Práctica 1: Comunicación con Sockets

SISTEMAS DISTRIBUIDOS

Sara Marcos Cornejo

Práctica 1: Comunicación con Sockets

UNIVERSIDAD NEBRIJA

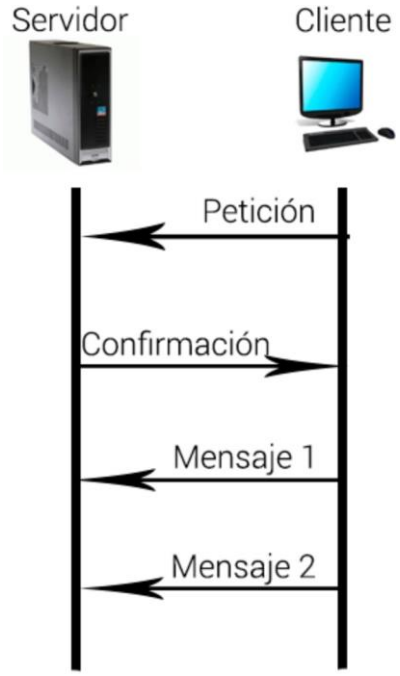
Tabla de contenido

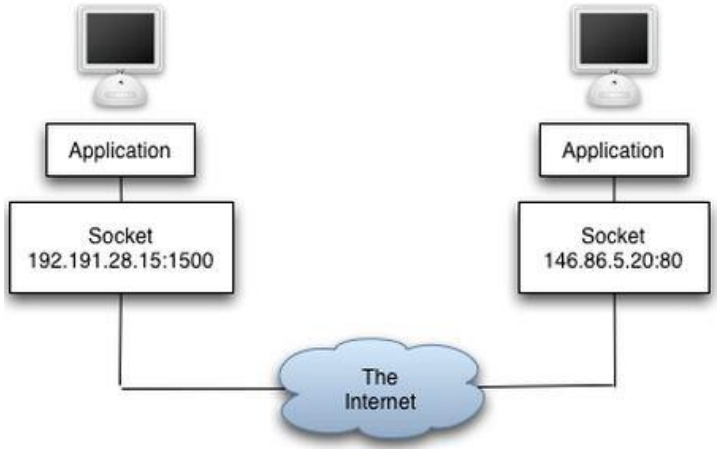
I. Introducción

Los Sockets fueron popularizados por Berckley Software Distribution, de la universidad norteamericana de Berkley. Los sockets han de ser capaces de utilizar el protocolo de streams **TCP** (Transfer Contro Protocol) y el de datagramas **UDP** (User Datagram Protocol).

Utilizan una serie de primitivas para establecer el punto de comunicación, para **conectarse** a una máquina remota en un determinado puerto que esté disponible, para **escuchar** en él, para **leer o escribir** y **publicar información** en él, y finalmente para **desconectarse**.
Con todas primitivas se puede crear un sistema de diálogo muy completo.

Los sockets son un mecanismo que nos permite establecer un enlace entre dos programas que se ejecutan independientes el uno del otro (generalmente un programa cliente y un programa servidor) Java por medio de la librería *java.net* nos provee dos clases: *Socket* para implementar la conexión desde el lado del cliente y *ServerSocket* que nos permitirá manipular la conexión desde el lado del servidor.





Los sockets en Java o cualquier otro lenguaje de programación son capaces de interconectar dos sistemas a través de la red, sólo utilizando un **número IP** o nombre de host y **un puerto** determinado. La arquitectura utilizada en los sockets es la de Cliente/Servidor.

Con el uso de sockets en Java se pueden desarrollar muchos sistemas, como por ejemplos chats, videos juegos online y multijugador o incluso una simple página web.

II. Objetivos

- Declaración de sockets
- Comunicación de Sistemas Distribuidos a través de Sockets

Previo a la implementación de los ejercicios propuestos, es necesario tener instalado unJDK de Java, ya sea a través de una máquina virtual o partición en los equipos personales.

A continuación, se muestra un ejemplo de implementación de una comunicación entre uncliente servidor a través de sockets.

