UNIVERSIDAD
DE GRANADA

PRÁCTICA 3: RMI

MEMORIA DE LA PRÁCTICA 3 DE DESARROLLO

DE SISTEMAS DISTRIBUIDOS

Implementación de los ejemplos propuestos en el

guión y el servidor replicado

AUTOR: ÁLVARO LÓPEZ VERGARA

3 Mayo 2024

1. **Introducción:**

En esta memoria vamos documentar la implementación de los ejemplos propuestos en el guión, demostrando que funcionan y explicando su funcionamiento; también trataremos la implementación del servidor replicado, su manual de usuario y algunos ejemplos de ejecución.

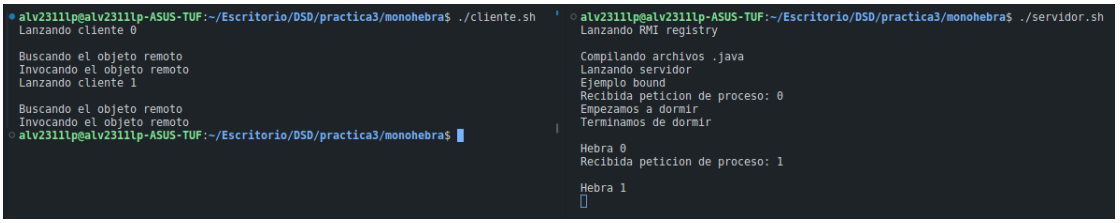
2. **Implementación de los ejemplos:**

Se han implementado los 3 ejemplos propuestos.

2.1. Ejemplo 1

En este ejemplo (una aplicación básica cliente-servidor), el servidor (Ejemplo.java) exporta los métodos contenidos en la interfaz Ejemplo_I.java del objeto remoto instanciado como Ejemplo_I de la clase definida en Ejemplo.java.

El programa servidor, cuando recibe una petición de un cliente, imprime el argumento enviado en la llamada. En caso de ser un "0", realiza un sleep de 5 segundos antes de volver a imprimir el mensaje, en otro caso no realiza esta espera.



2.2. Ejemplo 2

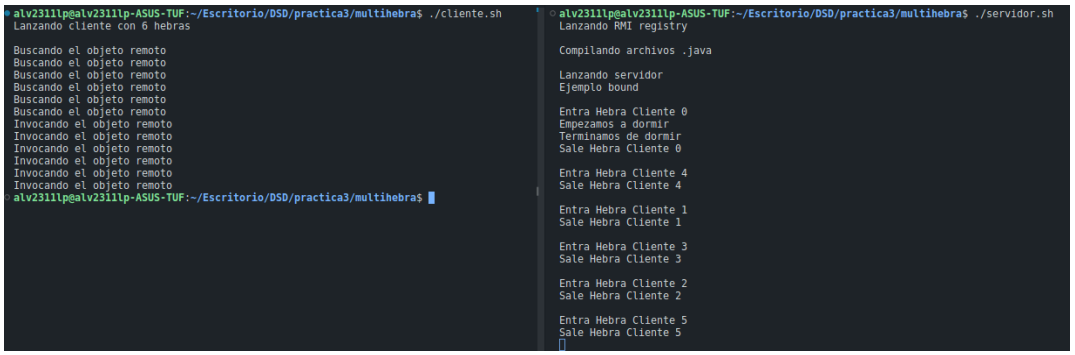
Se trata de un ejemplo similar al anterior pero, en lugar de lanzar varios clientes, creamos múltiples hebras que realizan la misma tarea de imprimir un mensaje remoto accediendo al stub de un objeto remoto. Este ejemplo nos permite ver la gestión de la concurrencia en RMI. En esta ocasión, en vez de pasar al objeto remoto un número entero, pasamos una cadena String.

¿Qué ocurre con las hebras cuyo nombre acaba en 0?¿Qué hacen las demás hebras?¿Se entrelazan los mensajes?

Que realizan un sleep de 5 segundos como en el ejemplo anterior, en la ejecución mostrada mas abajo solo lo hace la 0, pues únicamente se ejecutan 6 hebras, si fuesen 11 o más la hebra con id =10,20… también harían el sleep. El resto escriben el mensaje directamente. Los mensajes se entrelazan o no en función de si hemos puesto el modificador synchronized.

