SERVIDOR CON MÚLTIPLES CLIENTES SIMULTÁNEOS.

Hasta ahora los programas servidores vistos sólo son capaces de atender a un cliente en cada momento, pero lo más típico es que un programa servidor pueda atender a muchos clientes simultáneamente. La solución para poder atender a múltiples clientes está en el **multihilo,** cada cliente será atendido en un hilo.

El esquema básico en sockets TCP sería construir un único servidor con la clase **ServerSocket** e invocar al método *accept()* para esperar las peticiones de conexión de los clientes. Cuando un cliente se conecta, el método *accept()* devuelve un objeto **Socket**, éste se usará para crear un hilo cuya misión es atender a este cliente. Después se vuelve a invocar a *accept()* para esperar a un nuevo cliente; habitualmente la espera de conexiones se hace dentro de un bucle infinito:

```
import java.io.*;
import java.net.*;

public class Servidor {
    public static void main(String args[]) throws IOException{
        ServerSocket servidor;
        servidor = new ServerSocket(6000);
        System.out.println(" Servidor iniciado ... ");
        while (true) {
            Socket cliente = new Socket();
            cliente=servidor.accept();//esperando cliente
            HiloServidor hilo = new HiloServidor(cliente);
            hilo.start (); // se atiende al cliente
        }
    }
}
```

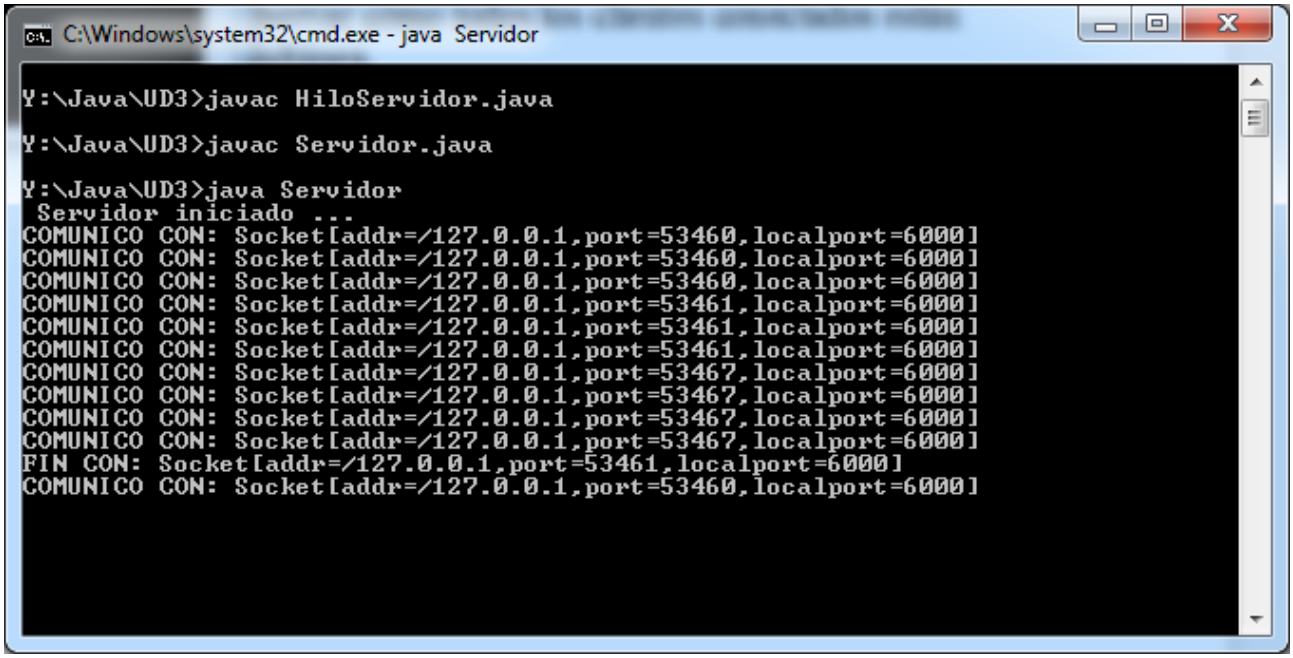
Todas las operaciones que sirven a un cliente en particular quedan dentro de la clase hilo. El hilo permite que el servidor se mantenga a la escucha de peticiones y no interrumpa su proceso mientras los clientes son atendidos.
Por ejemplo, supongamos que el cliente envía una cadena de caracteres al servidor y el servidor se la devuelve en mayúsculas, hasta que recibe un asterisco que finalizará la comunicación con el cliente. El proceso de tratamiento de la cadena se realiza en un hilo, en este caso se llama *HiloServidor:*

