**CSD Prácticas**

**Práctica 2**

*Pregunta 1*

En el problema de los 5 filósofos, la condición de Coffman de ‘exclusión mutua’ implica principalmente:

Que los filósofos no pueden usar el recurso ‘mesa’ a la vez.

Que los filósofos no pueden usar el recurso ‘tenedor’ a la vez.

Que los filósofos no pueden pensar (filosofar) a la vez.

Que los filósofos pueden compartir los cinco tenedores de la mesa.

Que los filósofos pueden compartir la mesa.

*Pregunta 2*

En la solución para la Versión4 (capacidad mesa) de los 5 filósofos, puede producirse que todas las condiciones de Coffman se cumplan simultáneamente.

Falso, pues la condición de ‘exclusión mutua’ nunca se cumple.

Falso, pues la condición de ‘espera circular’ nunca se cumple.

Verdadero, en cuyo caso decimos que hay riesgo de interbloqueo.

Verdadero, en cuyo caso decimos que hay una condición de carrera.

Verdadero, en cuyo caso todos los filósofos están comiendo.

*Pregunta 3*

La solución para la Versión4 (capacidad mesa) de los 5 filósofos:

Nunca podrá presentar interbloqueos, pues todos los métodos de la mesa llevan la etiqueta synchronized.

Nunca podrán presentar mayor probabilidad de interbloqueo conforme se aumenta el tiempo que tarda un filósofo entre coger su primer tenedor y su segundo tenedor.

Podrá presentar mayor probabilidad de interbloqueo conforme se disminuye el tiempo que tarda un filósofo entre coger su primer tenedor y su segundo tenedor.

Nunca podrá presentar interbloqueos, pues rompe al menos una de las condiciones de Coffman.

*Pregunta 4*

Una manera alternativa de resolver los interbloqueos en el ejemplo de los filósofos es:

Usar un único filósofo que coja los dos tenedores o ninguno (BothOrNonePhilo) y 4 filósofos normales.

Establecer un tiempo de espera (delay) más corto entre coger el tenedor derecho y coger el izquierdo.

Hacer que un filósofo salga de la mesa a pensar (pondering) cuando tiene cogido el tenedor derecho y espera por el izquierdo.

Hacer que todos los filósofos sean diestros.

Hacer que todos los filósofos sean zurdos (cojan primero el izquierdo y luego el derecho).

*Pregunta 5*

La solución para la Versión4 (capacidad mesa) de los 5 filósofos requiere:

Un tipo de filósofos que realice las operaciones de coger los tenedores de forma atómica.

Dos tipos de mesa, una para los filósofos pares y otra para los filósofos impares.

Dos tipos de filósofos, cada uno cogiendo los tenedores en un orden distinto.

Dos tipos de filósofos, de forma que cuando los filósofos de un tipo comen los del otro tipo piensa, y viceversa.

Una mesa en la que se controle cuantos filósofos hay sentados en ella.

**Práctica 3** (de estas hay 1 que está MAL)

*Pregunta 1*

En el territorio Terrain0, que emplea la sincronización básica de Java, en el método “move”, si se reemplaza el método notifyAll() por el método notify():

Java dará un error de compilación, pues el método notify() es un método de la clase Condition y no de la clase Object.

Java dará un error de compilación, pues el método notify() no existe como método en la clase Object.

Java dará un error de compilación, pues el método notify() es un método de la clase ReentrantLock y no de la clase Object.

Java no dará ningún erro de compilación, pero en la ejecución solamente se reactivará una de las hormigas esperando en la condición.

El problema funcionará igual de bien en todos los casos, pues ambos métodos se pueden utilizar de forma indistinta.

*Pregunta 2*

Tras implementar de forma correcta Terrain2 (Monitor general con una variable condition por celda del territorio), los métodos del territorio (hi, bye, move):

Utilizan un lock por cada celda del territorio.

Permiten controlar que no se produzcan interbloqueos entre las hormigas.

Se ejecutan en exclusión mutua.

Deben llevar la etiqueta synchronized en su declaración.

Podrían provocar condiciones de carrera.

*Pregunta 3*

En Terrain3, la solución propuesta rompe con la condición de Coffman:

Exclusión mutua.

Retención y espera.

No expulsión.

No se rompe ninguna condición de Coffman.

Espera circular.

*Pregunta 4*

En el territorio Terrain0, que emplea la sincronización básica de Java, los métodos del territorio (hi, bye, move):

Utilizan una condición por cada celda del territorio.

Permiten controlar que no se produzcan interbloqueos entre las hormigas.

Se ejecutan en exclusión mutua entre sí.

Utilizan un lock por cada celda del territorio.

Pueden provocar condiciones de carrera.

*Pregunta 5*

En el territorio Terrain0, que emplea la sincronización básica de Java en el método “move”, si se reemplaza el método wait() por el método await():

El problema funcionará igualmente, pues ambos métodos se pueden emplear de forma indistinta.

Java no dará ningún error de compilación, pero en la ejecución la hormiga no se quedará suspendida en la condición.

Java dará un error de compilación, pues el método await() es un método de la clase ReentrantLock y no de la clase Object.

Java dará un error de compilación, pues el método await() es un método de la clase Condition y no de la clase Object.

Java dará un error de compilación, pues el método await() es un método de la clase Object.