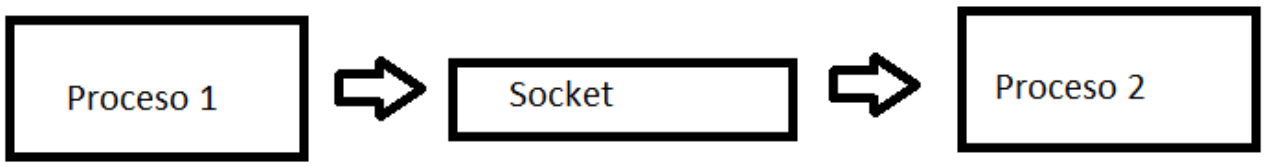
Sockets

Un Socket es un punto de conexión (Una Tubería o un Canal) entre 2 procesos e identificado por una IP y un Puerto. En este caso serán 2 procesos ejecutándose simultáneamente conectados por un socket.



Un socket funciona de una forma similar a la de los ficheros.

Cliente

```
Socket Socket = new Socket("127.0.0.1", 5000)

BufferedReader br = null;
PrintWriter pw = null;

br = new BufferedReader(new InputStreamReader(socket.getInputStream()))//Buffer de lectura
pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()))//Buffer de Escritura

br.readLine();
pw.println("texto a enviar");//Texto a mandar por el Socket
pw.flush();//Similar al Fflush stdin de C OJO
pw.push();//
```

Servidor

```
//Primero creamos el socket
ServerSocket serverSocet = new ServerSocket(2013)//Indicamos el puerto de escucha, en este caso 2013

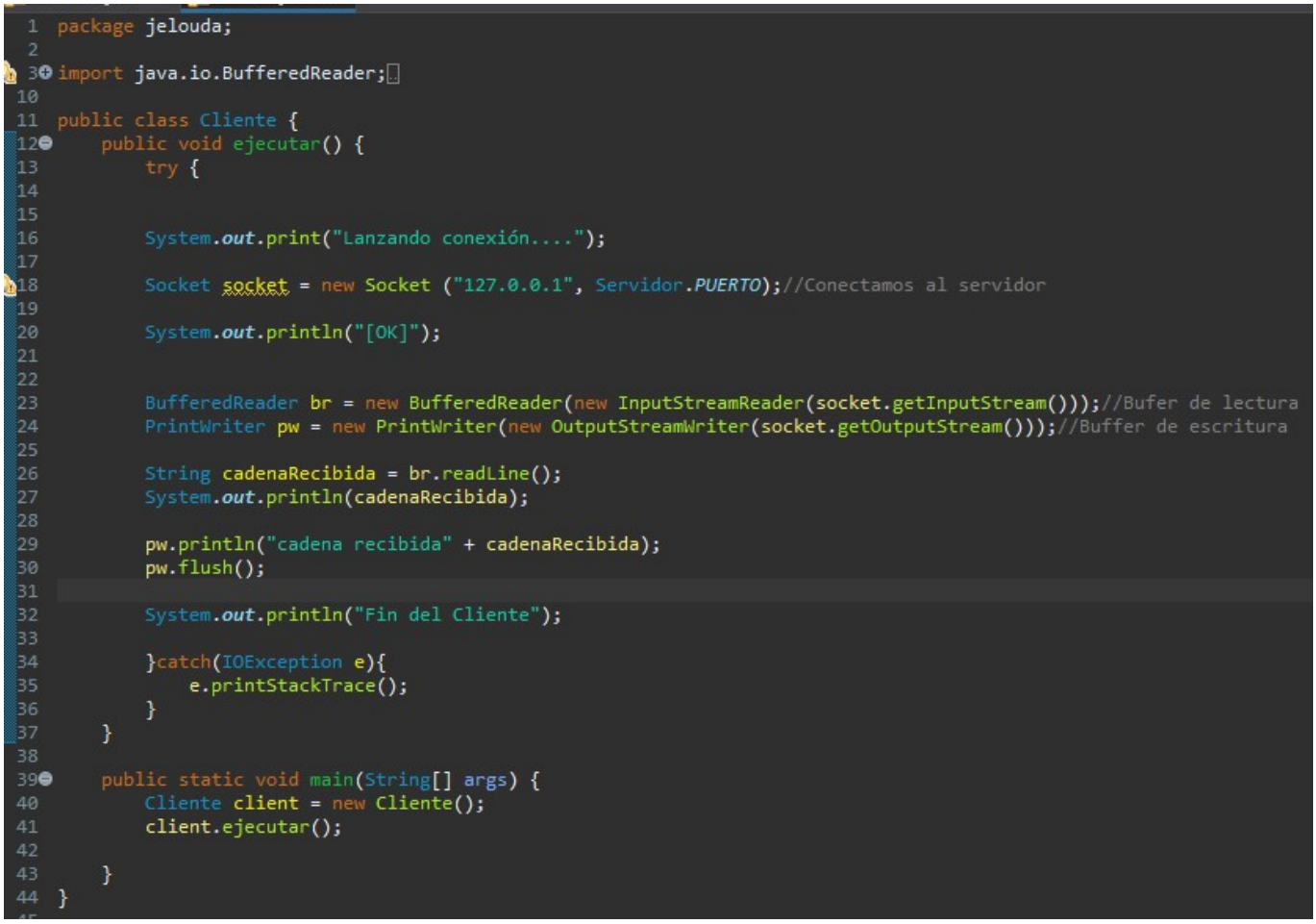
//esperamos una petición
Socket socket = ServerSocket.accept();//Esto para la ejecución y nos devuelve un socket

BufferedReader br = null;
PrintWriter pw = null;

br = new BufferedReader(new InputStreamReader(socket.getInputStream()))//Buffer de lectura
pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()))//Buffer de Escritura

br.readLine();
pw.println("texto a enviar");//Texto a mandar por el Socket
pw.flush();//Similar al Fflush stdin de C OJO
pw.push();
```

