Aproximación a la programación concurrente en Java

Índice de contenido

1. Requisitos software

- Eclipse IDE for Java Developers, http://www.eclipse.org/downloads/
- VisualVM incluida en jdk versión 6 o superiores

2. Objetivos específicos

- Crear y manipular hilos en Java
- Analizar y reflexionar sobre las ejecuciones no deterministas en programación concurrente
- Monitorizar la ejecución de programas concurrente en Java analizando los estados de los hilos (en ejecutable, en ejecución, en espera sincronizada)

3. Enunciado

En la práctica se presentan cinco casos de estudio de programación concurrente en Java.

(a) Caso de estudio 1: Concurrencia básica

Desarrollar una aplicación Java, en un paquete llamado `pg01.c01`, que cree tres tipos diferentes de hilos y los lance con el método `start()`. Cada uno de ellos escribirá 100 veces un carácter por pantalla. El primero escribe el carácter 'x', el segundo el carácter 'o' y el tercero el carácter '-'.

Realizar varias implementaciones de la práctica utilizando la `interface Runnable` y la subclase `Thread`.

- **Preguntas:** Ejecuta el programa varias veces y analiza el resultado de cada ejecución. ¿Son las trazas de ejecución siempre iguales? Describe el funcionamiento del planificador de hilos. El método `yield` de la clase `Thread` se encarga de dejar paso a otro hilo para que se ejecute, ¿ha cambiado algo en la ejecución?
- Si quieres ver los estados por los que pasan los hilos en el monitor VisualVM modifica el comportamiento para que los hilos escriban 1000000 de caracteres

(b) Caso de estudio 2: Hilos en espera

Modifica el código del caso de estudio 1, en un paquete llamado `pg01.c02`, tal que se añada a cada uno de los hilos iniciales una instrucción de parada (`Thread.sleep()` ó `java.util.concurrent.TimeUnit.sleep()`).

- **Preguntas:**¿Qué ocurre si todas los hilos tienen el mismo tiempo de espera? ¿Y si estos tiempos de espera son distintos? Haz que el programa principal espere la terminación de todos los hilos usando

el método `Thread.join()` .

(c) Caso de estudio 3: Constructores parametrizados

Modifica el código del caso de estudio 2, en un paquete llamado `pg01.c03`, tal que se cree y lance un único tipo de hilo que reciba tres parámetros en su creación. El primero, de tipo `char`, se usará para indicar que carácter se quiere escribir por pantalla. El segundo, de tipo `int`, se usará para indicar el tiempo que el hilo debe estar parado entre dos escrituras sucesivas de su carácter. El tercero,de tipo `int`, indicará cuántas veces tiene que escribir el carácter en la pantalla.

- **Preguntas:**¿Qué se puede decir del orden en el que se activan los hilos? Cambia las prioridades de los hilos `Thread.setPriority()` y analiza los resultados de ejecución y el orden de activación.

(d) Caso de estudio 4: Ejecutores

Modifica el código del caso de estudio 3, en un paquete llamado `pg01.c04`, tal que utilice un servicio de ejecución para lanzar los hilos (`java.util.concurrent.ExecutorService`).

Ejecuta el programa varias veces utilizando diferentes tipos ejecutores:
`Executors.newCachedThreadPool(), Executors.newFixedThreadPool(1);`
`Executors.`*`newSingleThreadExecutor`*`();`*`Executors.newFixedThreadPool(2);Executors.newFixedThreadPool(3);`*

- **Preguntas:**¿Son las trazas de ejecución siempre iguales? Describe el comportamiento de cada ejecutor. ¿Cuando se aconseja utilizar ejecutores en vez del método `Thread.start()`? ¿Qué se entiende por *degrade gracefully* de una aplicación?

(e) Caso de estudio 5: Análisis y monitorización de ejecuciones concurrentes con interrupción

Dado el código de la clase `SimpleThreads` disponible en http://docs.oracle.com/javase/tutorial/essential/concurrency/simple.html realizar un conjunto de ejecuciones parametrizadas tal que se obtengan los resultados de la Tabla 1.

Salida de ejecución 1	Salida de ejecución 2

`main: Starting MessageLoop thread`
`main: Waiting for MessageLoop thread`
`to finish`
`main: Still waiting...`
`main: Still waiting...`
`main: Still waiting...`
`main: Still waiting...`
`Thread-0: Mares eat oats`
`main: Still waiting...`
`main: Still waiting...`
`main: Still waiting...`
`main: Still waiting...`
`Thread-0: Does eat oats`
`main: Tired of waiting!`
`Thread-0: I wasn't done!`
`main: Finally!`
	`main: Starting MessageLoop thread`
	`main: Waiting for MessageLoop thread to finish`
	`main: Still waiting...`
	`main: Still waiting...`
	`main: Tired of waiting!`
	`Thread-0: I wasn't done!`
	`main: Finally!`

Salida de ejecución 3	Salida de ejecución 4
main: Starting MessageLoop thread main: Waiting for MessageLoop thread to finish main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: Mares eat oats main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: Does eat oats main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: Little lambs eat ivy main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: A kid will eat ivy too main: Finally!	main: Starting MessageLoop thread main: Waiting for MessageLoop thread to finish main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: Mares eat oats main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: Does eat oats main: Still waiting... main: Still waiting... main: Still waiting... main: Still waiting... Thread-0: Little lambs eat ivy main: Tired of waiting! Thread-0: I wasn't done! main: Finally!

Tabla 1: Salidas de ejecución de un programa concurrente SimpleThreads

- **Preguntas:** ¿Cuál es el valor del parámetro de entrada con el que se ha conseguido cada salida? ¿Qué puedes decir de las terminación de los hilos y su relación con los métodos `Thread.interrupt()` y `Thread.join()`?

4. Productos a entregar

En esta práctica no es obligatorio entregar nada al profesor, pero es interesante documentar la experiencia. Con la siguiente información de los casos de estudio

- Descripción del caso
- Código fuente en Java
- Conjunto de salidas de ejecución del programa Java en distintos instantes de tiempo,
- Conjunto de *snapshots* de monitorización de la ejecución obtenidos con VisualVM
- Respuesta a las preguntas de reflexión sobre el caso de estudio.

La práctica será evaluada mediante un cuestionario presencial que se realizará en el laboratorio durante horas de prácticas.

5. Bibliografía

- Oracle. «Lesson: Concurrency (The Java™ Tutorials > Essential Classes)». 2013. http://docs.oracle.com/javase/tutorial/essential/concurrency/index.html.
- Oracle. «VisualVM». Accedido febrero 13, 2014. http://docs.oracle.com/javase/6/docs/technotes/guides/visualvm/index.html.