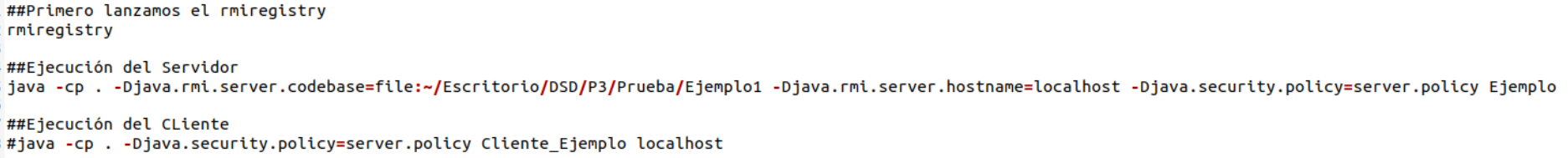
Manuel Contreras Orge

MEMORIA PRACTICA 3

# Ejemplo 1:

En este ejemplo se ha realizado un sencillo programa de comunicación RMI en el, que se comunican un cliente (el programa *Cliente\_Ejemplo*) y un servidor (el programa *Ejemplo*, cuya declaración se hace en el programa *Ejemplo\_I*). En este programa se produce una conexión entre los dos elementos señalados antes, en el que se usa la hebra que se le mete como dato de entrada en la llamada, con la excepción de que si se quiere usar la hebra número 0 el programa se duerme durante 5s (5000ms).

Para usar el programa, primero debemos llamar al *rmiregistry*, después se debe de ejecutar la orden del servidor y, por último, se llama al cliente, llamando en la orden a la hebra que se quiere usar.

Órdenes usadas en este ejemplo.

Texto

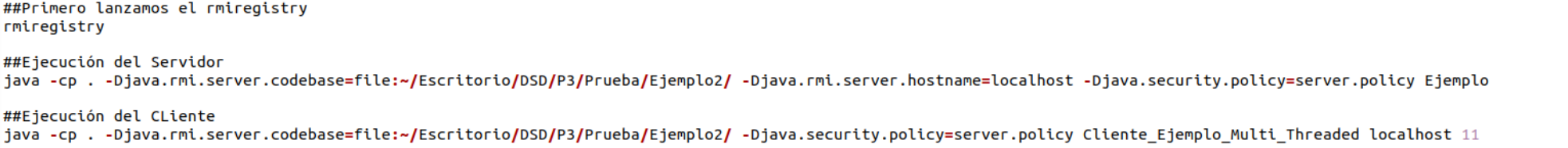
Descripción generada automáticamenteEn esta imagen se observa cómo se funciona correctamente el ejemplo, tanto con la hebra 0 como con otra hebra diferente.

# Ejemplo 2:

## Sin synchronized:

En este ejemplo se realiza un ejemplo de como sería el ejercicio explicado en el Ejemplo 1, pero con multi hebras, tantas como se quiera usar. En este caso también se usan los archivos *Cliente\_Ejemplo* y *Ejemplo* como cliente y servidor, respectivamente, y en el cliente se va a dormir cuando se va a ejecutar una hebra que acabe por 0, es decir, que sea divisor de 10.

Para el uso del programa, también es casi igual que el anterior, se llama a *rmiregistry*, se llama al servidor y se llama al cliente, pero con la única diferencia de que ahora en la llamada se dice la cantidad de hebras que se va a usar, empezando con la hebra número 0.

Órdenes usadas en este ejemplo.

Texto

Descripción generada automáticamenteEn la imagen se ve como se ejecuta el programa con multihebra, en este caso se hace con 11 hebras, quedándose dormido con las hebras 0 y 10.

## Con synchronized:

En esta modificación del ejercicio hemos añadido la cláusula *synchronized* después del *public* en el archivo *Ejemplo*, con este cambio lo que se hace es que, como se puede ver en la imagen de abajo, cuando se entra en una hebra, hasta que no se salga de dicha hebra, no se puede entrar en otra (también se hace la espera de 5 segundos cuando se entra en una hebra terminada en 0).

