sakai.iframe.site

FLIP UD01-1: Que es Prog. Concurrente

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Uno de los inconvenientes de la programación concurrente es:

- ✔ ◯ **Que resulta muy compleja su depuración.**
- ◯ **Que permite una mayor flexibilidad de los programas.**
- ◯ **Que permite una mayor escalabilidad.**
- ◯ **Que ofrece un menor hueco semántico.**

Respuesta correcta: A

Preguntas 2 de 5

2.0 Puntos

Indique qué afirmación es FALSA:

- ◯ **La programación concurrente es en general más eficiente y escalable que la secuencial.**
- ✖ ◯ **Las aplicaciones concurrentes suelen responder a un esquema modular.**
- ◯ **La programación secuencial destaca por su flexibilidad.**
- ◯ **La programación concurrente mejora la interactividad y flexibilidad.**

Respuesta correcta: C

Preguntas 3 de 5

2.0 Puntos

Un programa secuencial es aquel en el que sus instrucciones se van ejecutando .....

- ◯ **En paralelo.**
- ✔ ◯ **En serie, una tras otra.**
- ◯ **En concurrencia.**
- ◯ **En serie, varias a la vez.**

Respuesta correcta: B

Preguntas 4 de 5

2.0 Puntos

La programación concurrente permite mejorar ....

- ◯ **La depuración de las aplicaciones**
- ◯ **Las prestaciones del sistema.**
- ◯ **La cooperación de las actividades**
- ✖ ◯ **Todas las opciones anteriores.**

Respuesta correcta: B

Preguntas 5 de 5

2.0 Puntos

Indique cuál de los siguientes es un ejemplo de un programa concurrente:

- ◯ **Un proceso con un solo hilo de ejecución.**
- ◯ **Un compilador ejecutándose sobre un sistema operativo monoprogramado.**
- ✔ ◯ **Un navegador web con múltiples hilos de ejecución.**
- ◯ **Un videojuego con un único proceso mono-hilo.**

Respuesta correcta: C

Anuncios recientes

sakai.iframe.site

FLIP UD01-2: Creación de hilos en Java

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Los hilos en Java son un contexto de ejecución formado por:

- ✔ **El código del hilo, los datos (propios del hilo, y compartidos entre varios hilos) y la JVM.**
- **Código, Datos (los atributos propios del hilo) y una CPU virtual.**
- **Código y Datos.**
- **El código que se indica en el método void run() de la interfaz Runnable**

Respuesta correcta: A

Preguntas 2 de 5

2.0 Puntos

Para lanzar a ejecución un hilo en Java debemos utilizar el método:

- **run()**
- **execute()**
- **create()**
- ✔ **start()**

Respuesta correcta: D

Preguntas 3 de 5

2.0 Puntos

Indique cuál de las siguientes afirmaciones es FALSA:

- ○ **Al crear un hilo podemos asociarle un nombre utilizando el constructor de la clase Thread.**
- ○ **El método setName() de la clase Thread permite asociar un nombre al hilo.**
- ○ **Java asigna un nombre al hilo de forma automática cuando se crea el hilo.**
- ✔ ○ **Se puede obtener el nombre asociado al hilo con el método printName() de la clase Thread.**

Respuesta correcta: D

Preguntas 4 de 5

2.0 Puntos

Si queremos que una clase A, que ya extiende de otra clase, pueda ser ejecutada como un hilo...

- ○ **La clase A debe extender Thread.**
- ○ **La clase A debe extender la clase Runnable.**
- ○ **La clase A no podrá ser ejecutada nunca como hilo, pues Java no soporta herencia múltiple.**
- ✔ ○ **La clase A debe implementar Runnable.**

Respuesta correcta: D

Preguntas 5 de 5

2.0 Puntos

¿Cuándo es posible asignarle un nombre a un hilo en Java?

- ○ **No se puede asignar un nombre. Java lo asigna automáticamente.**
- ○ **Se asigna únicamente en el momento de su creación.**
- ○ **Se puede asignar cuando el hilo ya haya empezado a ejecutarse.**
- ✔ ○ **Se puede asignar en su creación o en cualquier momento de su ejecución.**

Respuesta correcta: D

Anuncios recientes

sakai.iframe.site

FLIP UD01-3: Ejemplos Aplicaciones Concurrentes

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

En el problema de los cinco filósofos...

- ○ **Los cinco filósofos comparten los cinco tenedores, y cada tenedor puede ser usado solamente por un filósofo en cada momento.**
- ○ **Los cinco filósofos comparten los cinco tenedores y cada tenedor puede ser usado por varios filósofos en cada momento.**
- ○ **Cada par de filósofos vecinos comparte un tenedor, que puede ser usado por ambos filósofos en cada momento.**
- ✔ ○ **Cada par de filósofos vecinos comparte un tenedor, que puede ser usado solamente por un filósofo en cada momento.**

Respuesta correcta: D

Preguntas 2 de 5

2.0 Puntos

La aplicación Therac-25 es un ejemplo real que muestra...

- ○ **la eficiencia de las aplicaciones concurrentes (frente a las aplicaciones secuenciales).**
- ○ **La utilización de thread-pools, creando conjuntos de hilos en espera.**
- ○ **los problemas generados por un mal diseño de la aplicación concurrente.**
- ○ **la dificultad de sincronizar un elevado número de hilos en una aplicación concurrente.**

Respuesta correcta: C

Preguntas 3 de 5

2.0 Puntos

Indique cuál de los siguientes NO es un problema clásico de la programación concurrente:

- ◯ **Tres lectores**
- ✖ ◯ **Cinco filósofos**
- ◯ **Productor-Consumidor**
- ◯ **Lectores-Escritores**

Respuesta correcta: A

Preguntas 4 de 5

2.0 Puntos

Indique cuál de los siguientes NO es un ejemplo de aplicación concurrente real:

- ◯ **Therac-25**
- ◯ **Videojuego FIFA**
- ✔ ◯ **Intérprete de órdenes bash**
- ◯ **Servidor Web Apache**

Respuesta correcta: C

Preguntas 5 de 5

2.0 Puntos

En el problema de Lectores-Escritores, el escritor tendrá que esperar si accediendo al recurso hay....

- ◯ **Un escritor**
- ✔ ◯ **Algún hilo (lector o escritor)**
- ◯ **Un lector**
- ◯ **No hay nadie**

Respuesta correcta: B

sakai.iframe.site

FLIP UD01-Resumen

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Indique cuáles de los siguientes ejemplos se corresponden con ejemplos de APLICACIONES CONCURRENTES:

- [ ] Una hoja de cálculo con un hilo para gestionar la entrada de texto, otro para recalcular el valor de las celdas
- [x] y actualizar la visión del documento, otro para grabar en segundo plano las versiones transitorias, otro para gestionar los menús, etc.
- [ ] Un compilador que se ejecute sobre un sistema operativo monoprogramado.
- [x] Un videojuego en el que se utilice un hilo distinto por cada elemento móvil que aparezca en la pantalla.
- [x] Cualquier aplicación distribuida que siga el modelo cliente/servidor, visto en las asignaturas de Redes.