1. Clase Servidor

```
1  import java.io.DataInputStream;
2  import java.io.DataOutputStream;
3  import java.io.IOException;
4  import java.net.ServerSocket;
5  import java.net.Socket;
6  import java.util.logging.Level;
7  import java.util.logging.Logger;
8
9  public class Servidor {
10     public static void main(String[] args) {
11         ServerSocket servidor = null;
12         Socket sc = null;
13         DataInputStream in;
14         DataOutputStream out;
15         final int PUERTO = 5001;
16
17         try {
18             servidor = new ServerSocket (PUERTO);
19             System.out.println("Servidor iniciado");
20
21             while(true) {
22                 sc = servidor.accept();
23                 System.out.println("Cliente conectado");
24
25                 in = new DataInputStream(sc.getInputStream());
26                 out = new DataOutputStream(sc.getOutputStream());
27
28                 String mensaje = in.readUTF();
29                 System.out.println(mensaje);
30                 out.writeUTF("Hola mundo desde el cliente");
31                 sc.close();
32                 System.out.println("Cliente desconectado");
33             }
34         }catch(IOException ex) {
35             Logger.getLogger(Servidor.class.getCanonicalName()).log(Level.SEVERE, null, ex);
36         }
37     }
38 }
39
```

2. Clase Cliente

```java
import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Cliente {

    public static void main(String[] args) {
        final String HOST="127.0.0.1";
        final int PUERTO = 5001;
        DataInputStream in;
        DataOutputStream out;

        try {
            Socket sc = new Socket(HOST, PUERTO);
            in = new DataInputStream(sc.getInputStream());
            out = new DataOutputStream(sc.getOutputStream());

            out.writeUTF("Hola mundo desde el cliente");
            String mensaje = in.readUTF();
            System.out.println(mensaje);
            sc.close();

        }catch(IOException ex) {
            Logger.getLogger(Servidor.class.getCanonicalName()).log(Level.SEVERE, null, ex);
        }
    }
}
```

III. Entregables

A. Ejercicio 1

Implementar un programa que sea capaz de calcular el área de un círculo. Para ello, será necesario crear la clase Circulo e implementar el método dentro de esta clase. Los parámetros se pasarán por referencia. Justificar el proceso 4 PUNTOS.

VisualStudio > practica1sd > Circulo.java > ...

```java
package practica1sd;

public class Circulo {
private static double radio = 5;
private double area;

public Circulo(double r) {
    this.area = Math.PI * Math.pow(r, b: 2);
}
public double getArea() {
    return area;
}
public void showArea() {
    System.out.println("Radio:" + radio +", Area: " + getArea());
}

Run | Debug
public static void main(String[] args) {
    Circulo c = new Circulo(radio);
    c.showArea();
    }
}
```

Para este primer ejercicio se implementa la clase Circulo. Se pide calcular su área, por lo que el único parámetro necesario para ello será el radio del círculo.
Se implementa la función que calcula el área del objeto Circulo en función de su radio, siendo la fórmula: **$A = \pi r^2$**

En la función main() se crea un nuevo objeto Circulo llamado c, con radio establecido previamente en la clase. Llamando a la función showArea(), se muestra por pantalla el radio del círculo y su correspondiente área.

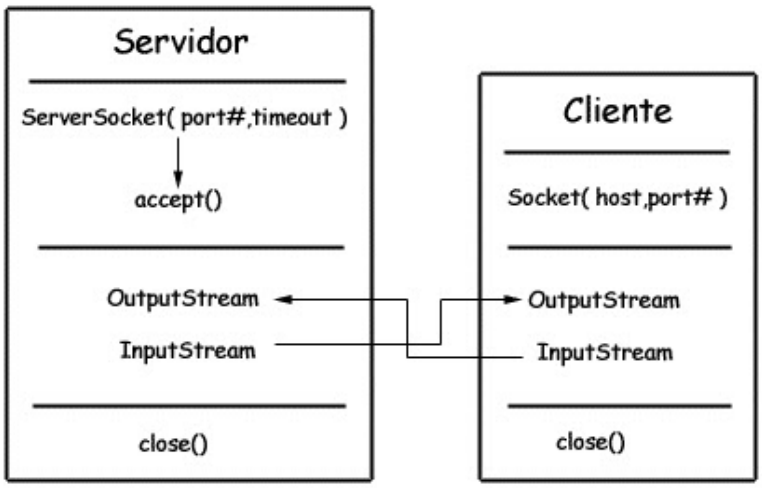
La salida por consola sería la siguiente:

```
PS C:\Users\saram\Documents\VisualStudio>  c:; cd 'c:\Users\saram\Documents\VisualStudio'; & 'C:\Program Files\
lsInExceptionMessages' '-cp' 'C:\Users\saram\AppData\Roaming\Code\User\workspaceStorage\f183b30151b9d3f7b4a3bdc
b\bin' 'practica1sd.Circulo'
Radio:5.0, Area: 78.53981633974483
```