Prueba a introducir el modificador synchronized en el método de la implementación remota: public synchronized void escribir_mensaje (String mensaje) {… y trata de entender las diferencias en la ejecución de los programas.

Con synchronized, no se entrelazan las hebras nas con otras:



En cambio si lo quitamos:

```
Invocando el objeto remoto
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/multihebra$ ./cliente.sh
Lanzando cliente con 6 hebras

Buscando el objeto remoto
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
Invocando el objeto remoto
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/multihebra$
```

```
Lanzando servidor
Ejemplo bound

Entra Hebra Cliente 3

Entra Hebra Cliente 0
Empezamos a dormir

Entra Hebra Cliente 4

Entra Hebra Cliente 2

Entra Hebra Cliente 1
Sale Hebra Cliente 3
Sale Hebra Cliente 4
Sale Hebra Cliente 2
Sale Hebra Cliente 1

Entra Hebra Cliente 5
Sale Hebra Cliente 5
Terminamos de dormir
Sale Hebra Cliente 0
```

2.3. Ejemplo 3

En este ejemplo de contador se crea por una parte el objeto remoto y por otra el servidor. El objeto remoto consta de varias funciones accesibles remotamente. El servidor (servidor.java) exporta los métodos contenidos en la interfaz icontador.java del objeto remoto instanciado como micontador de la clase definida en contador.java.

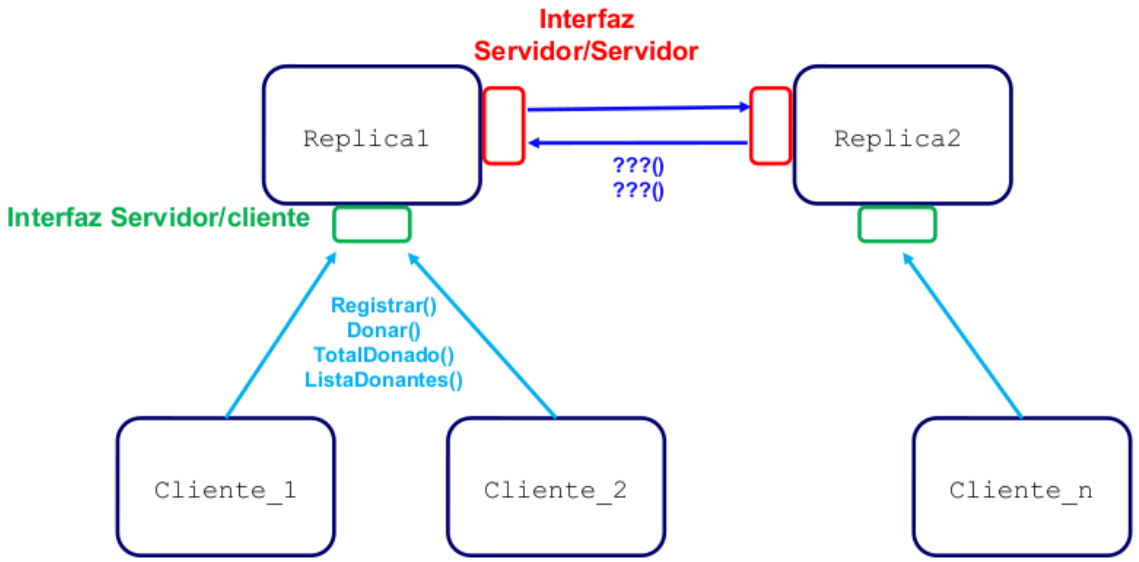
El programa cliente inicializa el contador del servidor a 0 y lo incrementa 1000 veces, luego imprime el tiempo medio de respuesta y el número de llamadas realizadas.

```
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/contador$ ./cliente.sh

Lanzando cliente

Poniendo contador a 0
Incrementando...
Media de las RMI realizadas = 0.092 msegs
RMI realizadas = 1000
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/contador$
```

```
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/contador$ ./servidor.sh
Compilando..
Lanzando servidor
Servidor RemoteException | MalformedURLExceptiondor preparado
```

3. Servidor Replicado

En este ejercicio se ha implementado un sistema cliente-servidor, en el que el servidor es replicado, con 2 o más réplicas, que proporcionan varias y distintas operaciones.



