**Apellidos:** **Nombre:** **Matrícula:**

**Respuestas**

| 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|
| | | | | | |

# Concurrencia (parte 2)/clave: b

**Curso 2017–2018 - 2º semestre (junio 2018)**
**Grado en Ingeniería Informática / Grado en Matemáticas e Informática /**
**Doble Grado en Ing. Informática y ADE**
UNIVERSIDAD POLITÉCNICA DE MADRID

**NORMAS:** Este es un cuestionario que consta de 7 preguntas. Todas las preguntas son de respuesta simple excepto la pregunta 7 que es una pregunta de desarrollo. La puntuación total del examen es de 10 puntos. La duración total es de una hora. No olvidéis rellenar vuestros datos.

Sólo hay una respuesta válida a cada pregunta de respuesta simple. Contestad a las preguntas de respuesta simple en los recuadros destinados a tal efecto en la esquina superior derecha de esta página. Toda pregunta incorrectamente contestada restará del examen una cantidad de puntos igual a la puntuación de la pregunta dividido por el número de alternativas ofrecidas en la misma.

---

## Cuestionario

(1 punto) 1. La figura 1 muestra una implementación del conocido problema de *lectores/escritores* mediante monitores. **Se pide** marcar la afirmación correcta:

   (a) La implementación del método `finEscribir` es correcta.

   (b) La implementación del método `finEscribir` es incorrecta.

(1 punto) 2. Seguimos con ese mismo código. **Se pide** señalar la respuesta correcta:

   (a) La sentencia `cLeer.signal();` en el método `inicioLeer` es necesaria.

   (b) La sentencia `cLeer.signal();` en el método `inicioLeer` es innecesaria.

(1½ puntos) 3. Con el mismo código, nos referimos ahora a la sentencia `cLeer.signal();` del método `finLeer`. **Se pide** señalar la respuesta correcta.

   (a) Es una sentencia correcta y necesaria.

   (b) Es una sentencia correcta, pero innecesaria.

   (c) Es una sentencia incorrecta, puede provocar que threads ejecuten `inicioLeer` indebidamente.

(1 punto) 4. Supóngase que dos threads ejecutan los siguientes códigos (nótese que se ejecutan indefinidamente dentro de un bucle):

```
while (true) {
   A1;
   ch1.in.read();
   A2;
   ch2.out.write(null);
   A3;
}
```

```
while (true) {
   B1;
   ch1.out.write(null);
   B2;
   ch2.in.read();
   B3;
}
```

Suponemos que no hay otros threads que afecten al problema aparte de los mostrados. **Se pide** señalar la respuesta correcta.

   (a) `A2` y `B3` nunca se ejecutan simultáneamente.

   (b) `A2` y `B2` nunca se ejecutan simultáneamente.

```
class LectoresEscritores1 {
    private int lectores;
    private int escritores;
    private Monitor mutex;
    private Monitor.Cond cLeer;
    private Monitor.Cond cEscribir;
    public LectoresEscritores1(){
        lectores = 0;
        escritores = 0;
        mutex = new Monitor();
        cLeer     = mutex.newCond();
        cEscribir = mutex.newCond();
    }
    public void inicioLeer() {
        mutex.enter();
        if (escritores > 0) {
            cLeer.await();
        }
        lectores++;
        cLeer.signal();
        mutex.leave();
    }
    public void finLeer() {
        mutex.enter();
        lectores--;
        if (lectores+escritores == 0 &&
            cEscribir.waiting()>0) {
            cEscribir.signal();
        } else {
            cLeer.signal();
        }
        mutex.leave();
    }
    public void inicioEscribir() {
        mutex.enter();
        if (lectores+escritores > 0) {
            cEscribir.await();
        }
        escritores++;
        mutex.leave();
    }
    public void finEscribir() {
        mutex.enter();
        escritores--;
        if (cLeer.waiting() > 0) {
            cLeer.signal();
        }
        if (cEscribir.waiting() > 0) {
            cEscribir.signal();
        }
        mutex.leave();
    }
}
```

**TIPO:** $LE = (l : \mathbb{Z} \times e : \mathbb{Z})$
**INVARIANTE:** $\forall r \in LE \bullet\ r.e \geq 0 \land r.l \geq 0 \land r.e \leq 1 \land ((r.e > 0 \Rightarrow r.l = 0) \land (r.l > 0 \Rightarrow r.e = 0))$
**INICIAL:** self $= (0, 0)$

**CPRE:** self.$e = 0$
**inicioLeer()**
**POST:** self $= (\text{self}^{pre}.l + 1, \text{self}^{pre}.e)$
**CPRE:** self.$e = 0 \land$ self.$l = 0$
**inicioEscribir()**
**POST:** self $= (\text{self}^{pre}.l, \text{self}^{pre}.e + 1)$
**CPRE:** *cierto*
**finLeer()**
**POST:** self $= (\text{self}^{pre}.l - 1, \text{self}^{pre}.e)$
**CPRE:** *cierto*
**finEscribir()**
**POST:** self $= (\text{self}^{pre}.l, \text{self}^{pre}.e - 1)$

Figura 1: Lectores/Escritores con monitores (especificación a la derecha).