B. Ejercicio 2

Implementar la clase Circulo desarrollada en el ejercicio 1 en un servidor y realizar la petición de cálculo del área desde el cliente. La comunicación se realizará a través de la implementación de sockets. Justificar el proceso 6 PUNTOS.

En este ejercicio se realiza el cálculo del área de un círculo al igual que en el primer ejercicio, pero esta vez implementado a través de sockets mediante un cliente y un servidor, de esta forma:



Class Cliente

VisualStudio > practica1sd > Cliente.java > Cliente > main(String[])

```java

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;

public class Cliente {
    Run | Debug
    public static void main(String[] args){
        final String HOST = "127.0.0.1";
        final int PUERTO = 5001;
        DataInputStream in; //Flujo de entrada del servidor al cliente
        DataOutputStream out; //Flujo de salida del cliente al servidor
        try {
            Socket sc = new Socket(HOST, PUERTO);
            in = new DataInputStream(sc.getInputStream());
            out = new DataOutputStream(sc.getOutputStream());

            out.writeUTF(str: "Hola mundo desde el cliente"); //Envia el mensaje al servidor
            out.writeInt(v: 3); //Envia el mensaje al servidor

            //desde el cliente se realiza la peticion al servidor y se imprime el mensaje que se recibe
            String mensaje = in.readUTF();
            System.out.println(mensaje);
            sc.close();

        } catch (Exception e) {
            Logger.getLogger(Servidor.class.getCanonicalName()).log(Level.SEVERE, msg: null, e);
        }
    }
}
```

Class Servidor

VisualStudio > practica1sd > Servidor.java > Servidor > main(String[])

```java
public class Servidor {
    Run | Debug
    public static void main(String[] args){
    ServerSocket servidor = null;
    Socket sc = null;
    DataInputStream in;//Flujo de entrada del cliente al servidor
    DataOutputStream out; //Flujo de salida del servidor al cliente
    final int PUERTO = 5001;

    try {
        servidor = new ServerSocket(PUERTO);
        System.out.println(x: "Servidor iniciado");
        while(true) {
            sc = servidor.accept();
            System.out.println(x: "Cliente conectado");

            in = new DataInputStream(sc.getInputStream());
            out = new DataOutputStream(sc.getOutputStream());

            String mensaje = in.readUTF(); //Lee el mensaje del cliente y lo guarda en la variable mensaje
            System.out.println( mensaje);
            out.writeUTF(str: "Hola mundo desde el servidor");

            //desde el servidor se llama al metodo que realiza la operacion y se envia el resultado al cliente
            int num = in.readInt();
            Circulo c = new Circulo(num);
            out.writeUTF("El area del circulo es: " + c.getArea());
            c.showArea();

            sc.close();
            System.out.println(x: "Cliente desconectado");
        }
    } catch (Exception e) {
        Logger.getLogger(Servidor.class.getCanonicalName()).log(Level.SEVERE, msg: null, e);
        }
    }
}
```

En la clase Cliente se establece la **dirección IP**, el **puerto** a través del cual se van a comunicar (siendo el mismo para cliente y para servidor), un **DataInputStream** (referenciando al flujo de entrada del cliente al servidor), un **DataOutputStream** (referenciando al flujo de salida del servidor al cliente y por último, el **socket** que permitirá la comunicación entre ambas clases, tomando como parámetros la dirección IP y puerto establecidos.

En la clase Servidor se crea el **servidor**, el **socket** que permitirá la comunicación entre ambas clases, un **DataInputStream** (referenciando al flujo de entrada del cliente al servidor), un **DataOutputStream** (referenciando al flujo de salida del servidor al cliente) y por último el **puerto** a través del cual se van a comunicar (siendo el mismo para cliente y para servidor).

Se inicializan los flujos de entrada y salida en ambas clases.

