Práctica 3: RMI

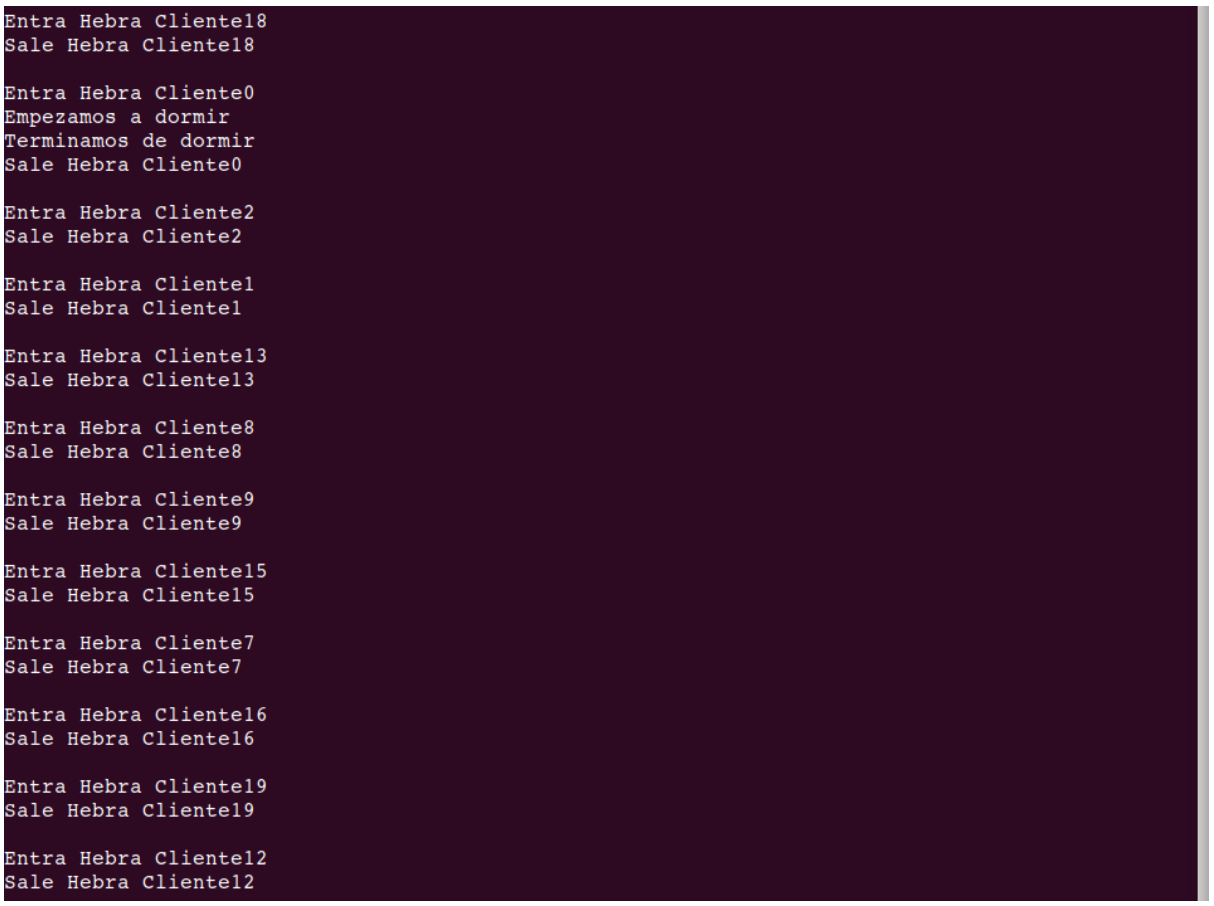
Ejercicio 1.

