

## PRÁCTICA 3:RMI

### Desarrollo de la práctica: Linux

#### Version de java que he utilizado:

openjdk 11.0.11 2021-04-20

OpenJDK Runtime Environment (build 11.0.11+9-Ubuntu-0ubuntu2.20.04)

OpenJDK 64-Bit Server VM (build 11.0.11+9-Ubuntu-0ubuntu2.20.04, mixed mode, sharing)

## 1.EJEMPLOS

### 1.1.Ejemplo 1

Para la ejecución del ejemplo 1, abrimos dos terminales en la ubicación del archivo.

Antes de nada lanzamos el comando **rmiregistry &** el cual nos permite registrar y localizar los objetos remotos, el & lo utilizamos para lanzarlo en segundo plano.

Una vez hecho, en una terminal ejecutamos el script del server mediante **./scriptServer.sh** , en la otra terminal lanzamos el script de los clientes (dos) mediante **./scriptClientes.sh** .

En cuanto al funcionamiento de este ejemplo consiste en que el servidor recibe una petición (método escribir\_mensaje) de un cliente (un número), si este numero es un 0 duerme 5 segundos.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo1$ ./scriptClientes.sh
Lanzando el primer cliente
Buscando el objeto remoto
Invocando el objeto remoto
Lanzando el segundo cliente
Buscando el objeto remoto
Invocando el objeto remoto
```

**Imagen 1:** Ejecución del script del cliente donde se lanzan dos clientes que invocan al objeto remoto.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo1$ ./scriptServer.sh
Compilando con javac ...
Lanzando el servidor
Ejemplo bound
Recibida peticion de proceso: 0
Empezamos a dormir
Terminamos de dormir
Hebra 0
Recibida peticion de proceso: 3
Hebra 3
```

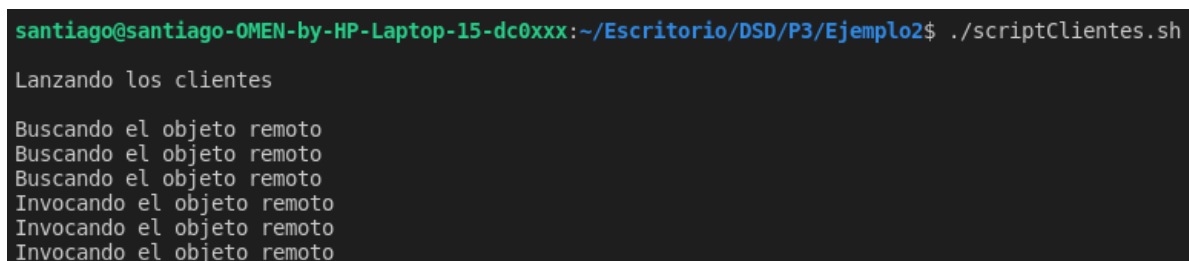
**Imagen 2:** Ejecución del script del servidor donde recibe la petición del primer cliente, como el número es un 0 este duerme 5 segundos. Luego recibe la petición del segundo cliente, como el número es un 3 este solo recibe la petición mostrando el mensaje.

## 1.2.Ejemplo 2

Para la ejecución del ejemplo 2 es la misma que la del ejemplo anterior, necesitamos dos terminales, en una lanzamos el script del server mediante **./scriptServer.sh** y en la otra terminal lanzamos el script del cliente mediante **./scriptClientes.sh**.

El funcionamiento de este ejemplo por parte del servidor es el mismo que el ejemplo anterior, el único cambio esta en que recibe un mensaje con el nombre de la hebra y el número de esta, si es 0 duerme 5 segundos, en caso contrario solo muestra el mensaje .