Implementación Cliente



```
package jelouda;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Cliente {
    public void ejecutar() {
        try {
        System.out.print("Lanzando conexión....");
        Socket socket = new Socket ("127.0.0.1",
Servidor.PUERTO);//Conectamos al servidor
        System.out.println("[OK]");
        BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));//Bufer de lectura
        PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));//Buffer de escritura
```

```
        String cadenaRecibida = br.readLine();
        System.out.println(cadenaRecibida);
        pw.println("cadena recibida" + cadenaRecibida);
        pw.flush();
        System.out.println("Fin del Cliente");
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        Cliente client = new Cliente();
        client.ejecutar();

    }
}
```

Implementación Servidor

```
package jelouda;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;

public class Servidor {
    public static int PUERTO = 5000;//Indicamos el puerto del servidor
    public void ejecutar() {
        try {

        System.out.println("Lanzando Servidor...");
        ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);

        Socket socket = serverSocket.accept();

        BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));//Buffer entrada
        PrintWriter pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));//Buffer salida

        String cadenaRecibida = br.readLine();

        pw.print(cadenaRecibida);
        pw.flush();
        System.out.println("Fin del Servidor");
        }catch(IOException e){
            e.printStackTrace();
        }
    }

    public static void main(String[] args) {
        Servidor server = new Servidor();
        server.ejecutar();
    }
}

```

```
package jelouda;

import java.io.BufferedReader;
import java.io.IOException;
```

```
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Servidor {
    public static int PUERTO = 5000;//Indicamos el puerto del servidor
    public void ejecutar() {
        try {
        System.out.println("Lanzando Servidor...");
        ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);
        Socket socket = serverSocket.accept();
        BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));//Buffer entrada
        PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));//Buffer salida
        String cadenaRecibida = br.readLine();
        pw.print(cadenaRecibida);
        pw.flush();
        System.out.println("Fin del Servidor");
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        Servidor server = new Servidor();
        server.ejecutar();
    }
}
```

