# PROGRAMACIÓN CONCURRENTE Y DISTRIBUIDA
# BOLETÍN DE PRÁCTICAS

## CONVOCATORIA DE JUNIO
## CURSO 2022-2023

# Contenido

# Entregables

La entrega constará de un único proyecto conteniendo 4 paquetes (uno para cada ejercicio) que se comprimirá en un fichero .zip y una memoria de prácticas en formato PDF. Los ejercicios se implementarán en Java 8 o una versión superior usando la codificación de caracteres UTF-8 (Project-Resource-Textfileencoding-Other-UTF-8). **Tanto el código fuente como la memoria se entregarán a través de una tarea que será creada en el AULA VIRTUAL. No se admitirán entregas realizadas por otros medios.**

En la memoria escrita **se deben incluir necesariamente para cada ejercicio** los siguientes apartados:

- **Recursos no compartibles.** Los que se hayan identificado en el problema.
- **Condiciones de sincronización.** Las detectadas en el problema.
- **Pseudocódigo.** El de los diferentes hilos y monitores que intervengan en la solución.
- **Cuestiones planteadas en los ejercicios.**

# Formato de los documentos

La memoria de prácticas deberá estar escrita correctamente en sus aspectos ortográfico y gramatical, evitando el uso de vulgarismos y lenguaje SMS. Las páginas deberán estar numeradas y se debe incluir un índice. El tipo de letra utilizado para el texto normal será Times New Román (o semejante) en 12 puntos y en color negro. El interlineado será sencillo.

Los listados de código deberán estar formateados cuidando varios aspectos:

- Indentación.
- Tamaño de la letra adecuado para que no se corten los renglones, o cuando esto ocurra el código siga siendo legible y ordenado.
- Tipo de letra de paso fijo (Courier o semejante) en color negro.

Además, el código fuente entregado debe estar debidamente comentado tanto en los bloques de código más relevantes en relación con la programación concurrente y distribuida como mediante el uso de Javadoc para las clases y métodos desarrollados.

En la portada de la memoria de prácticas deberá aparecer arriba a la derecha la siguiente información: Apellidos, Nombre, Asignatura, Grupo y Subgrupo de Prácticas, Convocatoria, Año académico y Profesor.

# Evaluación

La evaluación constará de dos partes principales: la corrección de las prácticas entregadas y la defensa de las mismas en una entrevista personal. Las prácticas se realizarán en parejas.

Como criterios de evaluación se tendrán en cuenta los siguientes:

- Corrección de las soluciones aportadas.
- Grado de concurrencia
- Entrevista de prácticas
- Claridad en la explicación (memoria de prácticas)
- Originalidad.

La copia de prácticas será motivo de suspenso para todos los participantes implicados.

La no asistencia (injustificada) a la entrevista de prácticas será motivo de suspenso de las mismas.

Este boletín se compone de cuatro ejercicios que serán valorados por el profesor con una puntuación de 0 a 10 cada uno. Para superar la parte práctica será necesario sacar una nota igual o superior a 5 en cada uno de ellos. Cuando todos los ejercicios estén superados la ponderación aplicada será la siguiente: 0.25*E1 + 0.25*E2 + 0.25*E3 + 0.25*E4. La fecha límite para la entrega del boletín de prácticas será **el 07 mayo de 2023 a las 23:00 horas.**

# Ejercicios

## *Ejercicio 1: Introducción a la programación concurrente.*

Supongamos dos hilos con la siguiente funcionalidad: uno se dedica a generar una matriz A de valores enteros de tamaño 3x3, calcular el cuadrado de dicha matriz ($A^2$) e imprimir en pantalla:

**A x A**

**a1 a2 a3 a1 a2 a3**

**a4 a5 a6 x a4 a5 a6**

**a7 a8 a9 a7 a8 a9**

**$A^2$**

**r1 r2 r3**

**r4 r5 r6**

**r7 r8 r9**

El otro hilo hace lo mismo pero en lugar de calcular $A^2$, calcula A+A y por lo tanto imprime en pantalla:

**A + A**

**a1 a2 a3 a1 a2 a3**

**a4 a5 a6 + a4 a5 a6**

**a7 a8 a9 a7 a8 a9**

**2A**

**r1 r2 r3**

**r4 r5 r6**

**r7 r8 r9**

Para que se impriman de forma lenta, entre cada línea, usaremos el método Sleep(1000) de Java.

El hilo principal, una vez hayan finalizado todos los hilos, imprimirá en pantalla: "FIN DEL PROGRAMA".