La diferencia significativa en este ejemplo esta en que en vez de lanzar varios cliente mediante distintas ejecuciones, lanzamos varias hebras que invocan al objeto remoto y realizan la misma tarea de imprimir mensaje.



```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo2$ ./scriptClientes.sh
Lanzando los clientes
Buscando el objeto remoto
Buscando el objeto remoto
Buscando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
Invocando el objeto remoto
```

sU

**Imagen 3:** Ejecución del script del cliente, en el cual se dice que se ejecuten 3 hebras y que invoquen al método escribir mensaje del objeto remoto.



```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo2$ ./scriptServer.sh
Compilando con javac ...
Lanzando el servidor
Ejemplo bound
Entra Hebra Cliente 2
Entra Hebra Cliente 0
Empezamos a dormir
Sale Hebra Cliente 2
Entra Hebra Cliente 1
Sale Hebra Cliente 1
Terminamos de dormir
Sale Hebra Cliente 0
```

**Imagen 4:** Ejecución del script del server, el cual recibe las peticiones de las 3 hebras, cual llega la petición de la hebra 0 este duerme 5 segundos.

Podemos observar en la ejecución anterior que los mensajes se entrelazan aunque el funcionamiento sea correcto. Para evitar ponemos el modificador **synchronized** en el método **escribir\_mensaje** de la implementación remota.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo2$ ./scriptServer.sh
Compilando con javac ...
Lanzando el servidor
Ejemplo bound
Entra Hebra Cliente 1
Sale Hebra Cliente 1
Entra Hebra Cliente 0
Empezamos a dormir
Terminamos de dormir
Sale Hebra Cliente 0
Entra Hebra Cliente 2
Sale Hebra Cliente 2
```

**Imagen 5:** Ahora observamos que no se entrelazan los mensajes lo que lo hace más entendible, esto se debe a que el modificador `synchronized` evita que se intercalen las invocaciones.

### 1.3.Ejemplo 3

Para ejecutar el ejemplo 3 hay que matar al proceso que está utilizando el puerto 1099, cambiar el puerto del servidor de este ejemplo o en vez de utilizar `createRegistry` utilizar `getRegistry` como en los ejemplos anteriores.

Con este comando: `netstat -putona | grep :::1099 | awk '{print $7}' | cut -d '/' -f 1` obtienes el PID de dicho puerto y con `kill PID` matas al proceso.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo3$ netstat -putona | grep :::1099 | awk '{print $7}' | cut -d '/' -f 1
c(No todos los procesos pueden ser identificados, no hay información de propiedad del proceso
no se mostrarán, necesita ser superusuario para verlos todos.)
8811
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo3$ kill 8811
```

**Imagen 6:** Ejemplo de cómo matar al proceso que utiliza el puerto 1099 como siempre, utilizo `rmiregistry &`, se que el puerto es 1099 porque es el por defecto.

La ejecución de este ejemplo es idéntica a los dos anteriores (lanzar en dos terminales distintos el `scriptServer.sh` y `scriptClientes.sh`).

En cuanto al funcionamiento de este ejemplo: es básico esta vez se crea por una parte el objeto remoto y por otra el servidor, este exporta los métodos de la interfaz del objeto remoto.

El cliente invoca al objeto remoto pone el contador a 0, lo incrementa 1000 veces y muestra la media RMI, todo esto invocando a los métodos del objeto remoto.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo3$ ./scriptServer.sh
Compilando con javac ...
Lanzando el servidor
Servidor RemoteException | MalformedURLException preparado
```

**Imagen 7:** Ejecución del `scriptServer` que pone en marcha al servidor para que el cliente pueda realizar sus funciones.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejemplo3$ ./scriptClientes.sh
Lanzando los clientes
Poniendo contador a 0
Incrementando...
Media de las RMI realizadas = 0.092 msecs
RMI realizadas = 1000
```

**Imagen 8:** Ejecución del scriptClientes donde obtiene la instancia del objeto remoto, inicializa su contador a 0, lo incrementa en 1000 y obtiene la media de las RMI realizadas, todo ello invocando a los métodos de dicho objeto remoto.

Para la realización del ejercicio siguiente me he basado en este ejemplo de programa cliente-servidor.