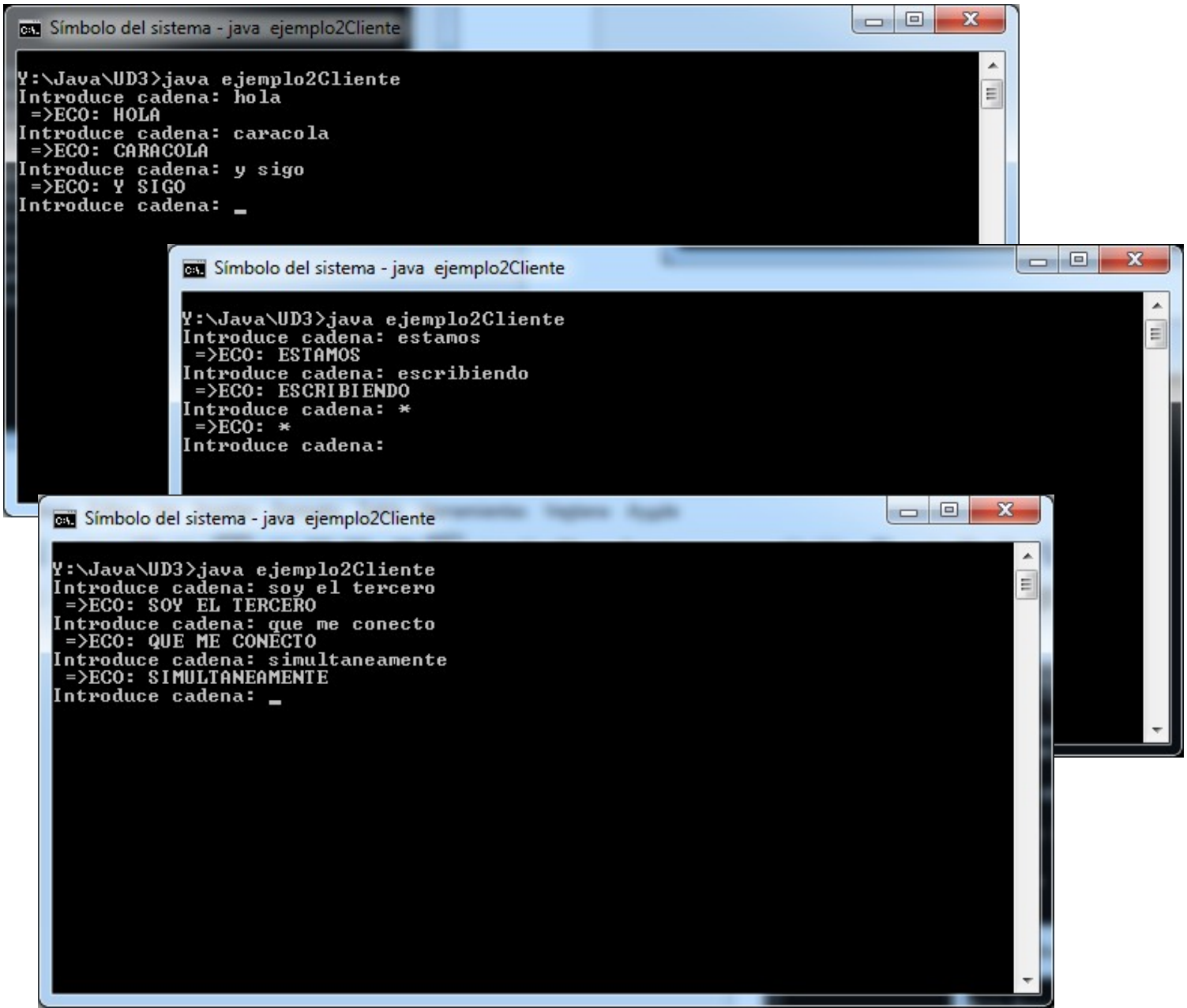
```
import java.io.*;
import java.net.*;
    public class HiloServidor extends Thread{
        BufferedReader fentrada;
        PrintWriter fsalida;
        Socket socket = null;

        public HiloServidor(Socket s){//CONSTRUCTOR
            socket =s;
            //se crean flujos de entrada y salida
            try{
                fsalida = new PrintWriter(socket.getOutputStream(),true);//True significa que
                //el bufer de salida se vacía
                fentrada = new BufferedReader(new InputStreamReader(socket.getInputStream())) ;
                }catch( IOException e ) {
            System.out.println( "Excepción de entrada/salida" );
        }
        }

        public void run() {//tarea a realizar con el cliente
            String cadena="";
            try{
                while (!cadena.trim().equals("*")) {
                    System.out.println("COMUNICO CON: "+ socket.toString()) ;
                    cadena = fentrada.readLine();//obtener cadena
                    fsalida.println(cadena.trim().toUpperCase());//enviar mayúscula
                }// fin while
                System.out.println("FIN CON: "+ socket.toString());
                fsalida.close() ;
                fentrada.close() ;
                socket.close() ;
                }catch( IOException e ) {
            System.out.println( "Excepción de entrada/salida" );
            }
        }
}
```