La actividad anterior la repite cada hilo 10 veces. Desarrollar un programa concurrente en Java que resuelva el problema anterior. Para resolver la sincronización necesaria entre los hilos se usarán los cerrojos de la clase ReentrantLock.

a) (1 punto sobre 10) Antes de usar la clase ReentrantLock en el problema comprueba y explica cómo es la salida en pantalla y qué problemas observas.

b) (1 punto sobre 10) Después de usar la clase ReentrantLock en el problema para corregir la salida, comprueba y explica cómo es esta salida en pantalla.

c) (1 punto sobre 10) Si tras usar ReentrantLock eliminas el uso de Sleep(), el orden en el que los hilos usan la pantalla repitiendo la ejecución varias veces ¿es siempre el mismo? ¿es correcto que ese orden varíe en cada ejecución? Justifica la respuesta.

d) (1 punto sobre 10) Indica qué acciones puedes llevar a cabo para comprobar que no se producen problemas debido a la concurrencia en el programa desarrollado, por ejemplo, para forzar distintas mezclas de los códigos de los procesos. Justifica la respuesta.

## *Ejercicio 2: Semáforos.*

Supongamos que en el sistema tenemos 5 instancias de la clase Panel que se proporciona implementada en Java. Una instancia de esta clase permite visualizar información en pantalla en una ventana del tamaño especificado en el constructor de la clase. Queremos que dichos paneles sean compartidos por 50 hilos para visualizar el resultado de una operación matemática cualquiera entre 2 números generados aleatoriamente. El uso de los paneles por parte de los hilos está sujeto a la siguiente restricción: el panel 1 sólo se puede usar para visualizar aquellas operaciones cuyo resultado sea un múltiplo de 5, los paneles 2 y 3 se utilizan cuando el resultado es impar y no es múltiplo de 5, y los otros 2 paneles para visualizar los resultados cuando el número resultante es par y no múltiplo de 5.

Un hilo repetirá 30 veces lo siguiente: generará de forma aleatoria dos números, después realizará la operación matemática y finalmente tendrá que imprimir el resultado en el panel adecuado. Los paneles deberán gestionarse de forma eficiente.

La impresión de cada hilo en el panel correspondiente será la siguiente:

**“Hilo con ID -----**

**Números generados X Y**

**Operación a realizar: -----**

**Resultado: Z**

**Fin hilo ID -----”**

Desarrollar un programa concurrente en Java para resolver el problema anterior usando semáforos como mecanismo para la sincronización.

Según las condiciones de sincronización del problema,

a) (0.5 puntos sobre 10) ¿Qué acciones pueden realizar simultáneamente los hilos?

b) (0.5 puntos sobre 10) Explica el papel de los semáforos que has usado para resolver el problema.

c) (0.5 puntos sobre 10) ¿Puede haber varios hilos escribiendo en pantalla simultáneamente? Justifica tu respuesta

## *Ejercicio 3: Monitores*

Se pretende simular mediante un programa concurrente la actividad de una gasolinera con túnel de lavado al que una serie de clientes van a lavar el coche y, a veces, también a repostar. Para repostar, la gasolinera dispone de 4 surtidores de gasolina que trabajan simultáneamente. Igualmente, para lavar el coche, la gasolinera dispone de 5 túneles de lavado. Los clientes que van a repostar lo hacen ANTES de lavar el coche.

Vamos a suponer que un cliente entra a la gasolinera. Algunos clientes deciden repostar en uno de los 4 surtidores en orden de llegada a la gasolinera. Esos coches repostan una cantidad aleatoria de entre 20 a 50 euros que dura un tiempo X. Ese tiempo de repostar X es directamente proporcional a la gasolina repostada.

Igualmente, los coches pueden seleccionar uno de los 3 tipos de lavado: básico, normal o premium que dura un tiempo Y, que oscila entre valores: B, N y P, correspondientes a los 3 tipos de lavado. Cualquier tipo de lavado puede realizarse en cualquiera de los 5 túneles de lavado disponibles.

Los coches que repostan, cuando terminan de repostar, se dirigen al túnel de lavado cuyo tiempo de espera estimado sea menor, **en ese momento**, de acuerdo con el valor Y de los clientes que ya están esperando para lavar el coche en las colas de lavado.

Los coches que solo van a la gasolinera para lavar el coche **solo pueden hacerlo en el quinto túnel de lavado**. Los coches que han repostado previamente pueden ir a cualquiera de los cinco túneles de lavado, y lo hacen, como se acaba de mencionar, al que tenga la cola de espera más breve en tiempo (no a la más corta en número de vehículos, sino a la de menor tiempo de espera teniendo en cuenta los tiempos de lavado de los coches que están en esa cola, que pueden ser diferentes) en el momento en que se dirigen a los túneles de lavado (tras repostar).