## 2.EJERCICIO DONACIONES

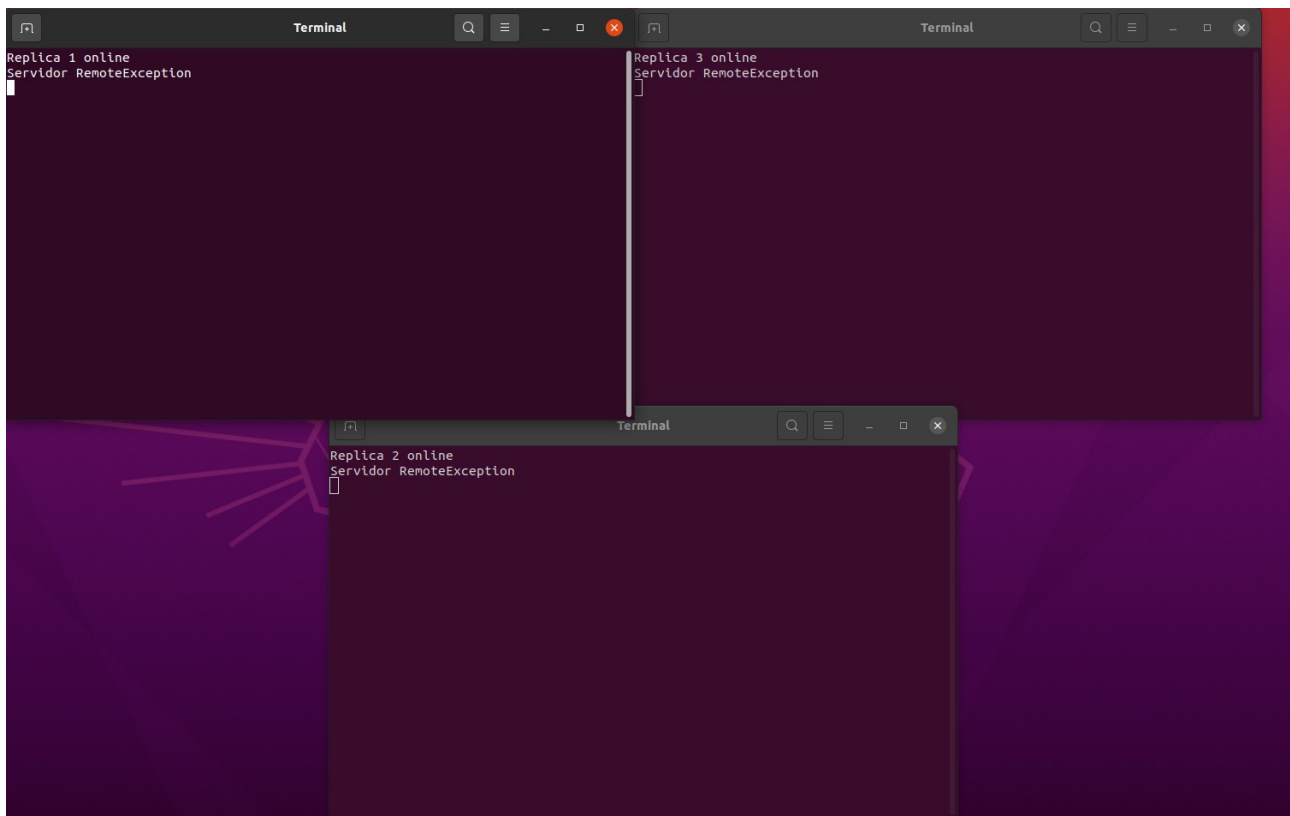
### 2.1.Ejecución

Si no se ha hecho antes, lanzar el comando **rmiregistry &** para poder registrar y localizar los objetos remotos(replicas del servidor), el & lo utilizamos para lanzarlo en segundo plano.