Nuestra implementación consta de varios archivos en los que hemos definido distintas interfaces y clases:

- interfazClienteServidor: interfaz que proporciona las operaciones al cliente
- interfazServidorServidor: interfaz de las operaciones entre las distintas réplicas
- Cliente: proporciona la interfaz por consola y es la que hace las llamadas con RMI
- Servidor: Es la que inicializa las n réplicas, por defecto 2.
- Réplica: Aquí están definidas todas las operaciones entre servidores, y cliente-servidor, esta clase implementa las dos interfaces anteriores.
- ClienteRegistrado: para definir los objetos cliente registrados, con nombre usuario, total donado, y número de donaciones.

Por defecto el cliente siempre se conectara a la réplica 1, salvo que le especifiquemos otra (más abajo manual de usuario). Las operaciones definidas para el cliente son las siguientes:

```
public interface interfazClienteServidor extends Remote {
    public boolean registrarCliente(String nombre, String contraseña) throws RemoteException;
    public boolean iniciarSesionCliente(String nombre, String contraseña) throws RemoteException;
    public boolean donar(String nombre, String contraseña, int n) throws RemoteException;
    public int totalDonado() throws RemoteException;
    public int totalDonadoUsuario(String nombre, String contraseña) throws RemoteException;
    public int numeroDonacionesRealizadas(String nombre, String contraseña) throws RemoteException;
    public String listaDonantes() throws RemoteException;
}
```

```
Selecciona una operación:
    r - Registrarse
    i - Iniciar sesión
    d - Realizar una donación
    e - Ver cantidad donada
    n - Ver cuantas donaciones he realizado
    t - Ver total donado
    l - Ver lista completa de donantes
    c - Cerrar sesion
    s - Salir
```

Por tanto, podemos registrarnos, iniciar sesión y cerrar sesión; solo te puedes registrar una vez, es decir los pares nombre contraseña son únicos. El resto de operaciones son donar, con la que nos pide un valor a donar, ver la cantidad que, yo como usuario actual que he iniciado sesión, he donado; ver el número de donaciones realizadas por mi; y finalmente ver el total donado y la lista de donantes (solo sus nombres).

En cada una de estas opciones el cliente realiza una llamada RMI, a la réplica a la que se conectó inicialmente, sin embargo, como puede estar registrado en cualquier replica, las consultas y demás operaciones se hacen desde la réplica a la que está conectada, comunicándose esta con la réplica que aloja al usuario, para que esta última le de la información necesaria, y la primera se la devuelve al cliente.

Las operaciones entre las réplicas son las siguientes:

```java
public interface interfazServidorServidor extends interfazClienteServidor{
    public void registrarClienteReplica(String nombre, String contraseña) throws RemoteException;
    public void donarReplica(String nombre, String contraseña, int valor) throws RemoteException;
    public int numeroDeRegistrados() throws RemoteException;
    public boolean comprobarRegistro(String nombre, String contraseña) throws RemoteException;
    public ArrayList<ClienteRegistrado> getRegistrados() throws RemoteException;
    public void registarDonacionUsuario(String nombre, String contraseña, int valor) throws RemoteException;
    public boolean getEsDonadorUsuario(String nombre, String contraseña) throws RemoteException;
    public int getTotalDonadoUsuario(String nombre, String contraseña) throws RemoteException;
    public int getNumeroDonacionesUsuario(String nombre, String contraseña) throws RemoteException;
    public int getTotalDonadoLocal() throws RemoteException;
    public int getTotalDonado() throws RemoteException;
    public String listaDonantesLocal() throws RemoteException;
}
```

Estas se usan internamente para las operaciones cliente-servidor, pues si el cliente está alojado en otra réplica, es necesaria la comunicación entre estas.

