ALUMNO:

Responder a las siguientes preguntas (0,3 puntos cada pregunta)

1.- ¿Cómo podemos determinar cuando termina un hilo?
2.- ¿Cuales son las herramientas para manejar la concurrencia?
3.- Que objetivo tiene la planificación de hilos
4.- ¿Cuál es la forma de comunicarse principalmente los hilos?
5.- ¿Cuando se produce una inconsistencia de memoria?
6.- ¿Cuales son los posibles estados de un hilo?
7.- ¿Cuales son las operaciones básicas que se pueden realizar con un hilo?
8.- ¿Qué indica el valor negativo de un semáforo?
9.- ¿Cuales son los problemas derivados de la sincronización?
10.- ¿Cuales son los mecanismos de sincronización?

Ejercicio 1 (2,5 puntos)

Crea la clase **MatarHilo** que extienda a Thread. Esta clase tiene dos métodos :

- **run()** : que preguntará si el hilo está parado y mientras el hilo esté lanzado , escribirá un mensaje.
- **main()**: Creará un objeto de la clase MatarHilo y lo lanzará , esperará un tiempo sin hacer nada y transcurrido este tiempo matará al hilo invocando al método pararHilo().

Ejercicio 2 (Bloqueo de hilos) (3 puntos)

Este ejercicio estará compuesto por tres clases:

- Clase ObjetoCompartido
- Clase BloqueoHilo
- Clase HiloCadena

1. Clase **ObjetoCompartido** , esta clase contiene el método "**PintaCadena()**" que recibe un String y lo muestra por pantalla.
2. Clase **BloqueoHilos** , esta clase contiene el main() , en el que se crea un objeto de la clase ObjetoCompartido que será compartido por tres hilos de la clase HiloCadena. Los hilos usarán el método del objeto compartido para mostrar por pantalla una cadena que será enviada al crear el hilo (new HiloCadena(objetoCompartido, cadena)).
3. Clase HiloCadena que extiende a Thread. Contiene el método run() en el que se invoca al método PintaCadena() del objeto compartido dentro de un bucle for.

El objetivo es mostrar de forma alternativa dos cadenas que inicializa cada uno de los tres hilos que que la salida generada deberá ser:

Ejecutando hilo 1
Ejecutando hilo 2
Ejecutando hilo 3
Ejecutando hilo 1
. . . . . .
Realizarlo mediante bloques sincronizados.

Para mantener una cierta coordinación entre los hilos, usar los métodos wait(), notify() y notifyAll() dentro del bloque sincronizado.

Ejercicio 3 (2,5 puntos)

Este programa tendrá la clase EjemploInterrupciones que tendrá los siguientes métodos:

- **run()**: Que testeará si el hilo está interrumpido y en caso contrario escribirá un mensaje y esperará un tiempo. Este método debe de capturar la interrupción InterruptedException , escribiendo el mensaje "HA OCURRIDO UNA EXCEPCIÓN" cuando el hilo finalice deberá escribir el mensaje "FIN HILO".
- **Interrumpir()** : cuando se llame a este método, se tiene que interrumpir la ejecución del hilo.
- **main()** : creará un objeto de la clase EjemploInterrupciones , lanzará el hilo , esperará un tiempo sin hacer nada e interrumpirá el hilo, cuando ocurra esto se debe lanzar la interrupción "InterruptedException", que informará del lanzamiento de la interrupción

---

1.-**¿Cómo podemos determinar cuando termina un hilo?**

El hilo puede indicarlo o puede ser el proceso el que lo finalice mediante la llamada correspondiente.

2.- **¿Cuales son las herramientas para manejar la concurrencia?**

(Un programa concurrente define un conjunto de acciones que pueden ser ejecutadas simultáneamente.)

Los Hilos. Ejecución del código del proceso

3.- **Que objetivo tiene la planificación de hilos**

Cuando se trabaja con varios hilos, en ocasiones es necesario pensar en la planificación de threads, para asegurarse de que cada hilo tiene una oportunidad justa de ejecutarse.
El planificador es quien determina qué proceso es el que se ejecuta en un determinado momento en el procesador, el hilo que se ejecutará estará en función del número de núcleos disponibles y del algoritmo de planificación que se esté utilizando.

4.- **¿Cuál es la forma de comunicarse principalmente los hilos?**

Los métodos wait(), notify() y notifyAll() son parte de todos los objetos porque están implementados por la clase Object. Estos métodos solo deben invocarse desde un contexto sincronizado.

5.- **¿Cuando se produce una inconsistencia de memoria?**

se produce cuando diferentes hilos tienen una visión diferente de lo
que debería ser el mismo dato. Las causas de los errores de coherencia son complejas y van desde la no liberación de datos obsoletos hasta el desbordamiento del buffer entre otros.

6.- **¿Cuales son los posibles estados de un hilo?**

•Nuevo: el hilo está preparado para su ejecución pero todavía no se ha realizado la llamada correspondiente en la ejecución del código del programa. Los hilos se inicialilzan en la creación del proceso correspondiente, ya que forman parte de su espacio de memoria pero no empiezan a ejecutar hasta que el proceso lo indica.

• Listo: el proceso no se encuentra en ejecución aunque está preparado para hacerlo. El sistema operativo no le ha asignado todavía un procesador para ejecutarse . El planificador del sistema operativo es el encargado de elegir cuándo el proceso pasa a ejecutarse.

• Pudiendo ejecutar (Runnable): el hilo está preparado para ejecutarse y puede estar ejecutandose. A todos los efectos sería como si el thread estuviera en ejecución, pero debido al número limidtado de núcleos no se puede saber si se está ejecutando o esperando debido a que no hay hardware suficiente para hacerlo.

Para simplificar, se puede considerar que todos los threads del proceso se ejecutan al mismo tiempo (en paralelo) sabiendo que los hilos deben compartir los recursos del sistema.

• Bloqueado: el hilo está bloqueado por diversos motivos (esperando por una operación de E/S, sincronización con otros hilos, dormido, suspendido) esperando que el suceso suceda para volver al estado Runnable. Cuando ocurre el evento que lo desbloquea, el hilo pasaría directamente a ejecutarse.

• Terminado: el hilo ha finalizado su ejecución. Sin embargo, frente a los procesos que liberan los recursos que mantenían cuando finalizan, el hilo no libera ningún recurso ya que pertenecen al proceso y no a él mismo.

Para terminar un hilo, él mismo puede indicarlo o puede ser el propio proceso el que lo finalice mediante la llamada correspondiente.

7.- **¿Cuales son las operaciones básicas que se pueden realizar con un hilo?**

Creación y arranque de hilos, espera de hilos e interrupción.

8.- **¿Qué indica el valor negativo de un semáforo?**

Que hay procesos en estado bloqueado por lo que despertará a uno de ellos obteniéndolo en cola.

9.- **¿Cuales son los problemas derivados de la sincronización?**

Cuando un hilo invoca un método sincronizado, adquiere automáticamente el monitor que el sistema crea específicamente para todo el objeto que contiene ese método. Solo lo libera cuando el método finaliza o si lanza una excepción no capturada. Ningún otro hilo podrá ejecutar ningún método sincronizado del mismo objeto mientras el monitor de ese objeto no sea liberado, es decir , el cerrojo está cerrado. Esto implica que se bloqueen los métodos que afectan a los recursos compartidos del objeto (indicados son synchronized).

10.- **¿Cuales son los mecanismos de sincronización?**

Las sentencias sincronizadas, las condiciones y las clases Object.