

Práctica 3. Primera parte: Implementación de los ejemplos

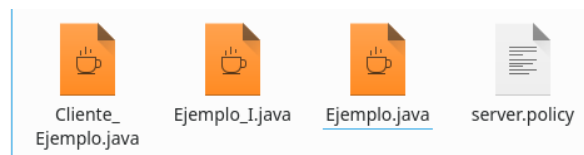
Noelia Escalera Mejías

Grupo DSD1

April 4, 2020

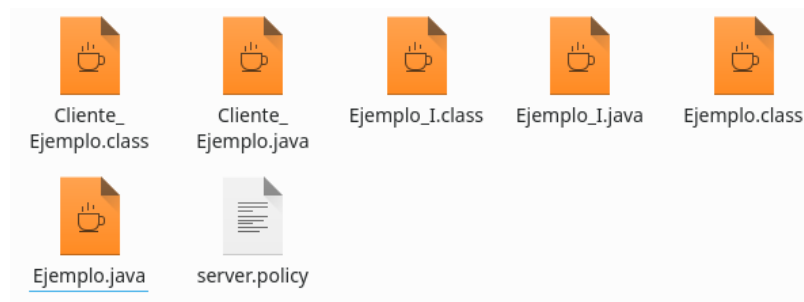
1 Ejemplo 1

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

5. Lanzamos el cliente

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

Si le pasamos 0 como argumento al cliente:

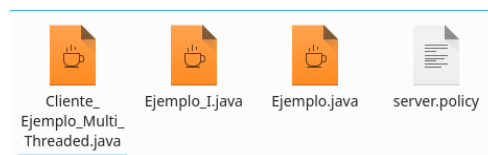
```
noelia@noelia-pc:~/Escritorio/Universidad/3º/DSD/Prácticas/P3/Parte 1/Ejemplo $ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy ClienteEjemplo localhost 0
Buscando el objeto remoto
Invocando el objeto remoto
```

```
Recibida petición 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.

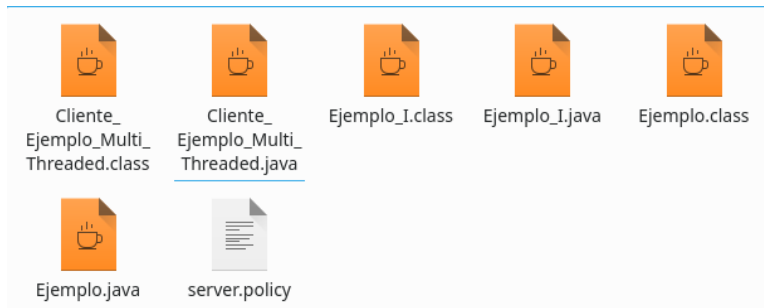
2 Ejemplo 2

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

```
root@lagonet1-pc:~/Escritorio/Universdad/3º/DSO/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp . -Djava.rmi.server.codebase=file:/ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejemplo
Ejemplo bound
```

5. Lanzamos el cliente

[illegible]

En el servidor obtenemos esto:

```

mal@malmele-pc:Escritorio/Universidad/2º/ODD/Prácticas/P3/Parte 1/Ejemplo 2$ java -cp -Djava.rmi.server.codebase=file:/ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Ejemplo
Ejemplo bound
Entra Hebra Cliente2
Sale Hebra Cliente2
Entra Hebra Cliente0
Entra Hebra Cliente1
Sale Hebra Cliente1
Dormimos 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 múltiples de 10.

Si añadimos ahora el modificador `synchronized`:

```
jml@joomla-1 ~ % curl -F "url=http://localhost:8080/PraCtices/P3/Parte_1/joomla_25" java -cp /opt/java/lib/server.codexbase/files:/opt/java/lib/server.hostname=localhost:/opt/java/security.policy:server.policy Cliente Ejemplo.MultiThreaded 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
jml@joomla-1 ~ % curl -F "url=http://localhost:8080/PraCtices/P3/Parte_1/joomla_25"
```

```

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.policy Ejemplo
Ejemplo bound
Entrar Hebra Cliente0
Espezamos a dorair
Terminamos de dorair
Sale Hebra Cliente0
Entrar Hebra Cliente4
Sale Hebra Cliente4
Entrar Hebra Cliente1
Sale Hebra Cliente1
Entrar Hebra Cliente2
Sale Hebra Cliente2
Entrar 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

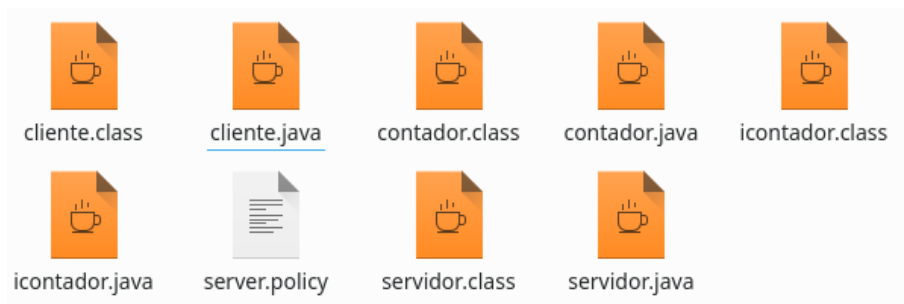


2. Compilamos los *.java*.

```

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.policy servidor
Servidor RemoteException | MalformedURLExceptionondor preparado

```

Antes apareció el siguiente error:

```
osallanmella-pc:~/Escritorio/Universidad/31/DSD/Prácticas/P3/Parte 1/Ejemplo 3$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy servidor
Exception: Port already in use: 1899; nested exception is:
    java.net.BindException: La dirección ya se está usando (Bind failed)
```

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

5. Lanzamos el cliente

```
osallanmella-pc:~/Escritorio/Universidad/31/DSD/Prácticas/P3/Parte 1/Ejemplo 3$ java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy cliente localhost
Poniendo contador a 0
Incrementando...
Media de las RMI realizadas = 0.099 msecs
RMI realizadas = 1000
```

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