Prácticas Concurrencia y Distribución (16/17)

Arno Formella, Francisco Rodríguez Martínez,
Javier Rodeiro Iglesias, David Olivieri

semana 30 enero – 3 febrero

Las prácticas de este curso están organizados en 5 actividades principales que están a su vez divididos en apartados que se deben realizar **semanalmente** en grupos de **una** o **dos** personas con entregas **individuales** al final de las horas de clase práctica.

Las entregas se realizan mediante la herramienta FaiTIC que estará configurada para permitir la subida de ficheros durante la segunda mitad de la clase práctica. Una entrega sirve al mismo tiempo como **testigo de asistencia** a clases prácticas. No obstante el profesor de prácticas puede usar otras medidas adicionales para monitorizar dicha asistencia.

Las entregas consisten en la subida de un **único** fichero simple o de tipo archivo (`.zip`, `.rar`, `.tgz`, etc.). Dichos ficheros **siempre** tendrán nombres que se forman de la siguiente manera:

`Apellido1_Apellido2_Nombre_Grupo_Fecha.Extensión`

donde el `Apellido1`, `Apellido2`, y `Nombre` se entienden como tales, el `Grupo` es uno de `CDI1` hasta `CDI5`, la fecha se indica en formato mes y día (MMDD), es decir, el 30 de enero será: 0130, y la extensión según tipo de fichero. Es decir, si Fran sube un fichero java el lunes, 30 de enero de 2017, este fichero se llamaría:

`Rodríguez_Martínez_Francisco_Javier_CD3_0130.java`

Ficheros con nombres que no cumplen con esta nomenclatura serán simplemente **ignorados**.

1. Actividad: Introducción a la concurrencia en Java

Objetivos: Adquirir conocimientos básicos sobre la forma como está implementada la concurrencia en Java, los métodos básicos para crear hilos, uso de esperas programadas, medición de tiempo de ejecución, sincronización simple.

Metodología: Esta actividad (que sigue con más tareas en las siguientes semanas) se realizará en horas de prácticas presenciales y en horas no presenciales. Al tratarse de la primera parte de esta actividad, además de las tareas descritas, se llevarán a cabo tareas de familiarización con las herramientas, la documentación de Java y métodos de trabajo.

Material adicional: Son de especial interés los siguientes enlaces

- {`http://docs.oracle.com/javase/7/docs/`}
- {`http://docs.oracle.com/javase/8/docs/`}

- {http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html}
- {http://www.stack.nl/~dimitri/doxygen/index.html}
- {http://en.wikipedia.org/wiki/Markdown}

Requisito general a todos los programas en todas las actividades es: siempre termina el programa con un mensaje como *Program of exercise X has terminated*, es decir, todos los componentes del programa concurrente terminan correctamente su ejecución.

Las preguntas que aparecen intercalados en los anunciados tienen como objetivos: animar a la reflexión y al auto-aprendizaje, servir como ejemplos de posibles preguntas en la fase de evaluación (examen), fundamentan la base para los breves informes que se entregan para las actividades (más adelante hacia final de cada actividad).

1.1. Introducción a los hilos en Java

1. Examina las dos formas que provee Java para crear un hilo: la clase `Thread` y la interfaz `Runnable`.

2. Utiliza ambas formas para crear un programa que cree y ejecute un hilo que imprima en pantalla un mensaje como *Hello world, I'm a java thread*.

   ¿Hay alguna diferencia de funcionamiento entre ambas formas? ¿A nivel de diseño, cuál te parece preferible, y por qué?

   Es decir, realiza dos programas: `Thread.java` y `Runnable.java`.

3. Modifica tu programa para que el hilo tarde aproximadamente un segundo en mostrar el mensaje. ¿Qué método usarás para ello?

   Es decir, realiza un programa: `Segundo.java`.

4. Cuando arrancas un hilo desde el programa principal, ¿cuántos hilos hay activos? ¿Qué métodos y herramientas tienes disponibles para averiguarlo?

   Saca una lista de los hilos activos por pantalla.

   Es decir, realiza un programa: `Activos.java`.

1.2. Creación de múltiples hilos

1. Escribe un programa en Java que mediante parámetros de línea de comando reciba cuantos hilos se debe crear. El programa creará y ejecutará el número de hilos indicado. Cada hilo debe imprimir en pantalla un mensaje como *Hello, I'm thread number X*, y después de uno o varios segundos (puedes usar un segundo argumento via línea de comando), un mensaje como *Bye, this was thread number X*, siendo *X* el valor de un contador que se va incrementando con el número de hilos creados. Experimenta con diferentes argumentos, incluso yiendo a valores grandes. ¿Las salidas del programa reflejan lo que has esperado?

   Es decir, realiza un programa: `Hilos.java`.