La primera parte de la entrega consiste en implementar los ejemplos mostrados en el guión de práctica y comentar sus diferencias. Durante la ejecución los ejemplos me encontré con el siguiente problema: el puerto por defecto, 1099, tras intentar ejecutar alguno de los ejemplos una vez, seguía en uso por el proceso lanzado, que no terminaba. Para solucionarlo, localizo el proceso que está utilizando dicho puerto con el comando netstat y lo termino antes de volver a intentar ejecutar alguno de los ejemplos.

He creado dos carpetas, RMIMultiHebra y RMIMultiHebra_syn, en los que se incluye el ejemplo multihebrado y su variación sincronizada respectivamente. La única diferencia apreciable a simple vista es que, en la versión síncrona, una vez entra una hebra, el servidor no atenderá a ninguna otra hasta que ésta salga, por lo que cuando una hebra acabada en 0 comienza a dormir, el servidor dormirá también hasta que esta termine.

```
Entra Hebra Cliente7
Sale Hebra Cliente7

Entra Hebra Cliente4
Sale Hebra Cliente4

Entra Hebra Cliente10
Empezamos a dormir

Entra Hebra Cliente13
Sale Hebra Cliente13

Entra Hebra Cliente1
Sale Hebra Cliente1

Entra Hebra Cliente15

Entra Hebra Cliente2
Sale Hebra Cliente2

Entra Hebra Cliente3
Sale Hebra Cliente3

Entra Hebra Cliente17

Entra Hebra Cliente11
Sale Hebra Cliente11
Sale Hebra Cliente17

Entra Hebra Cliente9
Sale Hebra Cliente9

Entra Hebra Cliente16
Sale Hebra Cliente16
Sale Hebra Cliente15

Entra Hebra Cliente14
Sale Hebra Cliente14
Terminamos de dormir
Sale Hebra Cliente0
Terminamos de dormir
Sale Hebra Cliente10
```

Ejecución del servidor multihebrado no síncrono para 20 hebras.



Ejecución del servidor multihebrado síncrono para 20 hebras.

Ejercicio 2.

El ejercicio 2 consiste en realizar un servidor replicado que reciba donaciones de sus clientes y cumpla una serie de requisitos. Para realizar el ejercicio me he inspirado en el último ejemplo a desarrollar del apartado anterior en el que se desarrollaba un objeto remoto contador.

- Estructura: en la carpeta RMIReplica se encuentran todos los códigos fuente del ejercicio: la interfaz icontador, contador, cliente y servidor. Además se han añadido unos script como los facilitados en el guión de práctica para facilitar su ejecución.
- Funcionamiento del servidor: La primera vez que lanzamos el servidor, éste creará un nuevo registro. Si lo volvemos a lanzar, como el registro en el puerto seleccionado (1234) ya habrá sido creado, lanzamos una réplica (utiliza el registro ya creado y realiza las mismas operaciones que el primer servidor). Ambas réplicas lanzan su propia instancia del contador.
- Funcionamiento del cliente: Al lanzar el cliente, éste se conectará a la primera réplica y en ella realizará su registro. En caso de que su registro se realice en la otra réplica, nos conectaremos a ella. A partir de este momento comienza un bucle en el que aparecen varias opciones en forma de menú: donar, consultar mis donaciones, consultar recaudaciones de esta réplica y consultar el total recaudado. El programa

leerá por teclado la selección del usuario y realizará la comunicación necesaria con la réplica.

- Icontador y contador: se han implementado varias funciones que permiten satisfacer los requisitos de este ejercicio, como son: registrar un cliente, donar, consultar las donaciones del cliente, consultar lo recaudado en la réplica y lo recaudado en total. Las interacciones entre réplicas también se realizan mediante operaciones del contador, como por ejemplo, al registrar un cliente, la decisión de en qué servidor se realizará (para ello será necesario pasarle como argumento el registro).

Durante la elaboración de la práctica no me he encontrado con ningún problema aparte del planteamiento inicial y algún fallo a la hora de escribir el código que ha hecho que la práctica me lleve algo más de tiempo. La única información adicional que he tenido que investigar ha sido la forma de referenciar a los distintos contadores del registro (`registro.list()[índice del contador]`).

A continuación se muestra algunos ejemplos del funcionamiento básico:

Réplica 0

```
Lanzando el servidor
Réplica 0 lanzada
Servidor RemoteException | MalformedURLExceptiondor preparado
El cliente 49087324 ha sido registrado en 0.
Clientes registrados: 49087324
El cliente 41947294 ha sido registrado en 0.
Clientes registrados: 49087324 41947294
El cliente 49087324 ha donado 200.
Ya van 200€ donados en réplica 0.
El cliente 41947294 ha donado 400.
Ya van 600€ donados en réplica 0.
```

Réplica 1

```
Lanzando el servidor
Lanzando réplica
Réplica 1 lanzada
Servidor RemoteException | MalformedURLExceptiondor preparado
El cliente 17930426 ha sido registrado en 1.
Clientes registrados: 17930426
El cliente 17930426 ha donado 100.
Ya van 100€ donados en réplica 1.
```

Cliente 41947294

```
Qué desea hacer?
1. Donar
2. Consultar mis donaciones
3. Consultar las recaudaciones de esta réplica
4. Consultar el total recaudado
5. Salir

1
Cuánto desea donar?
400
Donación exitosa.

Qué desea hacer?
1. Donar
2. Consultar mis donaciones
3. Consultar las recaudaciones de esta réplica
4. Consultar el total recaudado
5. Salir

2
Has donado 400 en total.

Qué desea hacer?
1. Donar
2. Consultar mis donaciones
3. Consultar las recaudaciones de esta réplica
4. Consultar el total recaudado
5. Salir

3
Se ha recaudado 600 en esta réplica.

Qué desea hacer?
1. Donar
2. Consultar mis donaciones
3. Consultar las recaudaciones de esta réplica
4. Consultar el total recaudado
5. Salir

4
Se ha recaudado 700 en total.

Qué desea hacer?
1. Donar
2. Consultar mis donaciones
3. Consultar las recaudaciones de esta réplica
4. Consultar el total recaudado
5. Salir
```

Como funcionamiento adicional se ha añadido la posibilidad de ver cuál ha sido el cliente que más dinero ha donado en todo el servidor. Para esto se han añadido nuevas funciones al contador: buscar al cliente que más ha donado en una réplica y comparar entre los "ganadores" de cada réplica.

```
Qué desea hacer?
1. Donar
2. Consultar mis donaciones
3. Consultar las recaudaciones de esta réplica
4. Consultar el total recaudado
5. Consultar el cliente más generoso
6. Salir

5
El cliente 49087324 ha sido el cliente más generoso donando un total de 1000!
```

Algunas anotaciones:

- La identificación de los clientes se haría por medio del número de DNI, sin la letra. En el script para la ejecución del cliente he añadido el mío pero a la hora de lanzar otro cliente habría que modificarlo. La forma de lanzarlo desde la terminal sería:
  `java -cp . -Djava.rmi.server.codebase=file:../ -Djava.security.policy=server.policy cliente [dni]`
- Para facilitar la comprobación del funcionamiento del ejercicio, he supuesto que un cliente solo puede iniciar sesión una única vez, pues cada vez que se lanza un cliente intenta registrarse. La otra opción que he dejado comentada permitiría al cliente iniciar sesión todas las veces que quisiera, y aunque intentara registrarse, su identificador no se volvería a almacenar. Si se desea que su funcionamiento sea el segundo mencionado, bastaría con ir al archivo fuente del contador, descomentar las líneas 38 y 39 y comentar la línea 41.

```
// Si el cliente ya estuviera registrado en alguna réplica
// Opción comentada: el cliente puede iniciar sesión varias veces
/*if (esCliente(dni)) return replica;
else if (otra.esCliente(dni)) return index;*/
// Opción no comentada: el cliente solo puede iniciar sesión una vez
if (esCliente(dni) || otra.esCliente(dni)) return -1;
```

Paula Cumbreras Torrente
paulacumbreras@correo.ugr.es
Grupo 2