Prácticas Concurrencia y Distribución (18/19)

Arno Formella, Anália García Lourenço, Hugo López Fernández, David Olivieri

semana 22 febrero – 26 febrero

2. Semana 5 (22/03–26/03): Sincronización y Exclusión

1. **(P4: para entregar en grupo de práctica)**:
   Objetos compartidos y acceso sincronizado: Objetivo: este ejercicio pretende reiterar el concepto de compartir objetos entre hilos y utilizar la referencia del objeto compartido para controlar el acceso de subprocesos a las secciones críticas. **Configuración del problema:**

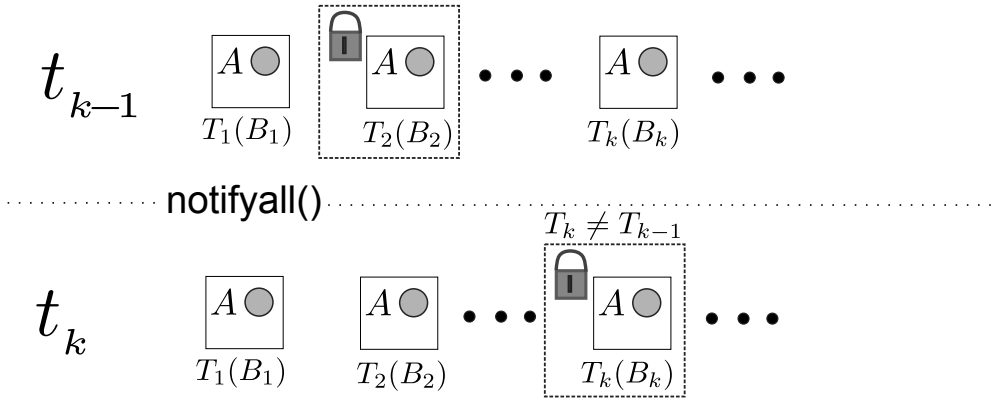
   a) **Class A:** Implementa una clase `ClassA` que contenga un solo método `EnterAndWait()`. Este método debe hacer lo siguiente, en este orden:
      1) imprimir un mensaje indicando cual es el hilo que está comenzando a ejecutarlo;
      2) luego se detendrá durante unos segundos; y
      3) vuelve a imprimir otro mensaje indicando el hilo que está acabando de ejecutar el método.

   b) **Class B:** Implementa una clase `ClassB` que implemente `Runnable` y que se construya recibiendo como parámetro un objeto de la clase `ClassA`. Haz que en el método `run()` simplemente llame al método `enterAndWait()` del objeto con el que ha sido construido.

   c) **Main:** Implementa una clase Main con un método principal en el que se crea un único objeto de la clase `ClassA` y varios objetos de la clase `ClassB` a los que se les pasa a todos como parámetro el objeto de la clase `ClassA`. Después se crearán y ejecutarán el mismo número de hilos (objetos de la clase `Thread`) que de objetos de tipo `ClassB` tengamos pasándoles como parámetros los objetos de la clase `ClassB`, de forma que cada método `run()` de cada objeto de clase `ClassB` se ejecute en un hilo diferente.

   d) **Análisis:** ¿Cuál es el resultado? ¿Cuántos hilos pueden estar simultáneamente ejecutando el método `enterAndWait()`?

   e) **Acceso limitado a la sección crítica:** Utilizando sincronización (el mecanismo que vimos la semana pasada), modifica el código para que solo un hilo pueda estar ejecutando el método `enterAndWait()` en cualquier instante. Explica lo que observas.



2. **(P4: Para empezar en grupo de práctica)**: mecanismo de sincronización `wait()` y `notify()`/`notifyAll`

   Objetivo: aprender a sincronizar hilos utilizando los mecanismos `wait()` y `notify()`/`notifyAll()`.

   Reutiliza el código del problema anterior con las siguientes modificaciones:

   a) **Modifica ClassA:** agrega un atributo de tipo entero llamado counter. Este contador se debe de inicializar en el constructor de la clase y cada vez que se invoque al método `enterAndWait()` se debe disminuir en una unidad. Crea también un método `isFinished()` que devuelva verdadero si el valor del counter es 0 y falso en caso contrario. Agrega otro atributo de tipo `Set<Long>`llamado `threadIds` en el que se almacenen los identificadores (`Thread.currentThread().getId()`) de los hilos que han ejecutado el método `enterAndWait()`. Crea también un método que permita recuperar este conjunto.

   b) **Modifica ClassB:** Modifica el método `run()` implementar la condición de exclusión condicional con `wait()` y `notify()`/`notifyAll()`. Los hilos deben ejecutarse continuamente hasta que el objeto de la `ClassA` compartido indique que ha finalizado. Básicamente, ahora, cada hilo debe de hacer lo siguiente: llamar al método `wait()` del objeto compartido y esperar a ser despertado para continuar la ejecución. Cuando es despertado, debe comprobar si todavía es posible ejecutar en el `enterAndWait()` y, en ese caso, ejecutarlo. Antes de ponerse a dormir de nuevo, el hilo debe avisar al siguiente hilo con `notify()`/`notifyAll()`.

   c) **Modifica el programa principal:** modifica el código del método principal de manera apropiada para que sea posible ejecutar el problema con con `M` hilos y `N` accesos totales a la sección crítica de `enterAndWait()`. Prueba distintas configuraciones (variando `M` y `N`, variando la utilización de `notify()`/`notifyAll()`) y comprueba que funciona como se espera. ¿Cuál es el valor del atributo counter del objeto de tipo ClassA compartido al finalizar la ejecución de todos los hilos?

   d) **Comprueba si todos los hilos ejecutan el método `enterAndWait()`:** utilizando la lista de identificadores que han pasado por el método `enterAndWait()` que tiene el objeto de clase `ClassA`, añade una comprobación al final del método principal para saber si todos los hilos creados ejecutan dicho método. Recuerda probar distintas configuraciones (variando `M` y `N`, variando la utilización de `notify()`/`notifyAll()`).

3. **(P3: para entregar dentro de una semana):** Exclusión condicional en el acceso a secciones críticas.

   Objetivo: El propósito de este ejercicio es comprender cómo controlar el acceso de hilos a secciones críticas. En los problemas anteriores todos los hilos compiten continuamente para entrar y es el sistema

operativo el que finalmente decide cual es el siguiente hilo. Aquí queremos controlar ese acceso de forma explícita dadas ciertas condiciones adicionales.

En este problema, tu programa debería controlar los hilos basándose en una condición adicional: su color. Se tiene que garantizar que dos hilos del mismo color no pueden ejecutar consecutivamente el método `enterAndWait()`.

Reutiliza el código del problema anterior con las siguientes modificaciones:

a) **Modifica ClassB:** Haz que los objetos de la clase `ClassB` reciban también en su constructor un objeto de tipo String llamado `color` y que lo guarden en un atributo.

b) **Modifica el programa principal:** modifica el programa principal para que asigne a los objetos de `ClassB` que crea un color al azar (el color puede tomar los valores "rojo" o "negro").

c) **Modifica ClassA y ClassB:** finalmente, modifica estas dos clases para que se seleccione al siguiente hilo cumpliendo la regla de que el color del hilo debe alternar entre las iteraciones (por ejemplo, si el hilo anterior fue rojo, el siguiente hilo con acceso a la sección crítica debe ser negro). Una posibilidad para hacer esto es almacenar en el objeto `ClassA` el color del último hilo que ha ejecutado el método `enterAndWait()`. Los hilos podrían obtener este valor para saber si, una vez que están despiertos, tienen autorización para ejecutar el método.