A continuación, al contrario que los anteriores ejemplos, solo hace falta tener un terminal con la ruta del fichero Ejercicio. Primero lanzamos las replicas ejecutando el script para el servidor mediante **./scriptServer.sh**, este va a lanzar 3 replicas del servidor de donación en tres terminales distintas, para abrir estos terminales utilizo el comando **gnome-terminal -e "comando para lanzar el servidor"**.

```
1  #!/bin/sh -e
2
3  echo
4  echo "Compilando con javac ..."
5  javac *.java
6
7  sleep 2
8
9  echo
10 echo "Lanzando replica 2"
11 gnome-terminal -e "java -cp . -Djava.rmi.server.codebase=file:/// -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy servidorDonacion 1"
12
13 echo
14 echo "Lanzando replica 2"
15 gnome-terminal -e "java -cp . -Djava.rmi.server.codebase=file:/// -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy servidorDonacion 2"
16
17 echo
18 echo "Lanzando replica 3"
19 gnome-terminal -e "java -cp . -Djava.rmi.server.codebase=file:/// -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy servidorDonacion 3"
```

**Imagen 9:** Podemos observar que lanzamos tres replicas en tres terminales distintas en el script, podemos lanzar mas repitiendo el comando y cambiando el número del final del comando que especifica el numero de replica. También se podría hacer un bucle for donde se repita x veces el comando donde x es el número de replicas, lo que quiero llegar a decir es que hay varias formas de crear varias replicas, pero yo he optado por crear solo 3 (una más de la que se pide) para la realización de la práctica.



**Imagen 10:** Los tres terminales se abren , cada uno creando una replica del servidor (replica1 , replica2 y replica3).

Luego de esto, en el propio terminal donde hemos lanzado el script anterior, ejecutamos el script del cliente mediante **./scriptClientes.sh**.

```
1  #! /bin/sh -e
2
3  echo
4  echo "Lanzando el cliente"
5  echo
6  java -cp . -Djava.security.policy=server.policy cliente
7
```

**Imagen 11:** Script del cliente, básicamente ejecutamos el main del cliente, simulando que un cliente va a utilizar el sistema de donaciones, si queremos podemos lanzar en otras terminales el script para simular a más de un cliente, pero para explicar el funcionamiento lo haré mejor con uno solo.

## 2.2.Utilización

Una vez lanzadas las replicas y el cliente, vamos a utilizar todo lo que hemos implementado. Al lanzar el cliente se mostrara una interfaz de texto por la terminal, esta es muy fácil de seguir, igualmente voy a hacer una simulación donde se muestren todas las funciones definidas.

Primero nos darán la bienvenida, podemos iniciar sesión o registrarnos, no podremos iniciar sesión al principio ya que no hay ningún usuario registrado, por lo que debemos registrarnos con un nombre de usuario y contraseña.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejercicio$ ./scriptClientes.sh
Lanzando el cliente
¡Bienvenido!, antes de empezar a donar debe registrarse en el sistema
Elija una opción:
 1-Iniciar Sesión
 2-Registrarse
Opcion: 2
Introduzca su nombre de usuario: Santi
Introduzca su contraseña: 12345
Se ha registrado el usuario Santi en el servidor replica 1
Hola Santi!!!!, bienvenido a nuestro sistema de donaciones.
```

**Imagen 12:** Observamos como nos hemos registrado como Santi y con contraseña 12345 en el servidor replica 1 ya que al principio todas las replicas tienen 0 usuarios por lo que se registra en la primera.

```
santiago@santiago-OMEN-by-HP-Laptop-15-dc0xxx:~/Escritorio/DSD/P3/Ejercicio$ ./scriptClientes.sh
Lanzando el cliente
¡Bienvenido!, antes de empezar a donar debe registrarse en el sistema
Elija una opción:
 1-Iniciar Sesión
 2-Registrarse
Opcion: 2
Introduzca su nombre de usuario: Santi
Introduzca su contraseña: 123
El usuario Santi ya se encuentra registrado en el servidor replica 1
Elija una opción:
 1-Iniciar Sesión
 2-Registrarse
Opcion: █
```

**Imagen 13:** Si volvemos a intentar registrarnos como Santi no nos dejen, diciéndonos que ya nos encontramos registrados en el servidor replica 1 (te dice que en que replica estas registrado).

