Los dos ejercicios que se proponen a continuación van a a implementar una aplicación cliente / servidor a través de sockets TCP. En ambos casos el servidor solo atenderá a un cliente.

Ejercicios Socket TCP

1. La aplicación implementará el servicio "SALUDO CON FECHA".
Tanto servidor como cliente se ejecutan inicialmente en "localhost".

Parte del servidor

```
Servidor conectado en puerto local 5000
Esperando una conexión de cliente

Conexión aceptada desde IP 192.168.0.194
```

a) Escucha peticiones en el puerto 5000
b) Acepta una conexión
c) Atiende a un cliente
   - Lee el nombre que el cliente le ha enviado
   - Saludo al cliente de la forma *"Hola Elena hoy es 3 – Enero – 2014"*
d) Cierra la conexión

Importa el proyecto ServidorSaludoFechaAL y complétalo.

- añade los atributos necesarios (socket y flujos)
  - para lectura flujo Scanner, para escritura flujo PrintWriter
- completa el método iniciarServidor(). Tendrás que incluir el código para iniciar el servidor, aceptar conexión del cliente, …..
  - muestra mensajes adecuados antes y después de aceptar la conexión del cliente
  - captura excepciones
  - cierra las conexiones y flujos
- completa obtenerFechaActual() que calcula y devuelve la fecha de hoy como un String de la forma dia / nombremes /año
  - puedes consultar la API o Internet para trabajar con la clase Calendar (http://lineadecodigo.com/java/obtener-fecha-actual-con-java/)

Parte del cliente

```
Error, Sintaxis: java <ip servidor> <puerto servidor>
```

```
Cliente conectado a servidor de IP 192.168.0.194
Recibido del servidor Hola Elena Hoy es 3 - Enero - 2014
```

a) se conecta al servidor
b) le envía su nombre
c) recibe del servidor el saludo con la fecha actual
d) muestra la fecha en pantalla
e) cierra la conexión

Importa el proyecto ClienteSaludoFechaAL y complétalo.

- la IP (o nombre de host) y el puerto se toman como argumentos del main(). La IP será un String y el puerto un valor entero. Si el nº de argumentos es incorrecto se muestra un mensaje de error y se acaba el programa. Si es correcto se aceptan los argumentos
- para escritura el flujo será PrintWriter, para lectura el flujo será Scanner
- captura todas las excepciones mostrando el error en cada caso (incluso en caso de conversión de puerto incorrecto)

➔ Ejecuta el programa probando la parte servidor y cliente en la máquina local.
➔ Desde línea de comandos haz una ejecución del programa.
➔ Con uno de tus compañero/a prueba la aplicación (tendrás que hacer previamente los cambios adecuados).
➔ Por último, haz una ejecución incluyendo la parte cliente en la VM (VirtualBox).

2. En este ejercicio vas a implementar un servidor TCP que recibe números hasta el 0 y envía al cliente la suma de todos ellos, el máximo y la media.

El diálogo entre el servidor y el cliente es el siguiente:

Servidor	Cliente
• acepta conexión del cliente • envía una vez aceptada la conexión un mensaje de bienvenida al cliente • lee uno a uno los números que le envía el cliente haciendo los cálculos requeridos • envía la suma de los números • envía el máximo • envía la media	• conecta con servidor • recoge y muestra el mensaje de bienvenida • envía al servidor uno a uno una serie de número (que lee del teclado) hasta el cero • recoge y muestra la suma • recoge y muestra el máximo • recoge y muestra la media formateada a dos decimales

Parte del servidor -

Crea un proyecto Eclipse de nombre SumadorNumeros:

- incluye un paquete pkgservidor.sumador que incluye la clase SumadorNumeros
- la clase únicamente tiene el main()
  - incluye aquí todo el código necesario para implementar el servidor.
- el puerto en el que escuchará el servidor será un argumento del main()
  - comprueba que se pasa el parámetro, si no es así se le muestra al usuario la sintaxis correcta de la llamada al programa.
- los flujos a utilizar serán DataOutputStream y DataInputStream
- captura todas las excepciones que se puedan producir (incluso error de conversión de puerto) mostrando mensajes adecuados en cada caso

```
Error, Sintaxis: java SumadorNumeros <puerto>
```

```
Error al convertir puerto - java.lang.NumberFormatException: For input string: "800p"
```

```
Servicio sumador de números, cálculo del máximo y la media
Servidor esperando conexiones en puerto 8000

Aceptada conexión del cliente desde 127.0.0.1

Recibido desde cliente 6
Recibido desde cliente 8
Recibido desde cliente 9
Recibido desde cliente 4
Recibido desde cliente 3
Recibido desde cliente 0
Enviando suma, máximo y media .......
Todo enviado.......
```

Parte del cliente -

Crea un proyecto Eclipse de nombre ClienteSumador:

- crea un paquete pkgcliente.sumador con la clase ClienteSumador
- la clase contiene únicamente el método main()
  - incluye aquí todo el código necesario para implementar el cliente
- el nombre del host y puerto del servidor se aceptarán como argumentos. Controla los posibles errores y muestra al usuario mensajes acerca de cómo invocar al programa
- los flujos a utilizar serán DataInputStream y DataOutputStream
- captura todas las posibles excepciones

```
Error, Sintaxis: java ClienteSumador <host> <puerto>
```

```
Conectado al servidor 127.0.0.1
Desde servidor - Bienvenido/a al sumador de números
Espero tu envío de números hasta el 0

Dame número para enviar al servidor, 0 terminar
6
Dame número para enviar al servidor
8
Dame número para enviar al servidor
9
Dame número para enviar al servidor
4
Dame número para enviar al servidor
3
Dame número para enviar al servidor
0
Desde servidor - La suma es 30
Desde servidor - El máximo es 9
Desde servidor - La media es   6,00
```

3. Abre el proyecto Escaneador de puertos TCP AL y complétalo.
Vas a implementar un sencillo analizador de puertos en Java que permite ver qué puertos TCP están abiertos o cerrados en un determinado host cuya ip o nombre y rango de puertos a analizar se tomarán como argumentos del main().

```
Conexión no establecida en puerto 75 - Puerto cerrado
Conexión no establecida en puerto 76 - Puerto cerrado
Conexión no establecida en puerto 77 - Puerto cerrado
Conexión no establecida en puerto 78 - Puerto cerrado
Conexión no establecida en puerto 79 - Puerto cerrado
Conexión no establecida en puerto 80 - Puerto cerrado
Conexión no establecida en puerto 81 - Puerto cerrado
Conexión no establecida en puerto 82 - Puerto cerrado
Conexión no establecida en puerto 83 - Puerto cerrado
Conexión no establecida en puerto 84 - Puerto cerrado
Conexión no establecida en puerto 85 - Puerto cerrado
```

El analizador intentará crear un socket al host y puerto indicados, si se puede crear el socket es que se ha aceptado la conexión, por tanto, en el host había un servidor escuchando y significará que el puerto estaba abierto.

Muestra en pantalla los mensajes adecuados que indique, *"Puerto abierto"* o *"Puerto cerrado*".

Haz una ejecución del programa desde línea de comandos.