Laboratorio 2

Comunicación entre procesos: Sockets e Hilos

Objetivo:

- Dominar el concepto de Sockets para el desarrollo de aplicaciones sencillas en el esquema Cliente-Servidor usando Eclipse
- Desarrollar un ejemplo práctico e una aplicación en dos niveles implementando un sencillo servidor de aplicaciones y un cliente
- Usar enchufes Multihilos
- Analizar el código suministrado y realizar las modificaciones correspondientes para el funcionamiento correcto de los ejemplos.

Parte I – Comunicación por sockets y con hilos.

a. Hilos

i. Servidor de imágenes

Descargar el archivo **servidorImagenes.zip**

(https://www.dropbox.com/s/snhw2dhhv5wi7rn/ServidorImagenes.zip) que contiene un programa que permite que un Servidor que cuando un cliente establezca comunicación con este, muestre una relación de imágenes en una lista transferida por el servidor, en la cual al escogerse una de ellas y presionar el botón **recibir** se visualiza en el cliente la imagen escogida. Las imágenes y los nombres inicialmente deben estar en el servidor, NO SON PREVIAMENTE CONOCIDAS EN DICHO CLIENTE. Modificar el programa para:

1. Al momento de conectarse el cliente al servidor, este únicamente debe recibir los nombres de las imágenes que están en dicho servidor.
2. Cuando se escoja un nombre de la lista recibida recién será enviada dicha imagen al cliente para su visualización.
3. El puerto de acceso a la lista de imágenes del servidor será 5998
4. El puerto de acceso a recibir la imagen enviada por el servidor, en base al nombre enviado por el cliente será 2889

ii. Servidor con múltiples clientes.

Diseñar una aplicación cliente/servidor usando socket TCP y ventanas JDialog, para permitir el acceso simultaneo de clientes, cada cliente debe identificarse con un alias (**nickname**), probando el cliente con diferentes mensajes, dichos mensajes deben ser vistos en el servidor y en todos los clientes conectados. Además deberá haber en el servidor un JList con el nombre de los clientes que están conectados y que por cada cliente que se desconecta debe eliminarse su nombre de la lista y aparecer el mensaje en el JEditPane del servidor y de los clientes "El cliente xxxxx se ha desconectado a las yy:yy:yy", ademas el servidor debe tener la capacidad de desconectar clientes si esto es necesario enviando la información respectiva a todos.

Nota importante: Debe permitirse el agregar **emoticons** al mensaje enviado por cualquiera de los clientes, para ello en cada pantalla de cliente debe haber una lista de estas figuras para agregar.

Nota importante: Los emoticons **deben enviarse** al servidor para que este los pueda visualizar y reenviar a los clientes conectados. No debe haber imágenes cargadas por defecto en dicho servidor, ya que esto invalidara el laboratorio.

Para el acceso de múltiples clientes debe usarse hilos.

Nota Importante: Los emoticones **(emoticons.rar)** se encuentran en https://www.dropbox.com/s/6u9vvvys2vvavj0/emoticons.rar y deben descargarse de ella para realizar el laboratorio.

b. Websockets

Diseñar una aplicación de tipo chat usando websocket en entorno java para permitir el acceso simultáneo de clientes, cada cliente debe identificarse con un alias (**nickname**), probando el cliente con diferentes mensajes, dichos mensajes deben ser vistos en el servidor y en todos los clientes conectados.

Además deberá haber en el servidor una lista con el nombre de los clientes que están conectados y que por cada cliente que se desconecta debe eliminarse su nombre de la lista y aparecer el mensaje en la pantalla del servidor y de los clientes "El cliente xxxxx se ha desconectado", además el servidor debe tener la capacidad de desconectar clientes si esto es necesario enviando la información respectiva a todos

Utilizar Eclipse Kepler, con el servidor web java de su elección.