```
Eliga una opción:
1-Iniciar Sesión
2-Registrarse

Opcion: 1

Introduzca su nombre de usuario: Santi
Introduzca su contraseña: 12345
Inicio de sesión completado

Hola Santi!!!!, bienvenido a nuestro sistema de donaciones.

Nuestras acciones:
1-Donar
2-Consultar total
3-Eliminar Cuenta
4-Consultar dinero donado
5-Cerrar Sesion

Introduzca que acción quieres realizar: █
```

**Imagen 14:** En cambio, si volvemos e iniciamos sesión con el nombre de Santi y contraseña 12345, podremos ingresar de nuevo, si la contraseña es incorrecta se nos notificara (se ha implementado así para hacer más realista el sistema).

```
Eliga una opción:
1-Iniciar Sesión
2-Registrarse

Opcion: 2

Introduzca su nombre de usuario: Raul
Introduzca su contraseña: 123

Se ha registrado el usuario Raul en el servidor replica 2

Hola Raul!!!!, bienvenido a nuestro sistema de donaciones.
```

```
Eliga una opción:
1-Iniciar Sesión
2-Registrarse

Opcion: 2

Introduzca su nombre de usuario: Fran
Introduzca su contraseña: 12

Se ha registrado el usuario Fran en el servidor replica 3

Hola Fran!!!!, bienvenido a nuestro sistema de donaciones.
```

**Imagen 15 y 16:** Podemos observar que si lanzamos dos clientes más y nos registramos, uno será enviado a la replica 2 y el otro a la replica 3, cumpliendo con la regla de registrar a los usuarios en la replica con menor número de usuarios (utilizo 3 replicas para la parte extra del ejercicio, podríamos tener más o menos replicas pero va a seguir funcionando).

Una vez estamos identificados están son las acciones que podemos realizar:

```
Hola Santi!!!!, bienvenido a nuestro sistema de donaciones.

Nuestras acciones:
1-Donar
2-Consultar total
3-Eliminar Cuenta
4-Consultar dinero donado
5-Cerrar Sesion

Introduzca que acción quieres realizar: 1

Introduzca la cantidad a donar: 200

Has donado 200€, ¡muchas gracias!
```

**Imagen 17:** Donamos una cantidad de dinero a nuestra replica.

En la **imagen 18** podemos observar como dicha replica ha recibido el dinero y de que usuario.

```
Terminal
Replica 1 online
Servidor RemoteException
La replica 1 ha recibido una donacion de 200€ del usuario Santi
```

**Imagen 18**

```
Nuestras acciones:
1-Donar
2-Consultar total
3-Eliminar Cuenta
4-Consultar dinero donado
5-Cerrar Sesion

Introduzca que acción quieres realizar: 2

La cantidad total donada a nuestro sistema de donaciones es de 200€
```

**Imagen 19:** Consulta el dinero donado al sistema, en este caso es igual al que habíamos donado antes y por eso nos deja consultarlo. En la **imagen 20** veremos como el usuario Raúl de la replica 2 no puede consultar el dinero ya que no ha donado.



```
Nuestras acciones:
1-Donar
2-Consultar total
3-Eliminar Cuenta
4-Consultar dinero donado
5-Cerrar Sesión

Introduzca que acción quieres realizar: 2

Lo sentimos, no puedes consultar el total donado sin haber realizado antes una DONACION
```

## Imagen 20

```
Introduzca que acción quieres realizar: 1

Introduzca la cantidad a donar: 100

Has donado 100€, ¡muchas gracias!

