Práctica 5: Pasos

Paso 0: Información

- Para resolver esta práctica necesitará haber leído tanto el enunciado de la práctica como los apuntes de sockets que se facilitaron en la práctica anterior (parte de sockets UDP)
- Realice una primera lectura completa de la práctica y luego conforme vaya haciendo cada uno de los pasos repase la parte relacionada. Note que a veces la información para realizar un paso está dividida en varias partes como por ejemplo para hacer el método **`crearSocket`** tiene información tanto en la primera como en la segunda y la tercera página

Paso 1: Métodos iniciales

- Descargue el código facilitado en el campus virtual
- Importe el proyecto en eclipse o su entorno favorito (para hacer las pruebas los equipos deben estar en la misma red por lo que se recomiendo usar un equipo del laboratorio)
- Sólo necesita modificar la clase **`ComunicacionImpl`** del paquete **`es.uma.informatica.rsd.chat.impl`**
- Implemente una primera versión simple de **`crearSocket`** en la que creará un socket UDP para aceptar mensajes desde el puerto recibido. También debe almacenar el alias para un uso posterior
- Implemente el método **`setControlador`**

Paso 2: Crear mensaje a enviar (unicast)

- En el método **`envía`**, cree un String con el formato del mensaje a enviar e imprímalo por la pantalla
- Ejecute el programa (**`ControladorImpl`**) y pruebe que el mensaje generado en la consola es correcto (esto no envía, solo es para probar que se está generando el formato de forma adecuada)

Paso 3: Envío (unicast)

- Modifica el método **`envía`**
- Cree la dirección destino del mensaje (**`InetAddress`**)
- Cree los datos (array de bytes) a enviar
- Cree el **`DatagramPacket`** a ser enviado con la información generada previamente
- Envíe el **`DatagramPacket`** usando el **`DatagramSocket`** creado en **`crearSocket`**
- Pruebe el funcionamiento intentando enviar mensajes a la máquina donde el profesor puso el programa de prueba

Paso 4: Recepción (unicast)

- En el método **`runReceptor`**, cree un **`DatagramPacket`** de recepción (es decir hay que reservar la memoria donde se almacenarán los datos recibidos)
- Reciba el mensaje usando el **`DatagramSocket`**
- Separe el mensaje recibido de forma apropiada (las tres partes: IP, alias, mensaje)
- Cree la dirección socket del remitente (**`InetSocketAddress`**)
- Escriba en el interfaz gráfico el mensaje recibido (usando el controlador)
- Este método NO debe finalizar, modifíquelo para conseguir que nunca acabe
- Pruébelo enviando y recibiendo mensajes de un compañero o del equipo del profesor

Paso 4: Configuración con multicast

- Cambie la clase utilizada para manejar el socket UDP de **`DatagramSocket`** a **`MulticastSocket`**
- Implemente los métodos **`joinGroup`** y **`leaveGroup`**
- Para su buen funcionamiento, se debe utilizar la variante del constructor/métodos que nos permita establecer cuál es el interfaz elegido para el envío y recepción de tráfico (debe usar el real)

Paso 5: Envío multicast

- En el método **`envía`** detecte si la dirección destino es multicast o no y modifique el String del mensaje a enviar de forma apropiada
- Para hacer pruebas multicast a partir de este momento póngase de acuerdo con sus compañeros para elegir el puerto a utilizar o use el facilitado por el profesor. La dirección multicast será **239.194.17.132**
- Pruebe el envío multicast

Paso 6: Recepción multicast: interfaz gráfica

- Observando el mensaje recibido detecte si era multicast o no y en caso de ser multicast modifique dirección socket del remitente (**`InetSocketAddress`**)
- Si lo ha hecho correctamente todos las recepciones multicast debe recibirlas en una única pestaña que tendrá como nombre la dirección multicast

Paso 7: Recepción multicast: mensajes propios

- Observando el remitente del **`DatagramPacket`** recibido, compruebe si el emisor era tu propio programa. En ese caso, descarte el mensaje sin mostrarlo en el interfaz gráfico
- Compruebe que no muestra sus propios mensajes

Paso 8: Ajustes finales

- ¿Funciona bien si el datagrama tiene un formato inválido?
- ¿Funciona si se envía el > como parte del mensaje?
- Vaya mostrando por la consola información de cómo va funcionando (envíos realizados, recepciones, datagramas descartados, …)

Paso 9: Ejercicios

- Realice los ejercicios, tomando las capturas de pantalla y trazas de Wireshark indicadas

Paso 10: Entrega

- Recuerde que además de completar los ejercicios debe explicar los elementos más importantes del código (igual en la práctica anterior)
- Recuerde que además del documento con la memoria debe adjuntar la traza de Wireshark y el código desarrollado (sólo el fichero **ComunicacionImp.java**)
- Si lo hizo en grupo, la memoria y ficheros son comunes para los dos y en la página inicial de la memoria debe aparecer el nombre de ambos
- Al final del enunciado de la práctica 5 tiene los detalles del formato y ficheros que debe entregar en este bloque (práctica 4 y 5)