**Apellidos:** **Nombre:** **Matrícula:**

(1½ puntos) 5. Se define una clase *servidor* `S` y dos clases *cliente* `C1` y `C2` que se comunican a través de los canales de comunicación `pet1` y `pet2` (se muestran las partes relevantes del código para este problema).

```
class S implements CSProcess {
  public void run() {
    boolean x = true;
    boolean y = false;
    final int PET1 = 0;
    final int PET2 = 1;
    final Guard[] entradas =
        {pet1.in(), pet2.in()};
    final Alternative servicios =
        new Alternative (entradas);
    final boolean[] sincCond =
        new boolean[2];
    while (true) {
      sincCond[PET1] = x;
      sincCond[PET2] = y;
      int sel = servicios.fairSelect(sincCond);
      switch (sel) {
      case PET1:
        pet1.in().read();
        x = !x;
        y = !y;
        break;
      case PET2:
        pet2.in().read();
        y = !y;
        break;
      }//end case
    }//end while
  }//end run()
```

```
class C1 implements CSProcess {
  public void run() {
    while (true) {
      pet1.out().write(null);
      System.out.print("A");
    }
  }
}

class C2 implements CSProcess {
  public void run() {
    while (true) {
      pet2.out().write(null);
      System.out.print("B");
    }
  }
}
```

Dado un programa concurrente con tres procesos `s`, `c1` y `c2` de las clases `S`, `C1` y `C2`, **se pide:** marcar cuál de las siguientes afirmaciones es la correcta:

(a) La salida del programa es `BA` y los tres procesos se quedarán bloqueados.

(b) La salida del programa es `AB` y los tres procesos se quedarán bloqueados.

(c) La salida del programa es `ABABABAB`... indefinidamente.

(d) Ninguna de las respuestas anteriores.

(1 punto) 6. Se define una clase *ContadorCSP* que implementa un recurso compartido *Contador* usando JCSP. Dicho recurso dispone de 2 operaciones, `inc()` y `dec()`, cuya *CPRE* = *cierto*:

```
public class ContadorCSP
       implements CSProcess {

  private Any2OneChannel chInc =
       Channel.any2One();
  private Any2OneChannel chDec =
       Channel.any2one();
  private int contador = 0;

  static final int INC = 0;
  static final int DEC = 1;

  public void inc() {
    chInc.out().write(null);
    contador ++;
  }

  public void dec() {
    chDec.out().write(null);
    contador --;
  }
```

```
  public void run() {

    Guard[] puertos =
       new AltingChannelInput[2];
    puertos[INC] = chInc.in();
    puertos[DEC] = chDec.in();

    final Alternative servicios =
       new Alternative(puertos);

    while (true) {

      int petIndex = servicios.fairSelect();

      switch(petIndex) {
      case INC:
        chInc.in().read();
        break;
      case DEC:
        chDec.in().read();
        break;
    }
  }
}
```

**Se pide** señalar la respuesta correcta.

(a) Se trata de una implementación incorrecta del recurso compartido

(b) Se trata de una implementación correcta del recurso compartido.

**Apellidos:** **Nombre:** **Matrícula:**

(3 puntos) 7. A continuación mostramos una especificación formal de un recurso para jugar al popular juego de mesa *Los nómadas que cantan*. Los desarrolladores han ideado un recurso compartido para representar las materias primas que tiene el jugador, siendo estas materias primas cereal, agua y madera.

**C-TAD** MateriasPrimas
**OPERACIONES**
**ACCIÓN** cargarCereal:
**ACCIÓN** cargarAgua:
**ACCIÓN** cargarMadera:
**ACCIÓN** avanzar:
**ACCIÓN** reparar:
**SEMÁNTICA**
**DOMINIO:**
**TIPO:** $\mathit{MateriasPrimas} = (\mathit{cereal} : \mathbb{N} \times \mathit{agua} : \mathbb{N} \times \mathit{madera} : \mathbb{N})$
**INICIAL:** $\text{self} = (0, 0, 0)$
**INVARIANTE:** $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} < 10$

**CPRE:** $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} + 1 < 10$
**cargarCereal**
**POST:** $\text{self}^{pre} = (c, a, m) \land \text{self} = (c + 1, a, m)$

**CPRE:** $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} + 1 < 10$
**cargarAgua**
**POST:** $\text{self}^{pre} = (c, a, m) \land \text{self} = (c, a + 1, m)$

