Práctica 3. Primera parte: Implementación de los ejemplos

Noelia Escalera Mejías Grupo DSD1

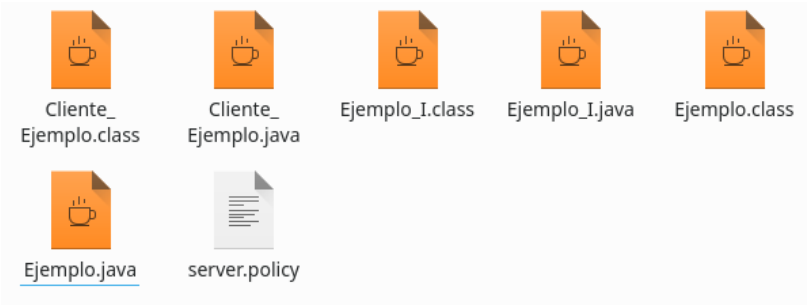
April 4, 2020

1 Ejemplo 1

1. Crear los archivos

2. Compilamos los *.java*.

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$ javac *.java
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$
```



3. Invocamos *rmiregistry*

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$ rmiregistry &
[1] 5394
```

4. Lanzamos el servidor

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Ejemplo
Ejemplo bound
```

5. Lanzamos el cliente

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Cliente_Ejemplo localhost 3
Buscando el objeto remoto
Invocando el objeto remoto
```

En el servidor obtenemos esto:

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Ejemplo
Ejemplo bound
Recibida peticion de proceso: 3

Hebra 3
```

Si le pasamos 0 como argumento al cliente:

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 1$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Cliente_Ejemplo localhost 0
Buscando el objeto remoto
Invocando el objeto remoto
```

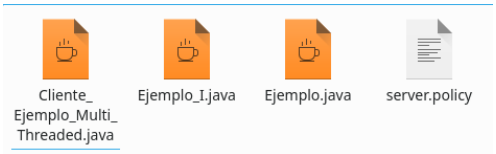
```
Recibida peticion de proceso: 0
Empezamos a dormir
Terminamos de dormir

Hebra 0
```

Lo que está ocurriendo es que el cliente busca un objeto remoto, encuentra el que ha creado nuestro servidor y lo llama pasándole como argumento el que le hemos pasado nosotros. El servidor los interpreta como número de proceso y dependiendo de si es el proceso 0 o no hace un sleep o no.

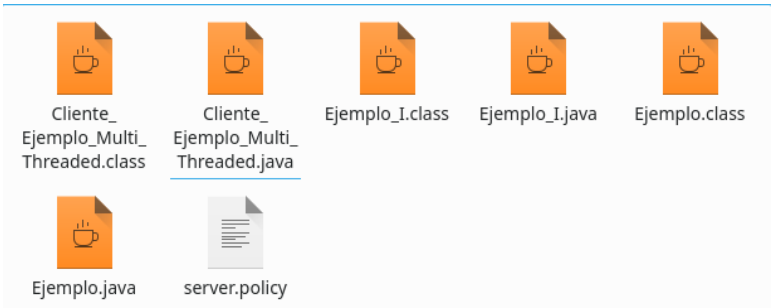
2 Ejemplo 2

1. Creamos los archivos



2. Compilamos los *.java*.

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ javac *.java
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$
```



3. Invocamos *rmiregistry*

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ rmiregistry &
[1] 7178
```

4. Lanzamos el servidor

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Ejemplo
Ejemplo bound
```

5. Lanzamos el cliente

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Cliente_Ejemplo_Multi_Threaded localhost 5
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
```

En el servidor obtenemos esto:

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Ejemplo
Ejemplo bound

Entra Hebra Cliente2
Sale Hebra Cliente2

Entra Hebra Cliente0

Entra Hebra Cliente1
Sale Hebra Cliente1
Empezamos a dormir

Entra Hebra Cliente3
Sale Hebra Cliente3

Entra Hebra Cliente4
Sale Hebra Cliente4
Terminamos de dormir
Sale Hebra Cliente0
```

Lo que ocurre aquí es que al cliente le pasamos como argumento el número de hebras que se quieren lanzar. El servidor hace sleep a los procesos múltiplos de 10.

Si añadimos ahora el modificador synchronized:

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Cliente_Ejemplo_Multi_Threaded localhost 5
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$
```

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy Ejemplo
Ejemplo bound

Entra Hebra Cliente0
Empezamos a dormir
Terminamos de dormir
Sale Hebra Cliente0

Entra Hebra Cliente4
Sale Hebra Cliente4

Entra Hebra Cliente1
Sale Hebra Cliente1

Entra Hebra Cliente2
Sale Hebra Cliente2

Entra Hebra Cliente3
Sale Hebra Cliente3
```

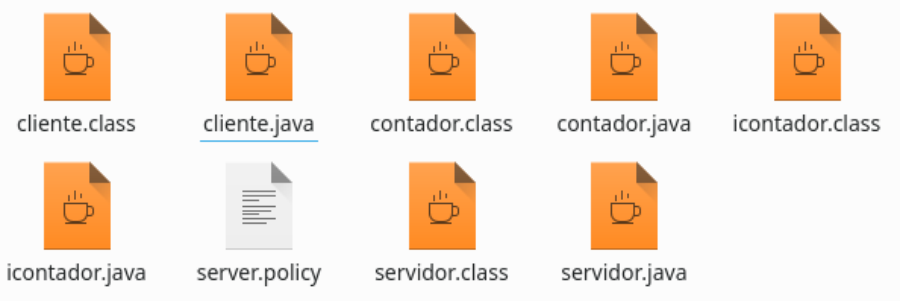
Lo que ocurre ahora es que no se ejecuta una hebra hasta que se ha terminado de ejecutar la que estaba antes.

3 Ejemplo 3

1. Creamos los archivos

2. Compilamos los *.java*.

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 3$ javac *.java
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 3$
```



3. Invocamos *rmiregistry*

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 3$ rmiregistry &
[1] 8841
```

4. Lanzamos el servidor

```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo 3$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.pol
icy servidor
Servidor RemoteException | MalformedURLExceptiondor preparado
```

Antes apareció el siguiente error:

Se ha solucionado cambiando el puerto en *servidor.java*

5. Lanzamos el cliente

Lo que ocurre aquí es que el servidor sirve un contador y el cliente lo llama 1000 veces.