Memoria de práctica

Ejemplos (DSD P1S1)

Román Larrosa Lewandowska

marzo 2020

Introducción

En el siguiente documento se detalla el proceso de prueba y los fenómenos observados en los tres ejemplos que se proponen en el guión, así como los cambios realizados en uno de los ejemplos derivados de los errores que se produjeron durante la prueba de éste.

Ejemplo 1

Para la prueba de este ejemplo uso el código del guión, que he transcrito en el directorio E1 (que se incluye en el zip aportado). El directorio contiene:

- server.policy: archivo donde se determinan los permisos que tendrán los ejecutables sobre el directorio de trabajo.
- Ejemplo_I.java: Se declara la interfaz, en la que se determinan que operaciones estarán disponibles para ser ejecutadas por el cliente en el servidor
- Ejemplo.java: Se implementan las operaciones declaradas en la interfaz, y se crea el objeto remoto con el que operará el cliente, que es enviado al registry
- Cliente_Ejemplo: en este proceso, busca en el registry el objeto remoto con el que se operará, y ejecuta la operación.

Para ejecutar este ejemplo, se procede a ejecutar los siguientes comandos **en el directorio donde está ubicado el ejemplo** (IMPORTANTE):

`$ rmiregistry &` Se inicia el rmiregistry

`$ javac *.java` Se compila

Ejecutamos el servidor en un terminal:

```
$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejemplo
```

En otro terminal, ejecutamos el cliente

```
java -cp . -Djava.security.policy=server.policy Cliente_Ejemplo localhost 0
```

El último número determina el número de proceso, que es relevante puesto que el proceso identificado con 0, hará que en la ejecución de la operación en el servidor, éste duerma a la hebra durante 5 segundos. En esos 5 segundos podemos lanzar desde otro terminal otro cliente

con otro número de proceso, que se ejecutará en el servidor mientras la hebra del primer proceso está durmiendo, al ser el servidor un proceso que implementa la multihebra.

Sin embargo, si reproducimos ese comportamiento, no se puede garantizar la exclusión mutua,algo con lo que experimentamos con el siguiente ejemplo

Ejemplo 2

Versión 1: **No sincronizada**

Para este ejemplo, todo se mantiene como en el primero, con la diferencia de que al cliente el número que le pasamos es el número de hebras que ejecutarán una llamada al cliente. Este ejemplo se encuentra en el directorio E2 del zip. Las hebras ejecutan la operación escribirMensaje pasandole al servidor su número de hebra a modo de String. El servidor, cuando va a ejecutar un mensaje, mira si la hebra acaba en 0, y en dicho caso, duerme a la hebra durante 5 segundos. Cuando desde el cliente lanzamos varias hebras al servidor:

```
java -cp . -Djava.security.policy=server.policy Cliente_Ejemplo_MT localhost 20
```

en el servidor se produce un comportamiento parecido a este:

```
Entra Hebra Cliente 8
Entra Hebra Cliente 10
Empiezo a dormir...
Entra Hebra Cliente 7
Sale Hebra Cliente 8
Sale Hebra Cliente 16
Sale Hebra Cliente 14
Sale Hebra Cliente 11
Entra Hebra Cliente 12
Entra Hebra Cliente 9
Sale Hebra Cliente 9
Entra Hebra Cliente 19
Sale Hebra Cliente 6
Entra Hebra Cliente 5
Entra Hebra Cliente 17
Entra Hebra Cliente 3
Sale Hebra Cliente 17
Sale Hebra Cliente 5
Sale Hebra Cliente 19
Sale Hebra Cliente 12
Sale Hebra Cliente 7
Sale Hebra Cliente 3
He terminado de dormir.
Sale Hebra Cliente 0
He terminado de dormir.
Sale Hebra Cliente 10
```

Como podemos observar, la ejecución de las hebras se entremezcla, lo que puede ocasionar problemas en un supuesto ejemplo en el que necesitemos que se garantice la exclusión mutua. Para probar ese comportamiento, podemos añadir al método implementado en el servidor la palabra reservada de java “synchronized”, que hará que no se pueda invocar más de una vez

ningún método synchronized para un mismo objeto. Es decir, para un mismo objeto, solo puede estar ejecutándose el código de un único método synchronized, con lo que es sencillo garantizar la exclusión mutua. Esto lo veremos con la versión 2 de este ejemplo.