Nuestras acciones:
1-Donar
2-Consultar total
3-Eliminar Cuenta
4-Consultar dinero donado
5-Cerrar Sesión

Introduzca que acción quieres realizar: 2

La cantidad total donada a nuestro sistema de donaciones es de 300€
```

**Imagen 21:** En esta imagen observamos como el usuario Raul ha donado esta vez 100 euros, por lo que ya puede consultar el dinero total, el cual es de 300 (100 suyos mas los 200 anteriores de Santi).

Las demás funciones no son necesarias de sacar captura para entender su funcionamiento, la de eliminar cuenta como dice su nombre elimina al usuario de la replica, por lo que si eliminamos por ejemplo la cuenta de Santi, la replica 1 ya no tendrá usuarios registrados pero si sigue guardando el dinero que ha sido donado (funcionalidad extra para hacerlo mas realista).

La de consultar dinero donado consulta el dinero donado a la replica por el usuario, en este caso no hace falta haber donado antes.

Por último la opción de cerrar sesión termina la ejecución del cliente.

Todas estas funcionalidades se han añadido para aportar más realismo y para la parte extra de implementar más funciones, al igual que utilizar más de dos replicas.

## 2.3.Cómo se ha implementado

Para finalizar vamos a explicar resumidamente como se ha programado e implementado el desarrollo de este Ejercicio.

Por una parte he creado el objeto remoto y por otra el servidor. El objeto remoto consta de todas las funciones vistas en el apartado anterior que son accesible remotamente. El servidor(servidorDonacion.java) exporta los métodos contenidos en la interfaz *idonacion.java* del objeto remoto instanciado como mmireplica de clase definida en donacion.java.

En servidorDonacion.java creo una instancia (con el nombre de mmireplica concatenada con un número que se pasa como argumento al ejecutar el servidor) de la clase donacion.java.

```
int numeroReplica=Integer.parseInt(args[0]);
idonacion replica = new donacion(numeroReplica);
String nombre="mmireplica"+args[0];
Naming.rebind(nombre, replica);

System.out.println("Replica "+args[0]+ " online");
```

### Imagen 22: servidorDonacion.java

Las funciones implementadas en donacion.java no tiene mucha complicación; este se compone de un array de usuariosRegistrados, contraseñasRegistradas ,dineroDonado, cantidadRecibida y número de replica.

- UsuariosRegistrados es un array de String que guarda los usuarios registrados en la replica.
- contraseñasRegistradas es un array de String que guarda las contraseñas de los usuarios registrados en la replica.
- DineroDonado es una array de enteros que guarda los dineros donados por los usuarios en el sistema.
- CantidadRecibida guarda la cantidad total de dinero de la replica.
- Numero de replica guarda el numero de la replica en la que estamos.

La relación entres los arrays es sencilla sabemos que la posición 0 de los tres va a ser la de un usuario, la posición 1 la de otro etc...

Por otro lado en cliente.java tiene un número de replica al que pertenece y un array de las replicas disponibles. Este array de replicas disponible esta formado por los objetos instanciados anteriormente, **se implementa mediante un for de 3 iteraciones para las 3 replicas, si creásemos 4 replicas deberíamos cambiar este for para que hiciese una iteración más.**

A partir de aquí, jugamos con este vector de replicas, por ejemplo:

```
for(int i=0;i<replicas.size() ;i++)
{
    if(replicas.get(i).iniciarSesion(nombreUsuario,contraseña))
    {
        numReplica=i;
        System.out.println("\nInicio de sesión completado \n");
        identificado=true;
    }
}
if(!identificado)
{
    System.out.println("\nNo se encuentra registrado en el sistema o el nombre de usuario y contraseña son incorrectos\n");
}
```

**Imagen 23:** En este caso recorreremos el vector de replicas, llamando a cada una al método iniciar sesión.

Una vez identificado el usuario, conocemos la replica a la que pertenece por lo que si queremos llamar al método donar de esa replica bastaría con poner (igual para el resto de funciones).

`replicas.get(numReplica).donar(nombreUsuario,cantidad);`

En resumen, a la hora de identificar al usuario (ya sea mediante registro o inicio de sesión) se va a buscar en el vector de replicas, pero una vez identificada la replica a la que pertenece o se le ha asignado, basta con llamar al método de esa replica específica para que todo funcione correctamente.

De esta forma podemos tener el número de replicas que queramos, que el sistema de donaciones va a funcionar correctamente.