En el cliente, se manda inicialmente un mensaje al servidor para comprobar que funciona correctamente, y seguidamente se manda al servidor el dato para que este calcule el área del círculo a través de out.write().
El servidor lee el dato que ha mandado el cliente a través de in.read(), y lo almacena en una variable. Crea el objeto Circulo c, pasándole como parámetro el radio que le mandó el cliente. A continuación, calcula el área de ese círculo y se lo manda al cliente de nuevo. El servidor ya ha terminado su función, por lo que se debe cerrar el socket.
Por último, el cliente está a la escucha del dato que el servidor envía como resultado; lo lee y lo muestra por pantalla. Finalmente, cierra también el socket creado.

La salida por consola sería la siguiente:

```
Cliente conectado
Hola mundo desde el cliente
Area: 28.274333882308138
Cliente desconectado
```

Nota:
Es importante a la hora de probar estos dos códigos, empezar ejecutando el servidor y luego el cliente ya que si se ejecuta el cliente en primer lugar fallará la conexión porque no podrá conectarse al servidor.

IV. Conclusiones

En esta práctica se ha podido profundizar acerca de los sockets, ampliando el conocimiento del temario visto en las clases de teoría.

En programación los sockets tienen mucha importancia debido a su uso extendido. A través de ellos y en conjunto con un patrón de diseño podemos implementar un servicio web robusto y extensible, para que el cliente logre obtener lo que desee, llegando incluso a mejores resultados facilitando la comunicación entre equipos y muchas otras cosas más.

V. Bibliografía

https://unipython.com/sockets-en-java-cliente-y-servidor/

https://www.ibm.com/docs/es/i/7.1?topic=communication-example-using-sockets interprocess

VI. Criterios de evaluación

A. Calificación mínima

La ponderación de las prácticas, solo se aplicará si el alumno obtiene al menos un 5 en la media en las cinco prácticas. Esta ponderación también se aplica solo en el caso de que el alumno obtenga al menos un 5 en este examen final extraordinario. Si se obtiene más de un 7 en la media de las prácticas y se suspende el examen extraordinario, se guardarán las prácticas únicamente para el año siguiente.

B. Asistencia

La asistencia a las prácticas es obligatoria, en caso de falta justificada la recuperación de esta se realizará en una fecha propuesta por el profesor. En el caso de no estar justificada la asistencia del alumno, obtendrá un 0 en dicha práctica.

C. Advertencia sobre plagio

La Universidad Antonio de Nebrija no tolerará en ningún caso el plagio o copia. Se considerará plagio la reproducción de párrafos a partir de textos de auditoría distinta a la del estudiante (Internet, libros, artículos, trabajos de compañeros…), cuando no se cite la fuente original de la que provienen. El uso de las citas no puede ser indiscriminado. El plagio es un delito.

En caso de detectarse este tipo de prácticas, se considerará Falta Grave y se podrá aplicarla sanción prevista en el Reglamento del Alumno.

D. Ponderación

En el caso de que haya ejercicios correctamente desarrollados y no se

justifique su implementación se calificará sobre la mitad de la puntuación de ese ejercicio.

Si la solución del ejercicio no es correcta por pequeños detalles de implementación y su justificación es correcta. La calificación se aplicará según criterio del profesor.

Si el ejercicio solo cuenta con una justificación teórica y no con una implementación práctica, el ejercicio obtendrá 0 puntos.

E. Entregas Ordinarias

Las prácticas serán entregadas al final de la clase. Los alumnos deberán subir la prácticaen formato ZIP al campus virtual. El fichero ZIP deberá contener:

- Memoria de la práctica deberá estar en formato PDF (índice, enunciados,resultados, conclusiones, bibliografía...)
- Códigos de cada ejercicio

***No se admitirán prácticas fuera de fecha y horario establecido.**

F. Entregas Extraordinarias

Las prácticas serán entregadas en el período de recuperación (Fecha por determinar). Los alumnos deberán subir la práctica en formato ZIP al campus virtual. El fichero ZIP deberá contener:

- Memoria de la práctica deberá estar en formato PDF (índice, enunciados, resultados, conclusiones, bibliografía...)
- Códigos de cada ejercicio

***No se admitirán prácticas fuera de fecha y horario establecido.**