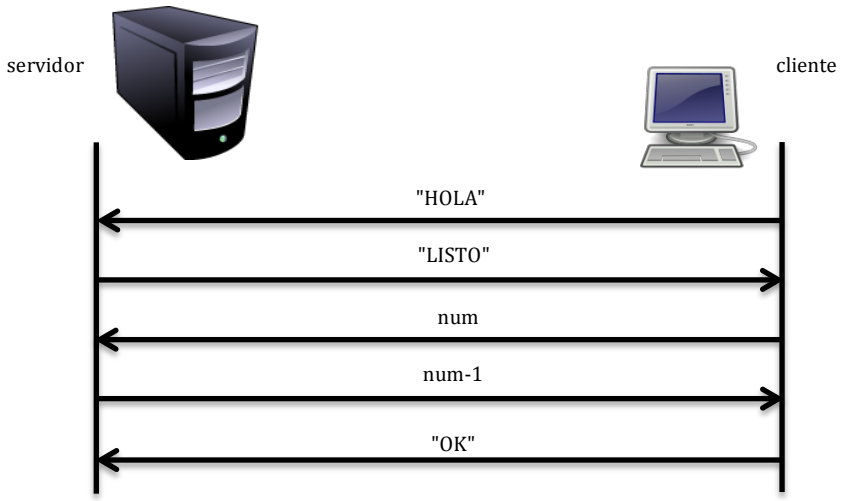
Universidad de los Andes
Ingeniería de Sistemas y Computación
ISIS 2203 Infraestructura Computacional

Taller - Servidores Concurrentes

El propósito de este taller es construir un servidor concurrente que responde a múltiples pedidos de varios clientes. El servidor recibe los pedidos y envía las respuestas por medio de sockets, uno de los medios tradicionales de comunicación entre aplicaciones en Internet.

Para el taller tendremos un servidor principal que se encarga de recibir las solicitudes de conexión de los clientes y crear un thread servidor por cada solicitud. Cada thread termina cuando el cliente termina la conexión. La siguiente figura ilustra el protocolo de comunicación entre el servidor y un cliente.



Cree un proyecto en Eclipse para el servidor y uno para el cliente. Al crear los proyectos en Eclipse verifique que el JRE sea el esperado (y no una versión reducida).

Servidor:

Escriba el servidor. Usaremos un esquema de servidor construido sobre dos clases: (1) el servidor principal, coordina la recepción de pedidos sobre un socket y la creación de threads para atender las solicitudes correspondientes, (2) el servidor delegado se ejecuta en un thread y atiende un pedido, debe además implementar el protocolo de comunicación con el cliente.

Abajo se incluye parte del código del servidor (para las dos clases). Usted debe entender el código y completarlo. Para ello, revise los comentarios y todo lo que aparece entre los signos <>. Los comentarios y los signos indican las instrucciones y variables a completar y remplazar.

1. En la clase del servidor principal:

a. Defina una constante para identificar el puerto que usará para recibir los pedidos de los clientes (entero) (identificado abajo por <PUERTO>)
b. ¿Por qué es apropiado definir este valor como una consntante?
c. Escriba el método main de la clase. A continuación encuentra parte del código. Observe que debe manejar un identificador de thread y la constante que identifica el puerto.

```
public static void main(String[] args) throws IOException {
        ServerSocket ss = null;
        boolean continuar = true;
        // defina variable para contar e identificar los threads

        try {
            ss = new ServerSocket(<PUERTO>);
        } catch (IOException e) {
            System.err.println("No pudo crear socket en el puerto:" + <PUERTO>);
            System.exit(-1);
        }

        while (continuar) {
            new ThreadServidor(ss.accept(),<identificador de thread>).start();
            // incremente identificador de thread
        }

        ss.close();
}
```

2. Escriba la clase ThreadServidor. Observe que debe definir los atributos de la clase y asignarlos en los métodos constructor y run.

```
public class ThreadServidor extends Thread{
          // atributo socket
          private Socket sktCliente = null;
          // defina un atributo identificador local de tipo int

          public ThreadServidor(Socket pSocket,int pId) {
            // asigne el valor a los atributos correspondientes
          }

          public void run() {

            System.out.println("Inicio de nuevo thread." + <id local>);

            try {
                  PrintWriter escritor = new
PrintWriter(sktCliente.getOutputStream(), true);
                  BufferedReader lector = new BufferedReader(new
InputStreamReader(sktCliente.getInputStream()));
                  procesar(lector,escritor);
                  escritor.close();
                  lector.close();
                  sktCliente.close();

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

```
            }
        }
}
```

3. Escriba la clase Protocolo. Primero construya un esquema gráfico del protocolo que quiere implementar. Piense en un protocolo simple (como el ejemplo en la presentación o el ejemplo que se incluye a continuación).

A continuación encuentra el ejemplo del método procesar. Usted lo puede modificar como considere conveniente.

```
public void procesar(BufferedReader pIn,PrintWriter pOut) throws IOException {
	String inputLine, outputLine;
	int estado = 0;

	while (estado < 3 && (inputLine = pIn.readLine()) != null) {
		switch (estado) {
			case 0:
				if (inputLine.equalsIgnoreCase("HOLA")) {
					outputLine = "LISTO";
					estado++;
				} else {
					outputLine = "ERROR-EsperabaHola";
					estado = 0;
				}
				break;
			case 1:
				try {
					int val = Integer.parseInt(inputLine);
					val++;
					outputLine = "" + val;
					estado++;
				} catch (Exception e) {
					outputLine = "ERROR-EnArgumentoEsperado";
					estado = 0;
				}
				break;
			case 2:
				if (inputLine.equalsIgnoreCase("OK")) {
					outputLine = "ADIOS";
					estado++;
				} else {
					outputLine = "ERROR-EsperabaOK";
					estado = 0;
				}
				break;
			default:
				outputLine = "ERROR";
				estado = 0;
				break;
		}
	}
}
```

Cliente:

4. Escriba el cliente.
    a. Defina una constante con la dirección IP de la máquina en la que está trabajando.
    b. A continuación encuentra parte del código del método main. Observe que debe definir el socket y manejar la dirección y el puerto del servidor correspondiente.

```
boolean ejecutar = true;
Socket <el socket> = null;
PrintWriter escritor = null;
BufferedReader lector = null;

try {
    <el socket> = new Socket(<el host>, <el puerto>);
    escritor = new PrintWriter(<el socket>.getOutputStream(), true);
    lector = new BufferedReader(new InputStreamReader(
                                    <el socket>.getInputStream()));
} catch (Exception e) {
    System.err.println("Exception: " + e.getMessage());
    System.exit(1);
}

BufferedReader stdIn = new BufferedReader(
                                new InputStreamReader(System.in));
String fromServer;
String fromUser;

while (ejecutar) {

    System.out.print("Escriba el mensaje para enviar:");
    fromUser = stdIn.readLine();
    if (fromUser != null && !fromUser.equals("-1")) {
          System.out.println("Cliente: " + fromUser);
          if (fromUser.equalsIgnoreCase("OK"))
                ejecutar = false;
          escritor.println(fromUser);
    }
    if ((fromServer = lector.readLine()) != null) {
         System.out.println("Servidor: " + fromServer);
    }
}
escritor.close();
lector.close();
// cierre el socket y la entrada estándar
```

Ejecute:

6. Primero ejecute el servidor y a continuación varios clientes de forma concurrente.