Respuesta correcta: A, C, D

Preguntas 2 de 5

2.0 Puntos

Sea el siguiente código en Java:

```
public class T2 extends Thread {
private int nivel;
public T2(int n){
nivel = n;
}
public void creaHilo(int i) {
T2 h = new T2(i);
h.start();
System.out.println("Hilo de nivel "+i+" creado.");
}
public void run() {
if (nivel>0)

creaHilo(nivel-1);
System.out.println("Fin hilo. Nivel:" + nivel);
}
public static void main(String[] argv) {
for (int i=1; i<3; ++i)
new T2(2).start();
}
}
```

Marque cuál(es) de las siguientes sentencias son VERDADERAS:

- ✔ ☐ Al ejecutar este programa llegan a generarse seis hilos aparte del inicial.
- ☐ Este programa no ejecuta ninguno de los hilos generados, pues jamás se invoca el método run() de éstos.
- ✔ ☐ Al ejecutar este programa habrá hilos cuyo atributo nivel valga 2, 1 o 0.
- ☐ Al ejecutar este programa no se generará ningún hilo cuyo atributo nivel valga cero.

Respuesta correcta: A, C

Preguntas 3 de 5

2.0 Puntos

Respecto a la utilización de hilos en Java, indique qué sentencias siguientes son VERDADERAS:

- ✔ [ ] Los objetos definidos en Java se mantienen en el heap. Pasan a ser residuos cuando ningún hilo los referencia. Por ello el run-time de Java utiliza un hilo "recolector" que los "elimina" en ese caso.
- ✔ [ ] Para asignar nombre a los hilos se puede utilizar el método setName().
- ✔ [ ] Para definir hilos basta con crear una clase que implante la interfaz Runnable o crear una clase que extienda la clase Thread.
- ✔ [ ] El código a ejecutar por cada hilo debe estar contenido en su método run(), que debe ser invocado llamando a start().

Respuesta correcta: A, B, C, D

Preguntas 4 de 5

2.0 Puntos

Indique cuál o cuáles de las siguientes sentencias representan INCONVENIENTES de la programación concurrente:

- ✔ [ ] Presenta algunos problemas potenciales que deben evitarse: interbloqueos, condiciones de carrera, inconsistencia, ...
- ✔ [ ] Genera aplicaciones que normalmente son más difíciles de depurar.
- ✖ [ ] Sólo está soportada en ordenadores modernos con procesadores de múltiples núcleos.
- [ ] Sólo tiene sentido en aplicaciones interactivas colaborativas.

Razonamiento:

Respuesta correcta: A, B

Preguntas 5 de 5

2.0 Puntos

Sobre las actividades de una aplicación concurrente, indique cuál(es) de las siguientes sentencias son VERDADERAS:

- [ ] Para que las actividades de una aplicación concurrente avancen en paralelo podemos utilizar el soporte de
- ✔ un sistema operativo multiprogramado en ordenadores con un solo procesador. Su planificador ofrecerá una visión lógica de avance simultáneo.
- ✔ [ ] Para que las actividades de una aplicación concurrente avancen en paralelo podemos utilizar los múltiples núcleos de un procesador moderno, para que el avance simultáneo sea real.
- [ ] Para que las actividades de una aplicación concurrente avancen en paralelo siempre tendremos que instalar esa aplicación en múltiples ordenadores e intercomunicarlos de alguna manera.
- ✔ [ ] Para que las actividades de una aplicación concurrente avancen en paralelo podemos tener cada actividad en una máquina diferente, y comunicarlas intercambiando mensajes.

Respuesta correcta: A, B, D

sakai.iframe.site

FLIP UD02-1. Cooperación hilos

Volver a la Lista de Exámenes

Parte 1 de 1 - 5.0 Puntos

Preguntas 1 de 5

1.0 Puntos

Indique cuál de las siguientes afirmaciones es FALSA:

- ○ **Dos hilos de dos procesos distintos pueden comunicarse entre sí utilizando el mecanismo de memoria compartida.**
- ○ **Dos hilos del mismo proceso pueden comunicarse entre sí utilizando el mecanismo de memoria compartida.**
- ✖ ○ **Dos hilos de dos procesos distintos pueden comunicarse entre sí utilizando el mecanismo de intercambio de mensajes.**
- ○ **Dos hilos del mismo proceso pueden comunicarse entre sí utilizando el mecanismo de intercambio de mensajes.**

Respuesta correcta: A

Preguntas 2 de 5

1.0 Puntos

Una sección se ejecuta en exclusión mutua cuando:

- ✔ ○ **Solamente un hilo puede ejecutarla en cada momento.**
- ○ **Solamente un hilo puede ejecutarla en cada procesador del sistema.**
- ○ **Un hilo debe suspenderse hasta que se cumpla determinada condición de dicha sección.**
- ○ **Las variables que la componen son mutuamente excluyentes entre sí.**

Respuesta correcta: A

Preguntas 3 de 5

1.0 Puntos

En el problema Lectores-Escritores se requiere acceso en exclusión mutua:

- ✖ ◯ **Solamente para los escritores entre sí.**
- ◯ **Solamente para los lectores entre sí.**
- ◯ **Para los escritores entre sí; y para escritores y lectores entre sí. Pero los lectores pueden acceder de forma concurrente al recurso.**
- ◯ **Cualquier tipo de hilo (lector o escritor) debe utilizar el recurso siempre en exclusión mutua.**

Respuesta correcta: C

Preguntas 4 de 5

1.0 Puntos

En el problema Productor-Consumidor se requiere sincronización condicional:

- ◯ **Para suspender a un productor si el buffer está vacío.**
- ✔ ◯ **Para suspender a un productor si el buffer está lleno.**
- ◯ **Para suspender a un consumidor si el buffer está lleno.**
- ◯ **No se requiere nunca la sincronización condicional.**

Respuesta correcta: B

Preguntas 5 de 5

1.0 Puntos

La sincronización condicional implica:

- ◯ **Suspender a un hilo durante un tiempo máximo especificado (en milisegundo).**
- ✖ ◯ **Suspender a un hilo hasta que se cumpla una determinada condición. Dicho hilo suspendido estará encargado de detectar cuándo la condición se cumple y reactivarse entonces.**
- ◯ **Suspender a un hilo hasta que se cumpla una determinada condición. Otro hilo se encargará de reactivar al hilo suspendido.**
- ◯ **Interrumpir a un hilo suspendido en una condición, porque se ha cumplido el tiempo máximo especificado.**

Respuesta correcta: C

sakai.iframe.site

FLIP UD02-3. Condiciones de Carrera

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Una acción es atómica si:

- ◯ **Es una única instrucción interrumplible.**
- ✔ ◯ **No puede dividirse en acciones menores.**
- ◯ **Consta de estados intermedios visibles a otros hilos.**
- ◯ **Garantiza que los hilos puedan acceder a estados intermedios.**

Respuesta correcta: B

Preguntas 2 de 5

2.0 Puntos

Se produce una condición de carrera cuando varios hilos acceden a un mismo objeto compartido...:

- ◯ **Y todos los hilos leen una misma variable de ese objeto.**
- ◯ **Y cada hilo imprime una variable distinta de ese objeto.**
- ◯ **Y uno o más hilos actualiza el valor de una variable del objeto, dejándolo en un estado consistente.**
- ✔ ◯ **Y uno o más hilos actualiza el valor de una variable del objeto de forma inconsistente.**

Respuesta correcta: D

Preguntas 3 de 5

2.0 Puntos

Los protocolos de entrada/salida a la sección crítica deben garantizar:

- ✔ **Exclusión mutua, progreso y espera limitada.**
- **Exclusión mutua y espera ilimitada.**
- **Que haya siempre un protocolo de entrada, aunque no es necesario el de salida.**
- **Que la sección crítica pueda ser ejecutada concurrentemente por distintos hilos.**

Respuesta correcta: A

Preguntas 4 de 5

2.0 Puntos

Se dice que un código es thread-safe si:

- **Solamente puede ser ejecutado por un único hilo.**
- **Si puede ser ejecutado concurrentemente por varios hilos.**
- ✔ **Si varios hilos lo ejecutan a la vez sin que pueda nunca producirse condiciones de carrera.**
- **Si utiliza protocolos de entrada al código que garanticen el progreso.**

Respuesta correcta: C

Preguntas 5 de 5

2.0 Puntos

Indique cuál de las siguientes afirmaciones es FALSA:

- **Una sección crítica es un fragmento de código que puede provocar condiciones de carrera.**
- **Una sección crítica es un fragmento de código que accede a variables u objetos compartidos por más de un hilo.**
- ✔ **Un fragmento de código en el que se acceden a las variables locales de un hilo es un ejemplo de sección crítica.**
- **Un fragmento de código en el que se accede a objetos no inmutables compartidos entre varios hilos es un ejemplo de sección crítica.**

Respuesta correcta: C

Anuncios recientes

sakai.iframe.site

FLIP UD02-4. Locks

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Un lock es un objeto que, cuando un hilo H1 utiliza su operación cerrar lock:

- ○ **Si estaba ya cerrado por H1, entonces H1 se bloquea.**
- ○ **Si estaba cerrado ya por otro hilo, entonces H1 lo cierra también y continúa adelante.**
- ✔ ○ **Si estaba abierto, lo cierra.**
- ○ **Todas las anteriores.**

Respuesta correcta: C

Preguntas 2 de 5

2.0 Puntos

Un lock es un objeto que, cuando un hilo H1 utiliza su operación abrir lock:

- ○ **Si estaba ya abierto por H1, entonces no tiene efecto.**
- ○ **Si estaba cerrado por otro hilo, entonces no tiene efecto.**
- ○ **Si estaba cerrado por H1, entonces se abre el lock.**
- ✔ ○ **Todas las anteriores.**

Respuesta correcta: D

Preguntas 3 de 5

2.0 Puntos

Java ofrece un mecanismo de lock implícito que consiste en:

- ◯ **Que todo objeto posee un lock asociado llamado Lock, con instrucciones Lock.open() y Lock.close() para abrirlo y cerrarlo, respectivamente.**
- ◯ **Que todo objeto posee un lock asociado oculto para el programador.**
- ◯ **Que se debe hacer uso del objeto Lock para sincronizar las tareas.**
- ✖ ◯ **Que solamente los objetos de la clase Thread tienen un lock asociado.**

Respuesta correcta: B

Preguntas 4 de 5

2.0 Puntos

En Java, etiquetar con synchronized los métodos de una clase implica:

- ◯ **Que estos métodos se ejecutarán de forma simultánea.**
- ✔ ◯ **Que estos métodos se ejecutarán en exclusión mutua entre sí.**
- ◯ **Que estos métodos abren todos a la vez el lock implícito de Java.**
- ◯ **Que estos métodos cierran todos a la vez el lock implícito de Java.**

Respuesta correcta: B

Preguntas 5 de 5

2.0 Puntos

Indique cuál de las siguientes afirmaciones es FALSA:

- ◯ **Las sentencias sincronizadas de Java implican utilizar la etiqueta synchronized en la declaración del método.**
- ◯ **Al usar sentencias sincronizadas de Java, se pueden intercalar métodos de un mismo objeto, pero el código de esos métodos se ejecuta en exclusión mutua.**
- ◯ **Al usar sentencias sincronizadas, se puede utilizar más de un lock dentro de un mismo método.**
- ◯ **Los métodos sincronizados de un mismo objeto siempre estarán en exclusión mutua todos ellos entre sí.**

Respuesta correcta: A

Anuncios recientes

sakai.iframe.site

FLIP UD02-Resumen

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

La sentencia h1.join(), siendo h1 una instancia de la clase Thread, permite...

- ✔ **Suspender el hilo actual la espera de que acabe h1**
- **Reactivar la ejecución de h1**
- **Suspender h1 a la espera de que acabe el hilo actual**
- **Lanzar a ejecución h1**

Respuesta correcta: A

Preguntas 2 de 5

2.0 Puntos

Un hilo en Java puede pasar del estado EN EJECUCION al estado PREPARADO:

- **Nunca puede producirse ese cambio de estado.**
- ✖ **Si realiza una llamada a Object.wait()**
- **Si es expulsado por el planificador**
- **Si espera la obtención de un lock**

Respuesta correcta: C

Preguntas 3 de 5

2.0 Puntos

Respecto al estado de planificación de los hilos en Java, indique cuál o cuáles de las siguientes sentencias son VERDADERAS:

- [ ] El método yield() de la clase Thread hace que un hilo pase del estado "preparado" al estado "suspendido".
- [ ] Si un hilo utiliza el método wait() de un objeto pasa del estado "en ejecución" al "preparado".
- [ ] Los estados Blocked, Waiting y Timed-waiting de Java corresponden al estado "suspendido" visto en Fundamentos de los Sistemas Operativos.
- [ ] Un hilo Java no puede pasar del estado "en ejecución" al estado "preparado" de ninguna forma.

Razonamiento:

Respuesta correcta: C

Preguntas 4 de 5

2.0 Puntos

En Java, si un hilo utiliza el método wait() de un objeto pasa del estado EN EJECUCION al estado:

- ◯ **Preparado**
- ◯ **Timed-waiting**
- ◯ **Blocked**
- ✔ ◯ **Waiting**

Respuesta correcta: D

Preguntas 5 de 5

2.0 Puntos

Una sección se ejecuta en exclusión mutua cuando:

- ✔ ◯ **Solamente un hilo puede ejecutarla en cada momento**
- ◯ **Solamente un hilo puede ejecutarla en cada procesador del sistema**
- ◯ **Un hilo debe suspenderse hasta que se cumpla determinada condición de la sección**
- ◯ **Las variables que la componen son mutuamente excluyentes entre sí**

Respuesta correcta: A

Anuncios recientes

sakai.iframe.site

FLIP UD03-Resumen

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Marque cuáles de los siguientes ejemplos son ejemplos VÁLIDOS de mecanismos de sincronización:

- [x] Semáforos
- [ ] Un planificador expulsivo, como pueda ser uno que utilice el algoritmo Round-Robin o uno que gestione prioridades expulsivas.
- [ ] El uso de memoria compartida.
- [x] Monitores.

Respuesta correcta: A, D

Preguntas 2 de 5

2.0 Puntos

Para construir monitores en Java... (indique cuáles de las siguientes respuestas son CIERTAS):

- [x] Si las llamadas a notify() aparecen como última sentencia en los métodos donde se necesiten, entonces se soporta directamente el modelo de monitores de Brinch Hansen.
- [ ] ... debemos añadir el calificador "protected" a cada uno de los métodos de la clase que vaya a utilizarse como un monitor.
- [ ] ... no es necesario añadir nada especial para que los métodos se ejecuten en exclusión mutua.
- [x] ... utilizaremos los métodos wait(), notify() y notifyAll() (heredados de la clase Object) para implantar la sincronización condicional.

Respuesta correcta: A, D

Preguntas 3 de 5

2.0 Puntos

Para construir monitores en Java... (marque las respuestas CORRECTAS):

✔ ☐ ... utilizaremos los métodos wait(), notify() y notifyAll() (heredados de la clase Object) para implantar sincronización condicional.

☐ ... no es necesario importar ningún "package" para definir los monitores.

☐ ... debemos asegurarnos de que el sistema operativo subyacente admita la creación y gestión de múltiples hilos de ejecución en cada proceso.

✖ ☐ ... utilizaremos el calificativo "synchronized" en la declaración de los atributos y métodos públicos de la clase que vaya a utilizarse como monitor.

Respuesta correcta: A, B

Preguntas 4 de 5

2.0 Puntos

Sea la siguiente clase Java, cuyos métodos serán invocados por múltiples hilos de manera concurrente:

```
public class BoundedBuffer {
private int first,last,numItems,capacity;
private long items[];

public BoundedBuffer(int size){
capacity = size;
items = new long[size];
numItems = first = last = 0;
}
public synchronized void put(long item) {
if (numItems == capacity) try {wait();}catch(Exception e){};
items[last] = item;
last = (last + 1) % capacity; // % = resto de la div.
numItems++;
notify();
}
public synchronized long get() {
long valor;
if (numItems == 0) try{wait();}catch(Exception e){};
valor = items[first];
```

```
first = (first + 1) % capacity;
numItems--;
notify();
return valor;
}
}
```

Indique cuál o cuáles de las siguientes sentencias son CIERTAS:

- ✔ [ ] Este código no funciona adecuadamente, pues debería utilizarse notifyAll() en lugar de notify().
- [ ] Si todas las veces que se invoca a notify() sólo hay un hilo suspendido en wait() o ninguno está suspendido, este monitor funcionará correctamente.
- [ ] Este código no funciona, pues al llamar a wait() no se dejaría abierto el monitor y nadie más podría acceder a él.
- ✔ [ ] Esta clase no funciona adecuadamente, pues las comprobaciones marcadas en negrita deberían asociarse a un bucle while, en lugar de a una sentencia if.

Respuesta correcta: A, D

Preguntas 5 de 5

2.0 Puntos

Respecto a los monitores, indique cuál o cuáles de las siguientes sentencias son CIERTAS:

- ✖ [ ] Un monitor es un mecanismo de sincronización integrado en lenguajes de programación concurrente que permite seleccionar cuántos hilos ejecutarán simultáneamente una sección de código.
- [ ] Un monitor proporciona mecanismos para resolver la sincronización condicional.
- [ ] Un monitor es un mecanismo de sincronización que sólo puede utilizarse en sistemas que implanten una planificación expulsiva.
- [ ] Un monitor es un mecanismo de sincronización proporcionado directamente por el sistema operativo.

Respuesta correcta: B

Anuncios recientes

sakai.iframe.site

FLIP UD04- Interbloqueos

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Indique cuáles de las siguientes NO es una estrategia de solución al problema de los interbloqueos:

- ○ **Estrategia de prevención de interbloqueos**
- ✔ ○ **Estrategia de limitación de interbloqueos**
- ○ **Estrategia de detección y recuperación de interbloqueos**
- ○ **Estrategia de evitación de interbloqueos**

Respuesta correcta: B

Preguntas 2 de 5

2.0 Puntos

Si a partir de un GAR se determina una secuencia segura, se puede afirmar que...

- ✖ ○ **Existe riesgo de interbloqueo**
- ○ **Ningún proceso puede obtener todos los recursos que necesita**
- ○ **Existe interbloqueo**
- ○ **No existe interbloqueo**

Respuesta correcta: D

Preguntas 3 de 5

2.0 Puntos

Indique cuál de las siguientes condiciones es una Condición de Coffman:

- ○ **Espera limitada.**
- ○ **Expulsión.**
- ○ **Progreso.**
- ✔ ○ **Exclusión mutua.**

Respuesta correcta: D

Preguntas 4 de 5

2.0 Puntos

Usando un Grafo de Asignación de Recursos NO podríamos conocer:

- ○ **Si hay riesgo de interbloqueo**
- ○ **Si un proceso está suspendido esperando un recurso**
- ✔ ○ **Si hay riesgo de livelock**
- ○ **Si hay un interbloqueo**

Respuesta correcta: C

Preguntas 5 de 5

2.0 Puntos

Indique cuál de las siguientes condiciones NO es una Condición de Coffman:

- ○ **Exclusión mutua.**
- ○ **No Expulsión.**
- ✔ ○ **Espera limitada.**
- ○ **Espera circular.**

Respuesta correcta: C

Anuncios recientes

sakai.iframe.site

FLIP UD05: Resumen

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Para utilizar un objeto Condition en un "monitor" protegido con ReentrantLock en Java... (indique la(s) respuesta(s) correcta(s))

- [x] ... se recomienda el uso de sus métodos await(), signal() o signalAll().
- [ ] ... no es aconsejable utilizar sus métodos wait(), notify() o notifyAll().
- [x] ... no se requiere proteger los métodos donde se use con el calificador "synchronized".
- [x] ... se pueden utilizar tantos objetos Condition dentro de ese "monitor" como se necesiten.

Respuesta correcta: A, B, C, D

Preguntas 2 de 5

2.0 Puntos

Los objetos Semaphore en Java...:

- [x] ...pueden recibir un valor inicial en su constructor, que indicará cuántas llamadas se podrían realizar al método acquire() (si nadie llama a release()) sin llegar a suspenderse.
- [x] ...pueden utilizarse tanto para garantizar exclusión mutua como para implantar sincronización condicional.
- [ ] ...ofrecen el método await() para suspender a un hilo y el método signal() para reactivar.
- [ ] ...deben ser generados con el método newSemaphore() de la clase ReentrantLock, de manera similar a los objetos Condition.

Respuesta correcta: A, B

Preguntas 3 de 5

2.0 Puntos

Dado el siguiente código Java:

```
class BufferOk implements Buffer {
private int items, head, tail, N;
private int[] data;
Condition notFull, notEmpty;
```

```
ReentrantLock lock;

public BufferOk(int N) {
data= new int[N];
this.N=N;
head = tail = items = 0;
lock= new ReentrantLock();
notFull=lock.newCondition();
notEmpty=lock.newCondition();
}

public int get() {
int x;
try {
lock.lock();
while (items==0) {
System.out.println("Consumer is waiting ...");
try {notEmpty.wait();}
catch(InterruptedException e) {}
}
x=data[head];
head= (head+1)%N; items--;
notFull.notify();
} finally {lock.unlock();}
return x;
}


public void put(int x) {
try{
lock.lock();
while (items==N) {
```

```
System.out.println("Producer is waiting ...");
try {notFull.wait();}
catch(InterruptedException e) {}
}
data[tail]=x; tail= (tail+1)%N; items++;
notEmpty.notify();
} finally {lock.unlock();}
}
```

Indique qué sentencias siguientes son CIERTAS:

- [ ] El código es incorrecto porque no se pueden utilizar múltiples condiciones en un mismo monitor Java.
- [ ] El código es incorrecto pues todos los métodos del monitor deberían ser "synchronized".
- [ ] El código es incorrecto, pues el "finally" empleado en el método get() no incluye la sentencia "return x;".
- [x] El código es incorrecto, pues se ha utilizado wait() y notify() en lugar de await() y signal().

Respuesta correcta: D

Preguntas 4 de 5

2.0 Puntos

Los argumentos enteros utilizados en los constructores de las clases de java.util.concurrent tienen la siguiente funcionalidad (marque las respuestas correctas):

- [ ] En CountDownLatch no se necesita especificar ningún argumento en su constructor.
- [x] En Semaphore, para fijar el valor inicial de su contador. Si es positivo indicará cuántos acquire() se podrán realizar sin suspenderse (asumiendo que nadie llame a release()).
- [x] En CyclicBarrier, para saber cuántas llamadas a await() habrá que esperar para abrir la barrera.
- [x] En Condition, no sirven para nada. No se utiliza un constructor. Basta con llamar al método newCondition() de algún Lock (normalmente, ReentrantLock).

Respuesta correcta: B, C, D

sakai.iframe.site

FLIP UD05-1: Locks y Variables Condicion

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

La interfaz Condition ofrece el método:

- ○ **notifyAll()**
- ○ **notify()**
- ✔ ○ **signal()**
- ○ **wait()**

Respuesta correcta: C

Preguntas 2 de 5

2.0 Puntos

Para utilizar un objeto Condition en un "monitor" protegido con ReentrantLock en Java...

- ○ **Se deben utilizar sus métodos wait(), notify() o notifyAll().**
- ○ **Se requiere proteger los métodos donde se use con el calificador "synchronized".**
- ○ **No se recomienda el uso de sus métodos await(), signal() o signalAll().**
- ✔ ○ **Se pueden utilizar tantos objetos Condition dentro de ese "monitor" como se necesiten.**

Respuesta correcta: D

Preguntas 3 de 5

2.0 Puntos

Se pueden crear muchas variables condición dentro de la clase:

- ◯ **Monitor**
- ✖ ◯ **Condition**
- ◯ **BlockingQueue**
- ◯ **Lock**

Respuesta correcta: D

Preguntas 4 de 5

2.0 Puntos

Si deseamos crear un objeto Condition en un monitor protegido con ReentrantLock en Java...

- ◯ **Debemos usar el constructor de la interfaz Condition().**
- ✔ ◯ **Debemos utilizar el método newCondition() del lock.**
- ◯ **Debemos crearlo dentro del bloque "finally" del constructor del monitor.**
- ◯ **Hay que asegurarse de que no haya otra instancia de la clase Condition declarada dentro del mismo monitor.**

Respuesta correcta: B

Preguntas 5 de 5

2.0 Puntos

Para la utilización de la herramienta ReentrantLock debemos hacer uso de la biblioteca:

- ✔ ◯ **java.util.concurrent**
- ◯ **Ninguna. Son primitivas básicas.**
- ◯ **java.concurrency**
- ◯ **java.locks**

Respuesta correcta: A

sakai.iframe.site

FLIP UD05-2: Colecciones Concurrentes y Variables Atómicas

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

La clase Executor permite:

- ✔ ◯ **Mantener un thread-pool, con un conjunto de hilos ya creados, que se pueden reutilizar.**
- ◯ **Ejecutar monitores de Java sofisticados.**
- ◯ **Especificar la duración de un intervalo de bloqueo.**
- ◯ **Suspender a un grupo de hilos, quedando éstos a la espera de que suceda algún evento.**

Respuesta correcta: A

Preguntas 2 de 5

2.0 Puntos

El uso de variables atómicas permite:

- ◯ **Utilizar la etiqueta synchronized de una forma más eficiente.**
- ◯ **Implementar contadores.**
- ◯ **Utilizar directamente la operación ++ sobre un entero.**
- ✖ ◯ **Dividir la operación c++ en varios pasos que se ejecutan cada uno de forma atómica.**

Respuesta correcta: B

Preguntas 3 de 5

2.0 Puntos

La clase AtomicLong NO ofrece el siguiente método:

- ○ **addAndGet();**
- ○ **decrementAndGet();**
- ✔ ○ **await();**
- ○ **getAndIncrement();**

Respuesta correcta: C

Preguntas 4 de 5

2.0 Puntos

En la interfaz BlockingQueue, el método ..... NO permite extraer elementos de la cola.

- ○ **poll()**
- ○ **remove()**
- ✔ ○ **put()**
- ○ **take()**

Respuesta correcta: C

Preguntas 5 de 5

2.0 Puntos

La operación de incremento atómica de la clase AtomicInteger counter es:

- ○ **getAndIncrement(counter);**
- ✔ ○ **counter.getAndIncrement();**
- ○ **counter ++;**
- ○ **counter.getAndSet(counter++);**

Respuesta correcta: B

Anuncios recientes

sakai.iframe.site

FLIP UD05-3: Semáforos y Barreras

Volver a la Lista de Exámenes

Parte 1 de 1 - 10.0 Puntos

Preguntas 1 de 5

2.0 Puntos

Se puede emplear la clase Semaphore para:
(i) Proteger secciones críticas; (ii) Sincronizar hilos; (iii) Gestionar las políticas de ejecución de los hilos

- ◯ **Sólo (ii)**
- ◯ **Sólo (ii) y (iii)**
- ◯ **Sólo (i) y (ii)**
- ✖ ◯ **Todas las opciones**

Respuesta correcta: C

Preguntas 2 de 5

2.0 Puntos

La herramienta .... permite que un conjunto de hilos se esperen mutuamente en un punto común, siendo reutilizable.

- ✔ ◯ **CyclicBarrier**
- ◯ **Semaphore**
- ◯ **BlockingQueue**
- ◯ **CountdownLatch**

Respuesta correcta: A

Preguntas 3 de 5

2.0 Puntos

Para garantizar cierto orden de ejecución entre dos o más hilos, se puede utilizar un semáforo inicializado a:

- ◯ **Valor 2.**
- ✔ ◯ **Valor 0.**
- ◯ **Valor 3.**
- ◯ **Valor 1.**

Respuesta correcta: B

Preguntas 4 de 5

2.0 Puntos

La herramienta .... no hace uso del método await()

- ◯ **CountdownLatch**
- ◯ **CyclicBarrier**
- ✔ ◯ **Semaphore**
- ◯ **Condition**

Respuesta correcta: C

Preguntas 5 de 5

2.0 Puntos

El método acquire() de la clase Semaphore:

- ✔ ◯ **Espera hasta disponer de un permiso, y entonces lo consume.**
- ◯ **Libera a un hilo esperando el semáforo.**
- ◯ **Adquiere siempre un permiso directamente, sin esperas.**
- ◯ **Añade un permiso.**

Respuesta correcta: A

Anuncios recientes

sakai.iframe.site

Test 1, Grupo C (temas 1,2,3)

Volver a la Lista de Exámenes

Parte 1 de 5 - Ventajas de la programación concurrente. 2.0 Puntos

Preguntas 1 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

La gestión de las comunicaciones resulta más natural y eficaz, pues permite simultanear el uso de la red con otras actividades.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Verdadero

Preguntas 2 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

La depuración es más sencilla que en la programación secuencial.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Preguntas 3 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Permite que en una misma aplicación haya múltiples actividades simultáneas que colaboren entre sí.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Verdadero

Preguntas 4 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Aprovecha mejor los recursos (memoria, comunicación vía red, dispositivos de E/S, múltiples núcleos del procesador, etc.) y así las aplicaciones pueden ser más eficientes.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Verdadero

Parte 2 de 5 - Desventajas de la programación concurrente 2.0 Puntos

Preguntas 5 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

La ejecución paralela de múltiples actividades puede plantear problemas en el acceso a algunos recursos compartidos.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Verdadero

Preguntas 6 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Se necesita que el sistema operativo proporcione un planificador especial del procesador. Muy pocos sistemas operativos son capaces de ofrecerlo.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Preguntas 7 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Repercute negativamente en las prestaciones de la aplicación, pues hay que sincronizar continuamente las diferentes actividades de un programa concurrente.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Preguntas 8 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Todo programa concurrente requiere múltiples ordenadores para ser ejecutado.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Parte 3 de 5 - El problema de la sección crítica 2.0 Puntos

Preguntas 9 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Sólo se puede solucionar correctamente el problema de la sección crítica mediante monitores.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Preguntas 10 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Con una solución correcta al problema de la sección crítica, logramos que el código de la sección crítica se comporte como una “acción atómica”.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Verdadero

Preguntas 11 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Una solución correcta al problema de la sección crítica, tiene que cumplir las propiedades de exclusión mutua y sincronización condicional.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Preguntas 12 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Es imposible lograr una solución correcta al problema de la sección crítica.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Parte 4 de 5 - Preguntas sobre "CuentaBancaria.java" 2.0 Puntos

Ficheros adjuntos

CuentaBancaria.java 0 KB

CuentaBancaria.java

```java
public class CuentaBancaria {
    private long identificador;
    private long saldo;
    public CuentaBancaria(long id){
        identificador = id;
    }
    public void reintegro(long cantidad) {
        saldo -= cantidad;
    }
    public void abono(long cantidad) {
        saldo += cantidad;
    }
    public long saldo() {
        return saldo;
    }
    public long idCuenta() {
        return identificador;
    }
    public void aplicarInteresesMensuales() {
        if (saldo > 50000)
            saldo *= 1.002;
        else if (saldo > 10000)
            saldo *= 1.001;
        saldo -= 2; // También hay comisiones :-(
    }
}
```

Preguntas 13 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

La clase CuentaBancaria es “thread-safe”.

(ver código de CuentaBancaria.java)

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Falso

Preguntas 14 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Los métodos reintegro() y abono() de la clase "CuentaBancaria" deben complementarse con un bucle de espera activa para implantar la sincronización condicional necesaria.

(ver código de CuentaBancaria.java)

✖ ◯ **Verdadero**
◯ **Falso**

Respuesta correcta: Falso

Preguntas 15 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

La clase "CuentaBancaria", implementa correctamente un monitor en la variante Lampson-Redell.

(ver código de CuentaBancaria.java)

✔ ◯ **Verdadero**
◯ **Falso**

Respuesta correcta: Falso

Preguntas 16 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Todos los métodos de la clase "CuentaBancaria" (a excepción del constructor) deberían etiquetarse con “synchronized” para generar una clase libre de condiciones de carrera.
(ver código de CuentaBancaria.java)

✖ ◯ **Verdadero**
◯ **Falso**

Respuesta correcta: Falso

Parte 5 de 5 - Los monitores de Lampson-Redell 2.0 Puntos

Preguntas 17 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Un monitor de tipo Lampson-Redell exige que todo método en el que se use wait() también tenga alguna llamada a notify() sobre esa misma condición, implantando así una reactivación en cascada.

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Falso

Preguntas 18 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

Un monitor de tipo Lampson-Redell suspende (y deja en una cola especial de entrada) al hilo que ha invocado a notify(), activando a uno de los hilos que llamó antes a wait().

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Falso

Preguntas 19 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

En un monitor de tipo Lampson-Redell jamás se suspende al hilo que invoca a notify().

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Verdadero

Preguntas 20 de 20

0.5 Puntos. Puntos descontados por fallo: 0.5

En un monitor de tipo Lampson-Redell la llamada a notify() o notifyAll() debe ser la última sentencia en los métodos del
monitor en los que aparezcan tales llamadas.

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Falso

Anuncios recientes

sakai.iframe.site

Test 2, Grupo C (temas 4,5)

Volver a la Lista de Exámenes

Parte 1 de 4 - Preguntas sobre el código de RW y MainRW 2.5 Puntos

Observe el código de la clase RW y mainRW. La clase RW trata de implementar cerrojos con una semántica adecuada para el problema de los lectores-escritores. Puede tratarse de una implementación correcta o no.

Ficheros adjuntos

```java
import java.util.concurrent.Semaphore;

public class RW {
    private final Semaphore semR = new Semaphore (1);
    private final Semaphore semW = new Semaphore (1);
    private int nr = 0;

    void enterR() throws InterruptedException {
        semR.acquire ();
        if (++nr == 1) semW.acquire();
        semR.release();
    }

    void exitR() throws InterruptedException {
        semR.acquire();
        if (--nr == 0) semW.release();
        semR.release();
    }

    void enterW() throws InterruptedException {
        semW.acquire();
    }

    void exitW() {
        semW.release();
    }
}

```

```java
import java.util.concurrent.*;
public class MainRW {
    static RW rw = new RW(); // Este es mi cerrojo especial.
    static int val = 0;      // Variable global.
    static CyclicBarrier bar = new CyclicBarrier (5, () -> {
        System.out.println ("Valor de val: " + val);
    });

    private static void readOrWrite (String name, int max, boolean reader) {
        try {
            boolean end = false;
            while ( !end ) {
                if (reader) {
                    rw.enterR();
                    if (val >= max) end=true;   // Leemos la variable.
                    rw.exitR();
                } else {
                    rw.enterW();
                    if (++val>= max) end=true;  // Modificamos la variable.
                    rw.exitW();
                }
            }
        } catch (InterruptedException e) {}
        System.out.println ("Hilo " + name + " terminado.");
        try {bar.await ();}catch (Exception e) {} // Usamos la barrera.
    }

    public static void main (String [] args) {
        new Thread ( () -> readOrWrite ("W1", 1000, false)  ).start();  // Escritor W1
        new Thread ( () -> readOrWrite ("W2", 1000, false)  ).start();  // Escritor W2
        new Thread ( () -> readOrWrite ("R1", 1000, true)  ).start();   // Lector R1
        new Thread ( () -> readOrWrite ("R2", 1000, true)  ).start();   // Lector R2
        new Thread ( () -> readOrWrite ("R3", 1000, true)  ).start();   // Lector R3
    }
}
```

Preguntas 1 de 20

0.5 Puntos

Dado el código de la clase RW (puede tomar la clase MainRW como ejemplo de uso), podemos afirmar que la clase RW prioriza a los escritores frente a los lectores, pues en caso de que un escritor esté esperando a escribir, ningún lector podrá leer.

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Falso

Preguntas 2 de 20

0.5 Puntos

Dado el código de la clase RW (puede tomar la clase MainRW como ejemplo de uso), podemos afirmar que la clase RW otorga prioridad a los hilos "lectores" frente a los hilos "escritores".

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Verdadero

Preguntas 3 de 20

0.5 Puntos

La clase RW presenta una implementación correcta de un cerrojo para solucionar el problema de los lectores-escritores, donde varios lectores puedan leer al mismo tiempo, siempre que no haya escritores. Por su parte sólo permite a un escritor acceder en modo escritura.

✖ ◯ **Verdadero**
◯ **Falso**

Respuesta correcta: Verdadero

Preguntas 4 de 20

0.5 Puntos

Dado el código de la clase RW (puede tomar la clase MainRW como ejemplo de uso), podemos afirmar que la clase RW implementa un cerrojo especial que permite a varios lectores acceder de forma concurrente a cierta variable, mientras que se exige exclusión mutua entre escritores y entre escritores con lectores.

✖ ◯ **Verdadero**
◯ **Falso**

Respuesta correcta: Verdadero

Preguntas 5 de 20

0.5 Puntos

Si ejecutamos el programa MainRW, en algún caso se podrán producir condiciones de carrera en el acceso a la variable global "val". Esto es debido a defectos en la implementación de la clase RW.

✖ ◯ **Verdadero**
◯ **Falso**

Respuesta correcta: Falso

Parte 2 de 4 - Preguntas sobre el código de RW y MainRW (2) 2.5 Puntos

Observe el código de la clase RW y mainRW. La clase RW trata de implementar cerrojos con una semántica adecuada para el problema de los lectores-escritores. Puede tratarse de una implementación correcta o no.

Ficheros adjuntos

```java
import java.util.concurrent.Semaphore;

public class RW {
    private final Semaphore semR = new Semaphore (1);
    private final Semaphore semW = new Semaphore (1);
    private int nr = 0;

    void enterR() throws InterruptedException {
        semR.acquire ();
        if (++nr == 1) semW.acquire();
        semR.release();
    }

    void exitR() throws InterruptedException {
        semR.acquire();
        if (--nr == 0) semW.release();
        semR.release();
    }

    void enterW() throws InterruptedException {
        semW.acquire();
    }

    void exitW() {
        semW.release();
    }
}

```

```java
import java.util.concurrent.*;
public class MainRW {
    static RW rw = new RW(); // Este es mi cerrojo especial.
    static int val = 0;      // Variable global.
    static CyclicBarrier bar = new CyclicBarrier (5, () -> {
        System.out.println ("Valor de val: " + val);
    });

    private static void readOrWrite (String name, int max, boolean reader) {
        try {
            boolean end = false;
            while ( !end ) {
                if (reader) {
                    rw.enterR();
                    if (val >= max) end=true;   // Leemos la variable.
                    rw.exitR();
                } else {
                    rw.enterW();
                    if (++val>= max) end=true;  // Modificamos la variable.
                    rw.exitW();
                }
            }
        } catch (InterruptedException e) {}
        System.out.println ("Hilo " + name + " terminado.");
        try {bar.await ();}catch (Exception e) {} // Usamos la barrera.
    }

    public static void main (String [] args) {
        new Thread ( () -> readOrWrite ("W1", 1000, false)  ).start();  // Escritor W1
        new Thread ( () -> readOrWrite ("W2", 1000, false)  ).start();  // Escritor W2
        new Thread ( () -> readOrWrite ("R1", 1000, true)  ).start();   // Lector R1
        new Thread ( () -> readOrWrite ("R2", 1000, true)  ).start();   // Lector R2
        new Thread ( () -> readOrWrite ("R3", 1000, true)  ).start();   // Lector R3
    }
}
```

Preguntas 6 de 20

0.5 Puntos

Al ejecutar el programa MainRW, es posible que el programa no termine. Esto puede ocurrir pues es posible que algún hilo no termine.

- [x] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Falso

Preguntas 7 de 20

0.5 Puntos

Si ejecutamos el programa MainRW, el resultado de la variable "val" al final de la ejecución será siempre 1001. Este resultado lo veremos por la consola tal y como lo imprime el código proporcionado a la barrera.

- [ ] **Verdadero**
- [ ] **Falso**

Respuesta correcta: Verdadero

Preguntas 8 de 20

0.5 Puntos

Al ejecutar el programa MainRW tendremos un máximo de 6 hilos en ejecución al mismo tiempo (incluyendo el hilo main). De estos hilos tendremos 2 escritores y 3 lectores.

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Verdadero

Preguntas 9 de 20

0.5 Puntos

Al ejecutar el programa MainRW, el último hilo en imprimir por la consola "Hilo XX terminado" será siempre un hilo escritor.

- ✔ ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Falso

Preguntas 10 de 20

0.5 Puntos

Al ejecutar el programa MainRW, es posible que tengamos al mismo tiempo 3 hilos consultando el valor de la variable "val".

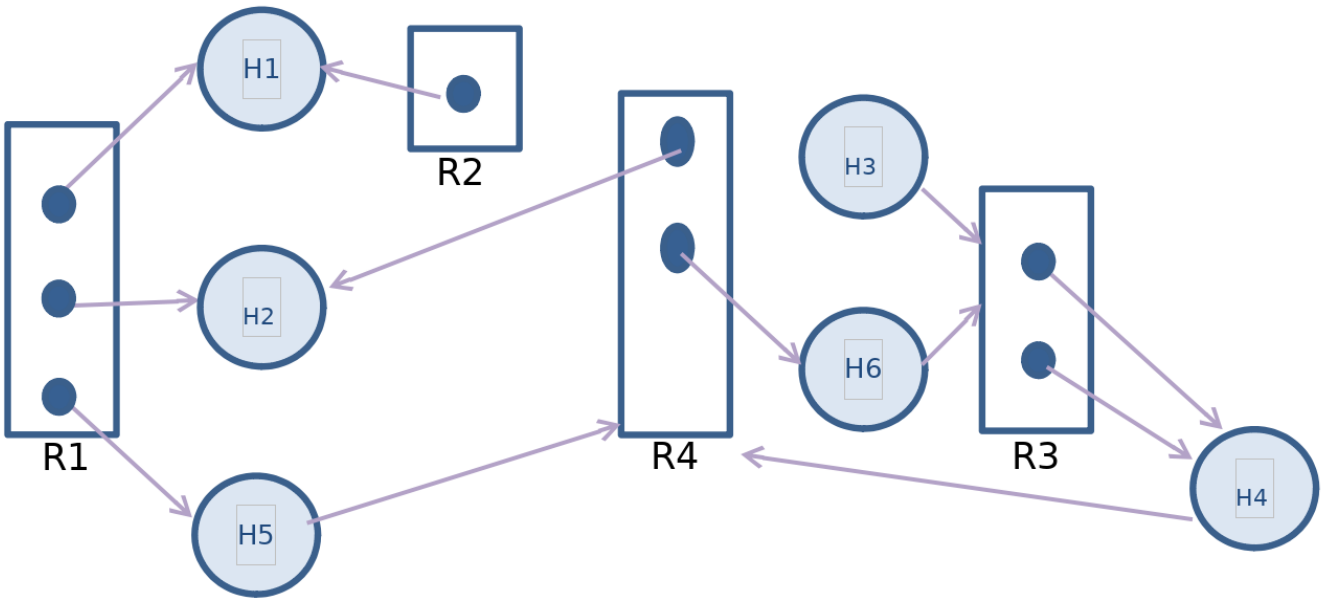
- ○ **Verdadero**
- ○ **Falso**

Respuesta correcta: Verdadero

Parte 3 de 4 - Preguntas sobre un Grafo de Asignación de Recursos. 2.5 Puntos

Observe el grafo adjunto y responda a las preguntas que se formulan.

Ficheros adjuntos



Preguntas 11 de 20

0.5 Puntos

Si H2 pide una nueva instancia del recurso R4, se producirá un interbloqueo.

- ✔ ◯ **Verdadero**
- ◯ **Falso**

Respuesta correcta: Verdadero

Preguntas 12 de 20

0.5 Puntos

Dada la situación de los procesos y recursos mostrados por el grafo y dada su posible evolución posterior, debido a que ciertos procesos terminen o pidan nuevos recursos, el proceso H3 nunca formará parte de un posible interbloqueo, pues no tiene recursos asignados.

- ✖ ◯ **Verdadero**
- ◯ **Falso**

Respuesta correcta: Falso

Preguntas 13 de 20

0.5 Puntos

Toda secuencia segura que podamos encontrar en el grafo empezará por el proceso H1 o por H2.

- ✔ ◯ **Verdadero**
- ◯ **Falso**

Respuesta correcta: Verdadero

Preguntas 14 de 20

0.5 Puntos

Dado el grafo adjunto, observamos que se cumplen todas las condiciones de Coffman

- ✔ ◯ **Verdadero**
- ◯ **Falso**

Respuesta correcta: Verdadero

Preguntas 15 de 20

0.5 Puntos

Dado el grafo, deducimos que el proceso H4 forma parte de un interbloqueo.

- ✖ ◯ **Verdadero**
- ◯ **Falso**

Respuesta correcta: Falso

Parte 4 de 4 - Preguntas sobre diversos mecanismos de sincronización entre hilos. 2.5 Puntos

Preguntas 16 de 20

0.5 Puntos

Utilizando semáforos podemos implementar monitores, barreras de tipo CyclicBarrier y barreras de tipo CountDownLatch.

✖ ○ **Verdadero**
○ **Falso**

Respuesta correcta: Verdadero

Preguntas 17 de 20

0.5 Puntos

Mediente una instancia de AtomicReference podemos convertir en Thread-safe cualquier instancia de cualquier objeto Java.

✖ ○ **Verdadero**
○ **Falso**

Respuesta correcta: Falso

Preguntas 18 de 20

0.5 Puntos

Suponga que creamos una barrera cíclica, indicando como argumento 4. Si tenemos 2 hilos que utilizan la barrera, y cada hilo llama al método "await" de la barrera 2 veces, la barrera se abrirá en la cuarta llamada a await, sin importar que realmente no tenemos 4 hilos, sino únicamente 2.

✖ ○ **Verdadero**
○ **Falso**

Respuesta correcta: Verdadero

Preguntas 19 de 20

0.5 Puntos

Si 2 hilos comparten 2 semáforos, se podrá producir un interbloqueo entre ellos, dependiendo del valor de los semáforos, y de las llamadas a acquire/release que cada uno de ellos haga.

✔ ○ **Verdadero**
○ **Falso**

Respuesta correcta: Verdadero

Preguntas 20 de 20

0.5 Puntos

Los semáforos pueden inicializarse con un valor positivo o negativo.

✔ ○ **Verdadero**
○ **Falso**

Respuesta correcta: Verdadero