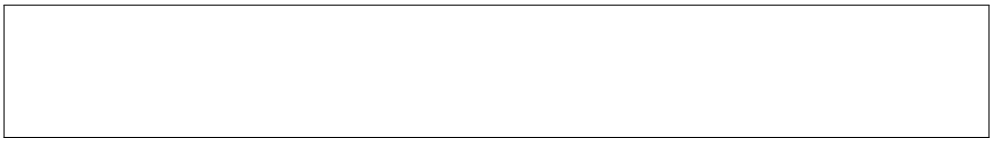
**CPRE:** $\text{self}.\mathit{cereal} + \text{self}.\mathit{agua} + \text{self}.\mathit{madera} + 1 < 10$
**cargarMadera**
**POST:** $\text{self}^{pre} = (c, a, m) \land \text{self} = (c, a, m + 1)$

**CPRE:** $\text{self}.\mathit{cereal} > 0 \land \text{self}.\mathit{agua} > 0$
**avanzar**
**POST:** $\text{self}^{pre} = (c, a, m) \land \text{self} = (c - 1, a - 1, m)$

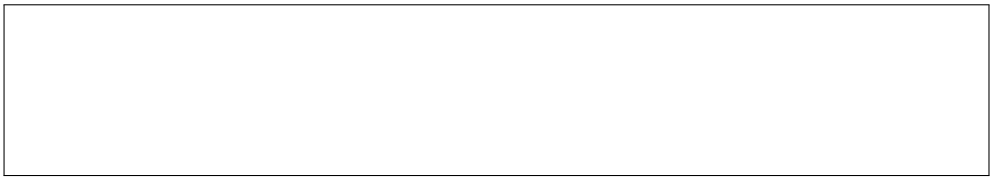
**CPRE:** $\text{self}.\mathit{agua} > 0 \land \text{self}.\mathit{madera} > 0$
**reparar**
**POST:** $\text{self}^{pre} = (c, a, m) \land \text{self} = (c, a - 1, m - 1)$

**Se pide:** implementar este recurso compartido usando monitores siguiendo la metodología vista en clase. En cuanto al código de desbloqueos os recomendamos implementar en el método `desbloqueoSimple()` un código genérico para todas las operaciones.

```
class MateriasPrimas  {
  // estado del recurso
```



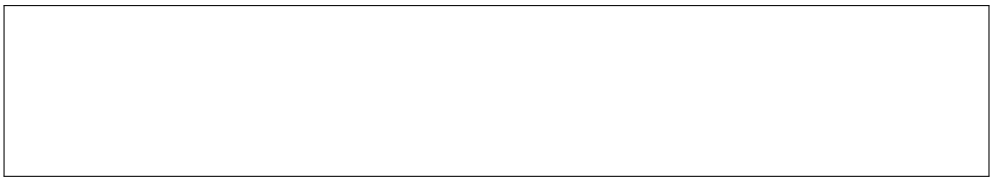
```
  // Declaramos monitores y colas condition
```



```
  public MateriasPrimas() {
```

```
  }

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



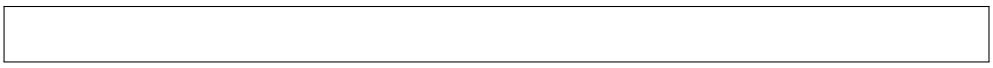
```
    // codigo de la operacion
```

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

```
  }

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

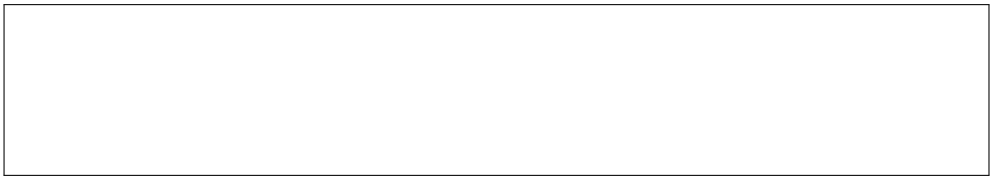
```
    // codigo de la operacion
```



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

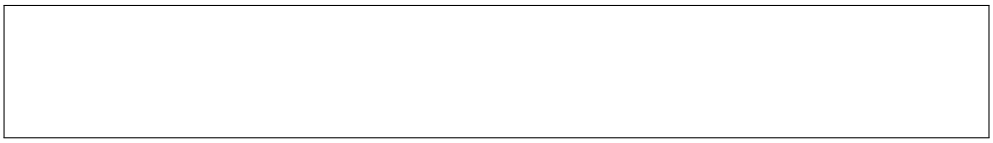
```
 }

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



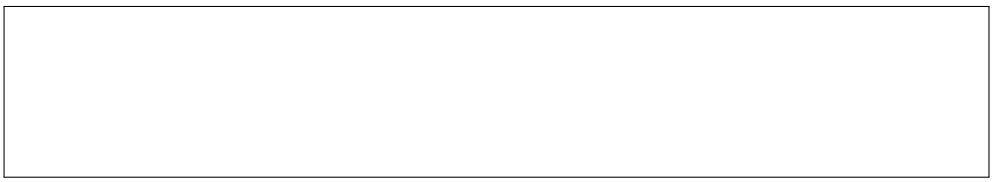
```
   // codigo de la operacion
```

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



```
 }

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



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

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

```
}

 public void reparar() {
   // 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() {
```

```
    }
}
```