Vamos a realizar un ejemplo de uso, registramos dos usuarios (se deben de registrar en réplicas distintas aunque nos conectemos solo a la primera), donaremos con ambos y luego miraremos el total dando y la lista de donantes.

```
Lanzando el cliente
Conexión a servidor: replica1
-------------------------------------------
Selecciona una operación:
    r - Registrarse
    i - Iniciar sesión
    d - Realizar una donación
    e - Ver cantidad donada
    n - Ver cuantas donaciones he realizado
    t - Ver total donado
    l - Ver lista completa de donantes
    c - Cerrar sesion
    s - Salir
r
Registro de usuario.
Nombre de usuario: usu1
Contraseña: usu1
Sesión iniciada correctamente
-------------------------------------------
```

```
-------------------------------------------
Selecciona una operación:
    r - Registrarse
    i - Iniciar sesión
    d - Realizar una donación
    e - Ver cantidad donada
    n - Ver cuantas donaciones he realizado
    t - Ver total donado
    l - Ver lista completa de donantes
    c - Cerrar sesion
    s - Salir
d
Cantidad para donar: 45
Donación completada
-------------------------------------------
Selecciona una operación:
```

```
e
usu1 has donado: 45€
-------------------------------
```

```
    c - Cerrar sesion
    s - Salir
r
Registro de usuario.
Nombre de usuario: usu2
Contraseña: usu2
Sesión iniciada correctamente
-------------------------------------
```

```
    c - Cerrar sesion
    s - Salir
d
Cantidad para donar: 120
Donación completada
-------------------------------------
Selecciona una operación:
    r - Registrarse
```

```
    l - Ver lista completa de donantes
    c - Cerrar sesion
    s - Salir
t
Total donado: 165€
-------------------------------------
Selecciona una operación:
    r - Registrarse
```

```
    s - Salir
l
Lista de donantes:
usu1
usu2

-------------------------------------
Selecciona una operación:
```

Como vemos las operaciones se realizan correctamente.

3.1 Manual de usuario

Vamos a comentar cómo usar el ejercicio implementado. Primero, hemos definido unos scripts para hacer más sencilla la ejecución:

cliente.sh: recibe un parametro que es el nombre de la réplica a la que conectarse, por cómo las definimos en el servidor seria de la forma: “replica1” o “replica2”, …

```
$ cliente.sh
1   javac *.java
2   echo
3   echo "Lanzando el cliente"
4   java -cp . -Djava.security.policy=server.policy Cliente $1
```

servidores.sh: lanza el ejecutable de la clase servidor que instala todas las réplicas, por defecto 2, que le pasemos por el primer argumento.

```
$ servidores.sh
1   #!/bin/sh -e
2   echo "Compilando archivos .java"
3   javac *.java
4   sleep 1
5   echo
6   echo "Lanzando servidores"
7   java -cp . -Djava.rmi.server.codebase=file:./ -Djava.rmi.server.hostname=localhost -Djava.security.policy=server.policy Servidor $1
```

Nota: en ambos scripts compilamos, pero bastaría con hacerlo en el servidor, mientras no cambie la implementación de los archivos java.

Ejecutando ambos script (primero el servidor), ya tendremos nuestro programa en ejecución, en el que nos aparece la interfaz vista anteriormente:

```
Servidor lanzado.
^C
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/servidor_donaciones$ ./servidores.sh
Compilando archivos .java

Lanzando servidores
Servidor lanzado.
```

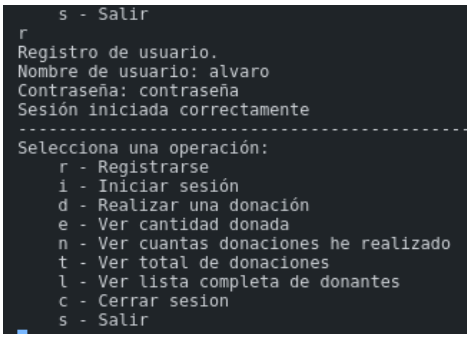
```
alv2311lp@alv2311lp-ASUS-TUF:~/Escritorio/DSD/practica3/servidor_donaciones$ ./cliente.sh

Lanzando el cliente
Conexión a servidor: replica1
-----------------------------------------------
Selecciona una operación:
    r - Registrarse
    i - Iniciar sesión
    d - Realizar una donación
    e - Ver cantidad donada
    n - Ver cuantas donaciones he realizado
    t - Ver total de donaciones
    l - Ver lista completa de donantes
    c - Cerrar sesion
    s - Salir
```

A partir de aquí, lo primero será registrarse pues si no lo hacemos nos dirá que es obligatorio para el resto de operaciones.