Implementación Servidor continuo

Añadimos un nuevo método al que llamaremos ejecutar infinito:



```
public void ejecutarInfinito() {
    try {


    System.out.println("Lanzando Servidor...");
    ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);
    Socket socket = serverSocket.accept();
    BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));//Buffer entrada
    PrintWriter pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));//Buffer salida

    while(true) {
        System.out.println(br.readLine());
        pw.println(br.readLine());
        pw.flush();
    }


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

```
package jelouda;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Servidor {
    public static int PUERTO = 5000;//Indicamos el puerto del servidor
    public void ejecutar() {
        try {
        System.out.println("Lanzando Servidor...");
        ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);
        //Lo hacemos contínuo
        //Scanner sc = new Scanner(System.in);
        //sc.nextLine();
        Socket socket = serverSocket.accept();
        BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));//Buffer entrada
        PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));//Buffer salida
        String cadenaRecibida = br.readLine();
        pw.print(cadenaRecibida);
        pw.flush();
        System.out.println("Fin del Servidor");
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    public void ejecutarInfinito() {
        try {
        System.out.println("Lanzando Servidor...");
```

```
        ServerSocket serverSocket = new ServerSocket(Servidor.PUERTO);
        Socket socket = serverSocket.accept();
        BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));//Buffer entrada
        PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));//Buffer salida
        while(true) {
            System.out.println(br.readLine());
            pw.println(br.readLine());
            pw.flush();
        }
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        Servidor server = new Servidor();
        server.ejecutar();
    }
}
```

Implementación cliente Continuo

Añadimos un nuevo método al que llamaremos ejecutar infinito:

```
public void ejecutarInfinito() {
    try {


    System.out.print("Lanzando conexión....");

    Socket socket = new Socket ("127.0.0.1", Servidor.PUERTO);//Conectamos al servidor

    System.out.println("[OK]");


    BufferedReader br = new BufferedReader(new InputStreamReader(socket.getInputStream()));//Bufer de lectura
    PrintWriter pw = new PrintWriter(new OutputStreamWriter(socket.getOutputStream()));//Buffer de escritura

    String lineaLeida ="";
    Scanner teclado = new Scanner(System.in);

    while(true) {
        lineaLeida = teclado.nextLine();
        pw.println(lineaLeida);
        pw.flush();
    }


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

```
package jelouda;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
```

```
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.Scanner;

public class Cliente {
    public void ejecutar() {
        try {
        System.out.print("Lanzando conexión....");
        Socket socket = new Socket ("127.0.0.1",
Servidor.PUERTO);//Conectamos al servidor
        System.out.println("[OK]");
        BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));//Bufer de lectura
        PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));//Buffer de escritura
        String cadenaRecibida = br.readLine();
        System.out.println(cadenaRecibida);
        pw.println("cadena recibida" + cadenaRecibida);
        pw.flush();
        System.out.println("Fin del Cliente");
        }catch(IOException e){
            e.printStackTrace();
        }
    }
    public void ejecutarInfinito() {
        try {
        System.out.print("Lanzando conexión....");
        Socket socket = new Socket ("127.0.0.1",
Servidor.PUERTO);//Conectamos al servidor
        System.out.println("[OK]");
        BufferedReader br = new BufferedReader(new
InputStreamReader(socket.getInputStream()));//Bufer de lectura
        PrintWriter pw = new PrintWriter(new
OutputStreamWriter(socket.getOutputStream()));//Buffer de escritura
        String lineaLeida ="";
        Scanner teclado = new Scanner(System.in);
        while(true) {
            lineaLeida = teclado.nextLine();
            pw.println(lineaLeida);
            pw.flush();
        }

        }catch(IOException e){
            e.printStackTrace();
        }
    }
    public static void main(String[] args) {
        Cliente client = new Cliente();
        client.ejecutar();
```

```
    }
}
```

From:
https://knoppia.net/ - **Knoppia**

Permanent link:
https://knoppia.net/doku.php?id=dad:sockets&rev=1696500731

Last update: **2023/10/05 10:12**