Versión 2: **Sincronizada**

Este ejemplo se encuentra en el directorio E2SYN del zip. La única diferencia entre esta versión y la primera es que el método escribirMensaje está declarado como synchronized. Gracias a esto, las hebras se ejecutarán una detrás de otra, esperando a que una acabe antes de que comience a ejecutarse la segunda. Esto genera un comportamiento como el siguiente:

```
Entra Hebra Cliente 4
Sale Hebra Cliente 4
Entra Hebra Cliente 10
Empiezo a dormir...
He terminado de dormir.
Sale Hebra Cliente 10
Entra Hebra Cliente 1
Sale Hebra Cliente 1
Entra Hebra Cliente 9
Sale Hebra Cliente 9
Entra Hebra Cliente 11
Sale Hebra Cliente 11
Entra Hebra Cliente 16
Sale Hebra Cliente 16
Entra Hebra Cliente 13
Sale Hebra Cliente 13
Entra Hebra Cliente 2
Sale Hebra Cliente 2
Entra Hebra Cliente 7
Sale Hebra Cliente 7
Entra Hebra Cliente 0
Empiezo a dormir...
He terminado de dormir.
Sale Hebra Cliente 0
```

Como se puede observar, las hebras entran, salen, y dan paso a la siguiente. En el caso de las que acaban con 0, la hebra espera los 5 segundos, sin que ninguna otra se mezcle hasta que despierta. Con esto podemos ver que se garantiza la exclusión mutua.

Ejemplo 3

Este ejemplo se encuentra en el directorio E3 del zip. Su contenido es:

- server.policy: al igual que en los ejercicios anteriores, establece los permisos.
- I_contador.java: interfaz a implementar
- contador.java: implementa la interfaz I_contador.java
- servidor.java: crea el objeto remoto con el que operará el cliente, que es enviado al registry
- cliente.java:: en este proceso, se busca en el registry el objeto remoto con el que se operará, guarda los tiempos que tarda en ejecutarse 1000 veces el método suma en el servidor, y calcula la media de tiempo que tarda en realizarse una RMI.

Para ejecutar este ejemplo, se procede a ejecutar los siguientes comandos **en el directorio donde está ubicado el ejemplo** (IMPORTANTE):

`$ rmiregistry &` Se inicia el rmiregistry

`$ javac *.java` Se compila

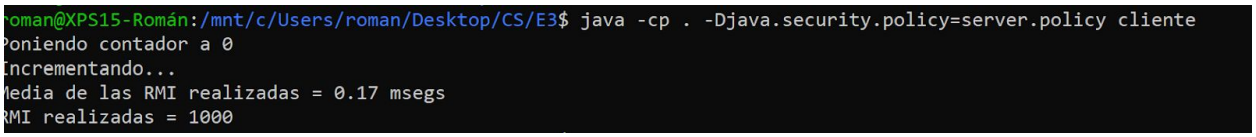
Ejecutamos el servidor en un terminal:

```
$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy servidor
```

En otro terminal, ejecutamos el cliente

```
java -cp . -Djava.security.policy=server.policy cliente localhost
```

El cliente mostrará por pantalla algo parecido a esto:



Como se producían varios errores en la ejecución de este ejemplo en mi máquina, he adaptado el código para que se parezca un poco más a los dos primeros ejemplos. En lugar de lanzar el registry desde el código, lo hago antes en el directorio, y el objeto no extiende a `UnicastRemoteObject`, por lo que en el servidor hay que exportar al objeto contador como un `UnicastRemoteObject` antes de hacer el rebind al registry.