_pre == (l,e,w) && self = (l,e,w+1)
    public void intencionEscribir() {
        mutex.enter();
        // CPRE --
        // codigo de la operacion
        this.nEscEsp++;

        // Codigo de desbloqueos
        // sabemos que la CPRE de iniciar_lectura es falsa
        // sabemos que la CPRE de iniciar_escritura es falsa
        // desbloquear();
        mutex.leave();
    }

    // CPRE: nEsc == 0 && nLec == 0
    // permisoEscribir()
    // POST:self_pre == (0,0,w) && self == (0,1,w-1)
    public void permisoEscribir() {
        mutex.enter();
        // comprobamos CPRE
        if (nEsc > 0 || nLec > 0) {
            esperandoEsc.await();
        }
        // codigo que establece la POST
        this.nEsc = 1;
        this.nEscEsp--;
        // desbloqueos: no ha lugar
        // sabemos que no se cumplen ni la CPRE de inicioLeer ni
        // la CPRE de permisoEscribir
        mutex.leave();
```

```
}

// CPRE: --
// finEscribir()
// POST: self_pre == (0,1,w) && self == (0,0,w)
public void finEscribir() {
    mutex.enter();
    // CPRE: --
    // codigo que establece la POST
    this.nEsc == 0;
    // codigo de desbloqueos
    // INCORRECTO
    // if(esperandoEsc.waiting() > 0)
    //      esperandoEsc.signal();
    // if(esperandoLeer.waiting() > 0)
    //      esperandoLeer.signal();

    // NAIVE (CON PRIORIDAD A ESCRITORES)
    // if(esperandoEsc.waiting() > 0)
    //     esperandoEsc.signal();
    // else
    //     if(esperandoLeer.waiting() > 0)
    //      esperandoLeer.signal();

    // DANDO PRIORIDADES JUSTAS
    // if(esperandoEsc.waiting() > 0)
    //      if(esperandoLeer.waiting() > 0 &&
    //         esperandoLeer.waiting() > esperandoEsc.waiting())
    //              esperandoLeer.signal();
    //      else
    //              esperandoEsc.signal();
    // else
    //      if(esperandoLeer.waiting() > 0)
    //              esperandoLeer.signal();


    // sabemos que !esc, luego se cumple la CPRE de iniciar_lectura
    // tambien sabemos que nLect == 0, luego tambien
    // se cumple la CPRE de iniciar_escritura
    // codigo de desbloqueos
    // desbloquear();
    if (nEscEsp == 0 && esperandoLeer.waiting() > 0) {
        esperandoLeer.signal();
    } else {
        esperandoEscribir.signal();
    }
    mutex.leave();
}

// CPRE: nEsc == 0 && nEscEsp == 0
// Iniciar_Lectura()
// POST:self=self_pre \self.NLect = self_pre.NLect + 1
public void iniciar_lectura() {
    mutex.enter();
    // comprobacion CPRE
    if(this.nEsc > 0 || this.nEscEsp > 0)
        esperandoLeer.await();
    // asumimos CPRE
    this.nLect++;
    // Codigo de desbloqueos
    // desbloquear();
    // sabemos que NO se cumple la CPRE de iniciar_escritura
    // sabemos que SI se cumple la CPRE de iniciar_lectura
    esperandoLeer.signal();
```

```
        mutex.leave();
    }

    // CPRE: cierto
    // finLeer()
    // POST:self_pre == (l,0,w) && self == (l-1,0,w)
    public void finLeer() {
        mutex.enter();
        // CPRE es Cierto, luego no hay bloqueo
        // codigo de la operacion
        this.nLect--;

        // if(this.nLect == 0 && esperandoEsc.waiting() > 0)
        //      esperandoEsc.signal();

        //desbloquear();
        // sabemos que se cumple la CPRE de iniciar_lectura
        // os prometo que no hay procesos esperando en
        // esperandoLeer !!!!!OJO!!!!!!!
        // esperandoLeer.waiting() == 0
        // a veces se cumple la CPRE de iniciar_escritura
        if (nLec == 0) {
            esperandoEsc.signal();
        }
        mutex.leave();
    }



    // DESACTUALIZADO
    private void desbloquear() {
        boolean senalizado = false;
        if (this.nLect == 0 && !this.esc) {
            if (esperandoEsc.waiting() > 0 && esperandoLeer.waiting() > 0 && !senalizado) {
                if(esperandoLeer.waiting() > esperandoEsc.waiting())
                    esperandoLeer.signal();
                else
                    esperandoEsc.signal();
                senalizado = true;
            }
            if (esperandoEsc.waiting() > 0 && !senalizado) {
                esperandoEsc.signal();
                senalizado = true;
            }
            if (esperandoLeer.waiting() > 0 && !senalizado) {
                esperandoLeer.signal();
                senalizado = true;
            }
        }
    }
}
```