Evaluable Concurrencia y Distribución (22/23)

Arno Formella, Alba Nogueira Rodríguez, David Ruano Ordás

semana 6 marzo – 10 marzo

Evaluable 1: Gestión de interrupciones de hilos

Objetivos: Demostrar los conocimientos adquiridos sobre interrupciones de hilos en java.

En las primeras tres prácticas aprendimos crear y lanzar hilos, medir el tiempo empleado en sus métodos run (estilo *carrera de San Martiño*), y sincronizar los hilos al final del programa con el objetivo de que el hilo principal siempre sea el último en terminar el programa.

En este evaluable creamos una lógica para interrumpir el hilo principal cuando esté esperando en el bucle final a todos sus trabajadores, ya que este bucle sería un bucle infinito si algún hilo trabajador no termine su metodo `run()`.

1. Cambia la tarea de los hilos de las prácticas pasadas por un simple bucle `do-while` con una condición booleana generada al azar (p.ej. un número aleatoria flotante entre 0 y 1 es menor o mayor que 0.5,) que o bien es un bucle infinito o bien es un bucle de una sola vuelta. Dentro del bucle realiza un `sleep` de un segundo. Cuando captas una posible interrupción de tal `sleep(...)` sal del bucle con un `break`.
2. El hilo principal, en principio, actua como hasta ahora:
   - crea cierto número de hilos, cuya cantidad se para via línea de comando,
   - guarda todas las referencias en una estructura de un array adecuada,
   - lanza todos los hilos para que ejecuten sus métodos `run()` mencionados arriba,
   - y espera en un bucle final con los `join()` correspondientes para que terminen todos los hilos.
3. Si ejecutas el hilo principal del programa, observarás que, dependiendo si algún hilo al final realiza un bucle infinito o no, se quedará también esperando sin acabar.
4. Por eso contestamos la última pregunta de la semana pasada: ¿Es posible captar una interrupción producida por un hilo trabajador en el hilo principal? Para eso:

5. Crea una clase `Guardian` que extiende de `Thread`, que
   - recibe la referencia al hilo principal (busca información como obtener tal referencia),
   - y ejecuta en su método `run()` una espera de unos 10 segundos y a acabar interrumpe el hilo principal.
   - Crea y lanza tal guardián en el programa principal en un sitio adecuado.

   El hilo principal saldrá del bucle (que contiene los `join()`) por interrupción (si el temporizador del guardián haya terminando) y puede mandar ahora interrupciones a todos los hilos que todavía están ejecutando su bucle infinito.
6. Para terminar de nuevo el programa correctamente con el hilo principal, ejecuta otra vez un bucle que espera a los hilos restantes (otra vez con los `join()`).
7. Para completar: si todos los hilos trabajadores, por suerte, no entran en un bucle infinito, te debes preocupar que el guardián no terminará después del hilo principal.

Método de evaluación:

1. Cada estudiante, de forma individual, realiza la tarea de programación durante las dos horas de prácticas.
2. Cada estudiante al final de su turno sube a la plataforma el programa tal como conseguido durante las dos horas.
3. Cada estudiante puede terminar su programa en las horas no presenciales hasta la semana que viene que correspondan a su grupo. (Se abrirá un apartado correspondiente en la plataforma.)
4. Cada estudiante evalúa su propia entrega con un informe corto—se dará una posible solución que se proporcionará después de los plazos de entregas—y sube tal informe con la autoevaluación (con una nota entre 0 y 10) a la plataforma.

Observa:

- Sí falta alguno de los pasos o se detecta que no se ha hecho de forma individual o con fraude, la nota será un cero.
- En caso contrario los informes contarán en el apartado (P3) de la evaluación continua, y la propia solución presentado en el apartado (P4) de la evaluación continua.
- La nota de autoevaluación no es determinante, pero sí si la autoevaluación y el informe están bien hechos.