Programación Concurrente en Java
Laboratorio 0: IncDecStar

J. Blanco, N. Wolovick

Objetivos

- Familiarizarse con el ciclo de edición, compilación y ejecución de Java.
- Probar en la práctica, usando Java, el lema de "Topología de Programas"
  (Práctico 0-Ejercicio 1 y Ejemplo 3 de Capítulo 5 de "On a Method of Multiprogramming")
- Entender que el grado de atomicidad es fundamental para la programación concurrente.
- Probar algunos mecanismos que provee Java5/6 para lograr atomicidad.
- Utilizar `assert` para revisar las aserciones de la lógica en run-time.

Actividades

1. [**25 pts**] Correr el programa `IncDecStar` y ver que eventualmente el `assert` de `TesterThread` falla. Copiar y pegar algunas trazas resumidas donde se vea este comportamiento.
   Ayuda: Si no consigue que falle el `assert` agregar `Thread.yield()`s en el código.

2. [**25 pts**] Mostrar la parte del Java bytecode donde queda explícito el grado de atomocidad del incremento.
   Ayuda: `javap -c IncDecThread`.

3. [**50 pts**] Obtener dos versiones `IncDecStar1.java` y `IncDecStar2.java` utilizando dos mecanismos distintos de atomicidad de Java. Comprobar que efectivamente no falla en distintos ambientes de ejecución (compilador, JVM, plataforma, con y sin `Thread.yield()`).

Actividades Extra

1. [**10 pts**] Utilizar algún compilador de Java a código de máquina nativo (*ahead-of-time compiler* – GCJ, Excelsior Jet, etc.) y desensamblar el código generado para ver si el incremento/decremento es atómico.

2. [**10 pts**] Comprobar si se llega a los mismos resultados eliminando `TesterThread` y agregando los asserts dentro del código. Incluir la aserción $\{0 < x\}$ entre el incremento y el decremento.

Laboratorio 0 – Revisión: 1518, (2010-03-08)