Como programa cliente podemos ejecutar el programa *ejemplo2Cliente.java* que se creó anteriormente. El programa se conectará con el servidor en el puerto 6000 y le enviará cadenas introducidas por teclado; cuando le envíe un asterisco el servidor finalizará la comunicación con el cliente. El cliente finaliza cuando se detien la entrada de datos mediante las teclas Ctrl+C o Ctrl+Z.

```java
import java.io.*;
import java.net.*;
public class ejemplo2Cliente {
    public static void main(String[] args) throws IOException {
        String Host = "localhost";
        int Puerto = 6000;// puerto remoto
        try{
            Socket Cliente = new Socket(Host, Puerto);

            // CREO FLUJO DE SALIDA AL SERVIDOR
            PrintWriter fsalida = new PrintWriter(Cliente.getOutputStream(), true);

            // CREO FLUJO DE ENTRADA AL SERVIDOR
            BufferedReader fentrada = new BufferedReader(new InputStreamReader(Cliente.getInputStream()));

            // FLUJO PARA ENTRADA ESTANDAR
            BufferedReader in = new BufferedReader(new InputStreamReader(System.in));

            String cadena, eco="";
            System.out.print("Introduce cadena: ");
            cadena = in.readLine();//lectura por teclado
            while(cadena !=null) {
                fsalida.println(cadena); //envio cadena al servidor
                eco=fentrada.readLine(); //recibo cadena del servidor
                System.out.println(" =>ECO: "+eco);
                System.out.print("Introduce cadena: ");
                cadena = in.readLine();//lectura por teclado
            }
            fsalida. close () ;
            fentrada.close();
            System.out.println("Fin del envio ... ");
            in.close();
            Cliente.close();
        }catch (IOException e) {
            e.printStackTrace();
        }
    }//
}//
```

A continuación se muestra un momento de la ejecución, primero se ejecuta el programa servidor ya continuación el programa cliente; se puede observar cómo todos los clientes conectados están siendo atendidos por el servidor de forma simultánea.


C:\Windows\system32\cmd.exe - java Servidor
Y:\Java\UD3>javac HiloServidor.java
Y:\Java\UD3>javac Servidor.java
Y:\Java\UD3>java Servidor
Servidor iniciado ...
COMUNICO CON: Socket[addr=/127.0.0.1,port=53460,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53460,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53460,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53461,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53461,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53461,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53467,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53467,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53467,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53467,localport=6000]
FIN CON: Socket[addr=/127.0.0.1,port=53461,localport=6000]
COMUNICO CON: Socket[addr=/127.0.0.1,port=53460,localport=6000]


Símbolo del sistema - java ejemplo2Cliente
Y:\Java\UD3>java ejemplo2Cliente
Introduce cadena: hola
=>ECO: HOLA
Introduce cadena: caracola
=>ECO: CARACOLA
Introduce cadena: y sigo
=>ECO: Y SIGO
Introduce cadena:
Símbolo del sistema - java ejemplo2Cliente
Y:\Java\UD3>java ejemplo2Cliente
Introduce cadena: estamos
=>ECO: ESTAMOS
Introduce cadena: escribiendo
=>ECO: ESCRIBIENDO
Introduce cadena: *
=>ECO: *
Introduce cadena:
Símbolo del sistema - java ejemplo2Cliente
Y:\Java\UD3>java ejemplo2Cliente
Introduce cadena: soy el tercero
=>ECO: SOY EL TERCERO
Introduce cadena: que me conecto
=>ECO: QUE ME CONECTO
Introduce cadena: simultaneamente
=>ECO: SIMULTANEAMENTE
Introduce cadena: