Ejemplos de sockets

Los sockets son un mecanismo de comunicación entre procesos de muy bajo nivel, donde el usuario tiene que hacer muchas cosas, pero que siguen siendo muy importantes. En esta práctica se muestran dos implementaciones. El socket servidor va en un proceso y el cliente (o clientes) en otros. Cada proceso puede estar en una máquina diferente o pueden compartir la computadora. Cuando el servidor está en la misma máquina del cliente, se usa url= "localhost" o "127.0.0.0". Cuando está en otra máquina se usa su nombre o dirección IP. En el centro de cómputo las direcciones corresponden a la cuadrícula de numeración de máquinas, como por ejemplo "E12".

Para los dos casos que siguen, primero cargue las dos clases en un proyecto. Colóquese en el servidor y mándelo a correr usando localhost. Luego cambie al cliente y láncelo. Alternativamente use dos copias de Eclipse o extraiga el servidor y láncelo desde MSDOS (ver al final).

Luego lleve el cliente a otra máquina y ejecute desde ahí usando el nombre de la máquina del servidor.

Ejemplo 1.

```
import java.io.*;
import java.net.*;
public class ServidorTCP {

	public static void main(String[] args) {
		String lin, maylin;
		try{
			int port =1234;
			ServerSocket welcome = new ServerSocket(port);
			while (true){
				System.out.println("Esperando solicitudes en puerto "+port);
				Socket conecta = welcome.accept(); //espera que alguien quiera conectarse
				System.out.println("Conectado a "+conecta.getInetAddress()+":"+conecta.getPort());
				BufferedReader entra = new BufferedReader(
						new InputStreamReader(conecta.getInputStream()));
				DataOutputStream sale = new DataOutputStream(conecta.getOutputStream());
				lin = entra.readLine();
				while (lin != null){
					System.out.println("Recibido: "+lin);
					maylin = lin.toUpperCase()+"\n";
					sale.writeBytes(maylin);
					lin = entra.readLine();
```

```
                        }
                    }
                }catch(Exception e){e.printStackTrace();}
        }
}
```

```
import java.io.*;
import java.net.*;
public class ClienteTCP {

        public static void main(String[] args) {
                String lin, maylin, host;
                int port;

                try{
                        System.out.println("args 0: "+args[0]+"| args 1: "+args[1]);
                        port =Integer.parseInt(args[1]);
                        host = args[0];
                }catch (Exception e){
                        System.out.println("Como hubo problemas en parámetros, usamos el localhost");
                        host="localhost";
                        port = 1234;
                }
                        try{
                                BufferedReader consola = new BufferedReader(new InputStreamReader(System.in));

                        Socket cliente = new Socket(host, port);
                        DataOutputStream sale = new DataOutputStream(cliente.getOutputStream());
                        BufferedReader entra = new BufferedReader(
                                        new InputStreamReader(cliente.getInputStream()));
                        System.out.print("Deme mensaje:");
                        System.out.flush();
                        lin = consola.readLine();
                        while (!lin.equals(".")){
```

```
                                sale.writeBytes(lin+"\n");
                                maylin = entra.readLine();
                                System.out.println("Recibido: "+maylin);
                                System.out.print("Deme mensaje:");
                                System.out.flush();
                                lin = consola.readLine();
                                }
                        cliente.close();
                }catch(Exception e){e.printStackTrace();}
        }
}
```

Desde el cliente, después de establecer contacto, se escriben líneas de texto que recibirá el servidor

Ejemplo 2

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

public class RemoteFileServer {
        protected int listenPort = 3000;
        private boolean alto = false;
        public static void main(String[] args) {
                System.out.println("Servidor de sockets: main");
                RemoteFileServer server = new RemoteFileServer();
                server.acceptConnections();
        }

        public void acceptConnections() {
                System.out.println("Servidor de sockets: inicia accept");
                try {
                        ServerSocket server = new ServerSocket(listenPort);
                        Socket incomingConnection = null;
                        while (!alto) {
                                System.out.println("Servidor de sockets: ciclo accept");
```

```
                        incomingConnection = server.accept();
                        handleConnection(incomingConnection);
                }
        } catch (BindException e) {
                System.out.println("Unable to bind to port " + listenPort);
        } catch (IOException e) {
                System.out.println("Unable to instantiate a ServerSocket on port: " + listenPort);
        }
}

public void handleConnection(Socket incomingConnection) {
        System.out.println("Servidor de sockets: handle inicia");
        try {
                System.out.println("Servidor de sockets: handle creará archivos");
                OutputStream outputToSocket = incomingConnection.getOutputStream();
                InputStream inputFromSocket = incomingConnection.getInputStream();
                BufferedReader streamReader =
                        new BufferedReader(new InputStreamReader(inputFromSocket));
                FileReader fileReader = new FileReader(new File(streamReader.readLine()));
                BufferedReader bufferedFileReader = new BufferedReader(fileReader);
                PrintWriter streamWriter =
                        new PrintWriter(incomingConnection.getOutputStream());
                String line = null;
                System.out.println("Servidor de sockets: handle maneja texto");
                while ((line = bufferedFileReader.readLine()) != null) {
                        streamWriter.println(line);
                        System.out.println("Recibí: "+line);
                        if (line.equals("ALTO")){
                                System.out.println("Me mandan parar");
                                alto=true;
                                break;
                        }
                }
```

```
                        fileReader.close();
                        streamWriter.close();
                        streamReader.close();
                } catch (Exception e) {
                        System.out.println("Error handling a client: " + e);
                }
        }

}
```

```
import java.io.*;
import java.net.*;
public class RemoteFileClient {
protected String hostIp;
protected int hostPort;
protected BufferedReader socketReader;
protected PrintWriter socketWriter;
public RemoteFileClient(String aHostIp, int aHostPort) {
hostIp = aHostIp;
hostPort = aHostPort;
}
public static void main(String[] args) {
        RemoteFileClient remoteFileClient = new RemoteFileClient("localhost", 3000); //"127.0.0.1"
        remoteFileClient.setUpConnection();
        String fileContents =
        remoteFileClient.getFile("RemoteFile.txt");  //"C:\\WINNT\\Temp\\RemoteFile.txt"
        remoteFileClient.tearDownConnection();
        System.out.println(fileContents);
        }

public void setUpConnection() {
        try {
                Socket client = new Socket(hostIp, hostPort);
                socketReader = new BufferedReader(
```

```
                new InputStreamReader(client.getInputStream()));
                socketWriter = new PrintWriter(client.getOutputStream());
                } catch (UnknownHostException e) {
                System.out.println("Error setting up socket connection: unknown host at " + hostIp); } catch (IOException e) {
                System.out.println("Error setting up socket connection: " + e);
                }}

public String getFile(String fileNameToGet) {
        StringBuffer fileLines = new StringBuffer();
        try {
        socketWriter.println(fileNameToGet);
        socketWriter.flush();
        String line = null;
        while ((line = socketReader.readLine()) != null)
        fileLines.append(line + "\n");
        } catch (IOException e) {
        System.out.println("Error reading from file: " + fileNameToGet);
        }
        return fileLines.toString();
}
public void tearDownConnection() {
        try {
                socketWriter.close();
                socketReader.close();
                } catch (IOException e) {
                System.out.println("Error tearing down socket connection: " + e);
                }
}
}
```

En la raíz del proyecto de la máquina donde esté el servidfor, crear un archivo “RemoteFile.txt” y escriba lo que quiera. En la última línea escriba la palabra ALTO

Nota: para hacer ejecutables los programas desarrollados en Eclipse, construya una jarra como sigue:

1. Asegúrese que el programa no tiene errores y de Save All
2. Vaya al menú File y elija Export; Si aparece “Java Runnable JAR File” haga clic; si no, de clic en la opción Java y luego en la de Runnable Jar. Haga clic en Next.
3. En “launch configuration” elija el proyecto y la clase donde está el main que le interesa. (Clase principal)
4. En “export destination” elija el folder donde desea le dejen la jarra y anote el nombre con que se ejecutará (suponer “LaJarra”)
5. De Finish
6. En el folder indicado, puede ejecutar el archivo “LaJarra.jar” dando doble clic. Sólo sirve si no usa la consola.
7. Otra forma es entrar a MSDOS y cambiar al folder donde está la jarra y ahí lanzarla como “java –jar LaJarra.jar”
8. Si hay archivos auxiliares en la raíz del proyecto, deben colocarse en la misma carpeta de la jarra.

Las jarras son como archivos compactados que pueden abrirse con zip, 7zip o winrar.

Existe un plugin FatJar que permite elegir mejor las opciones de ejecución.

También existe la opción estándar de Sun/Oracle, pero no es muy buena.