Usando monitores para la sincronización, haremos una simulación donde habrá 100 hilos que modelen el comportamiento de los coches con el valor X e Y inicializados de forma aleatoria. Con probabilidad del 30% un coche irá sólo a lavar el coche y no a repostar.

La impresión en pantalla por parte de los hilos clientes justo antes de ponerse en la cola para lavar el coche elegida debe tener el siguiente formato para los clientes que reposten y laven:

**"Cliente id ha sido atendido en el surtidor ---**
**Repostado --- euros en un tiempo X**
**Tiempo estimado de lavado Y**
**Seleccionado túnel ----**
**Tiempo estimado de espera para el lavado en la cola1=----, cola2=-----, cola3=-----, cola4=-----, cola5=-----"**

donde id, -----, X, Y, son valores particulares para cada hilo. Los valores de cola1, cola2 y

cola3, cola 4, cola 5 son los valores que le llevan a hilo a tomar su decisión, es decir, no incluyen su propio tiempo. Los tiempos X e Y para simular las acciones de repostar y lavar el coche se implementarán mediante el método sleep() de Java.
La impresión por pantalla no debe hacerse dentro del monitor.

Para los coches que solo usan el servicio de lavado la impresión es la siguiente:

**"Cliente id solo quiere lavar el coche**
**Tiempo estimado de lavado Y**
**Tiempo estimado de espera para el lavado en la cola5=-----"**

La sincronización hay que resolverla con monitores de la forma que consideres más eficiente.
a) (0.5 puntos sobre 10) ¿Qué tipo de monitor Java has usado?. Justifica la respuesta.
b) (0.5 puntos sobre 10) ¿Cómo resuelves la exclusión mutua de la pantalla?
c) (0.5 puntos sobre 10)¿Cuántas variables Condition necesitas y para qué?

## *Ejercicio 4 – Paso de Mensajes*

Se pretende implementar un sistema de votación teniendo en cuenta que:

- Los usuarios, para poder votar, deben registrarse en un Servidor.
- Sólo se permite el registro de hasta 100 usuarios en el sistema.
- Los usuarios pueden obtener del servidor en cualquier momento el listado de candidatos a votar.
- Cuando se registran obtienen un token de autenticación que les permitirá posteriormente votar (aleatoriamente) a uno de los candidatos.
- Cuando un usuario vota a un candidato (presentando el token), el servidor debe comprobar que el usuario se haya registrado previamente.
- El periodo de votación comienza sólo cuando hay más de 50 usuarios registrados.
- Se puede registrar un nuevo usuario estando la votación ya abierta.
- Con cierta probabilidad un usuario que se registra no vota.

Habrá un proceso especial **Admin** que contacta con el **Servidor** para cerrar la votación (transcurrido un tiempo C aleatorio desde el comienzo de la votación). En cualquier caso, el Servidor cerrará la votación si transcurrido un tiempo T no se recibe ningún mensaje de ningún tipo de ningún usuario.

El servidor mostrará por pantalla un mensaje cuando comience la votación. Cuando cierra la votación, el Servidor indicará por pantalla el resultado. Si la votación es valida mostrará por pantalla el candidato ganador. Para que una elección sea válida (una vez terminado el periodo de votación) deben haber votado al menos 25 personas (de los registrados). Si la votación no es valida lo informará también por pantalla al finalizar la votación.

Los usuarios imprimirán por pantalla lo siguiente dependiendo de la situación (no son excluyentes):

- Si se han podido registrar :

  *"Soy el usuario 1 y me he registrado correctamente, y este es mi token: XXXXX"*

- Si no se han podido registrar:

  *"Soy el usuario 1 y no me he podido registrar".*

- Si han conseguido votar:

  *"Soy el usuario 1 y he votado a YYYYY"*

- Si no han conseguido votar:

  *"Soy el usuario 1 y no han admitido mi voto"*

- Si han decidido no votar:

  *"Soy el usuario 1 y he decidido abstenerme de votar"*

La impresión en pantalla debe hacerse en exclusión mutua.

Se debe implementar siguiendo un mecanismo de paso de mensajes asíncrono con buzones.

Se deben lanzar 120 hilos que realicen la funcionalidad de 120 usuarios, además del hilo Servidor y el hilo Admin.

Preguntas:

a) 0.5 puntos Dado que no se pueden usar variables compartidas al resolver el problema, ¿cómo has resuelto que los usuarios puedan conocer la lista de candidatos?

b) 0.5 puntos ¿Cómo resuelves la exclusión mutua de la pantalla?

c) 0.5 puntos ¿Cómo has resuelto el que un usuario se informe de si se ha admitido su voto (votación abierta) o no (votación cerrada)?