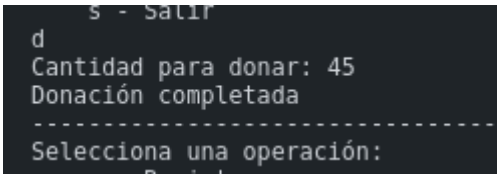
```
    l - Ver lista completa de donantes
    c - Cerrar sesion
    s - Salir
d
Hay que estar registrado para donar
-----------------------------------------
Selecciona una operación:
    r - Registrarse
```

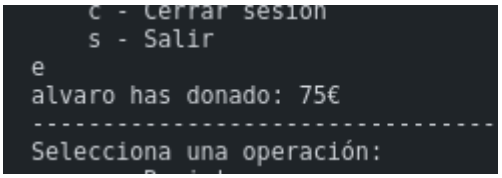
```
    c - Cerrar sesion
    s - Salir
t
Hay que estar registrado para ver el total donado
-------------------------------------------------
Selecciona una operación:
```

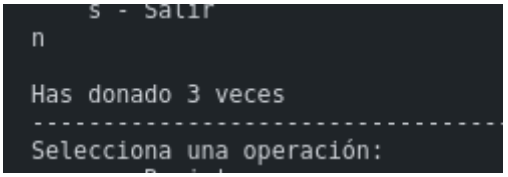
Podemos hacerlo pulsando la r, donde introducimos por teclado nombre y contraseña, independientemente de si estamos registrados ya o no, iniciará sesión automáticamente, dando acceso al resto de operaciones.

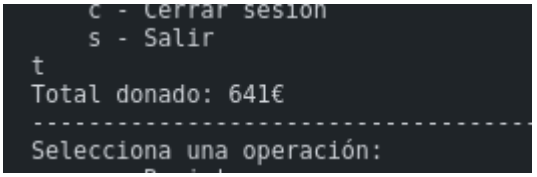


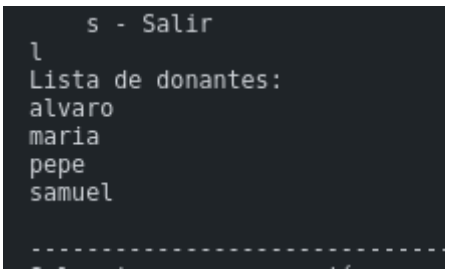
Para el resto de operaciones basta con introducir la letra correspondiente y, en el caso de donar, un valor numérico:



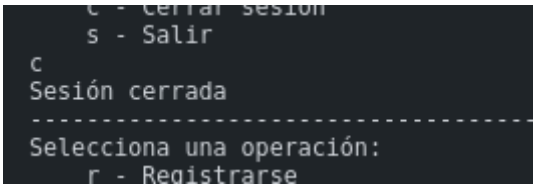


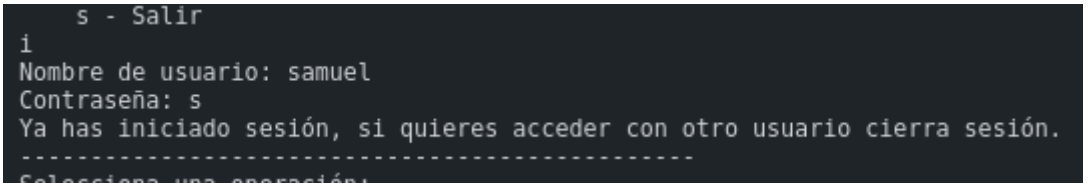






Además en cualquier momento podemos cerrar sesión e iniciar sesión de nuevo con cualquier usuario registrado.





4. **Anexos:**

Junto con esta memoria se incluye todo el código fuente.