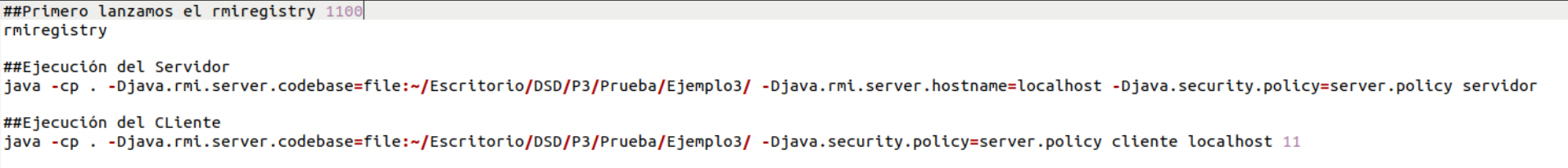
Gráfico, Texto

Descripción generada automáticamenteEn esta imagen se ve la salida del ejercicio de multi hebras con la cláusula *synchronized*.

# Ejemplo 3:

En este último ejemplo se realiza un programa cliente-servidor mediante un objeto remoto y un servidor, que exporta sus métodos en una interfaz *icontador* de la clase *contador*. Este programa realiza 3 acciones principales: inicializa el contador del servidor, incrementa el contador 1000 veces mediante el método *incrementar* e imprime el valor de final del contador, junto con el tiempo medio de respuesta del programa en esta ejecución.

Para el uso del programa, llamamos primero a *rmiregistry*, pero esta vez hay que asignarle un valor al puerto, ya que el que se pone por defecto, el 1099, que ya está usado y al llamar al servidor da un error, por ejemplo podemos poner el valor 1100, después llamamos al servidor, usando la misma orden que con las multi hebras, pero esta vez el fichero al que se invoca se llama *servidor*, y por último, llamamos al cliente, cambiando también el nombre del fichero a cliente.

Órdenes usadas en este ejemplo.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamenteError del puerto explicado

Texto

Descripción generada automáticamenteEjemplo de uso del ejemplo usando 11 hebras en la llamada.

# Ejercicio:

En este ejercicio se tiene que hacer un servidor con dos réplicas en las que los clientes pueden donar una cantidad de dinero. Las condiciones requeridas para realizar el ejercicio son: solo podrá donar un cliente que esté registrado en una de las réplicas existentes, un cliente solo se podrá registrar en más de una réplica y un cliente solicitará registrarse en la réplica que quiera, pero el sistema lo registrará en la réplica con menos usuarios registrados, en caso de que no haya ninguna con menos usuarios, se le asignará la que se quiera.

Para realizar la práctica encontramos, además de los ficheros *cliente* y *servidor*, los archivos *replica* (y el archivo declaración de sus funciones *Ireplica*) y *donación* (junto a *Idonacion*). En el fichero *donacion* se realiza la donación de dinero, conectándose con la réplica correspondiente, el archivo *replica* en el que se encuentra se realizan las acciones de registrar a los usuarios, la donación en cada una de las réplicas y la salida por pantalla de los datos necesarios que se piden (como la cantidad de usuarios registrados y la cantidad donada); después nos encontramos con el archivo *server*, que se encarga de iniciar a las réplicas de nuestro sistema, y además mostrará por pantalla cuando donarán los clientes y en que réplica se hace; por último, nos encontramos con el archivo *cliente*, en el que se introduce la réplica en la que se van a conectar los clientes, la cantidad de estos y se realiza la donación por parte de estos clientes, esto se hará con hebras, una para cada réplica, mediante el método run en este archivo.

Para usar este ejercicio, usaremos las mismas órdenes que en el Ejemplo 2 pero el rmiregistry igual que en el Ejemplo 3, es decir, las órdenes que usaremos son: se llama a *rmiregistry* 1100, se llama al servidor y se llama al cliente diciendo la cantidad de hebras que se va a usar, empezando con la hebra número 0.

Texto

Descripción generada automáticamente con confianza bajaÓrdenes usadas en el ejercicio.

Texto

Descripción generada automáticamente Ejemplo de uso del ejemplo usando 11 hebras en la llamada.