

Programación orIENTADA A oBJETOS

Práctica 5: Sockets Clientes



Alumno: Vargas Romero Erick Efraín

Profesor: Tecla Parra Roberto

Fecha: 16-05-2017

Grupo: 2CM4

**Introducción**

Los sockets son un mecanismo que nos permite establecer un enlace entre dos programas que se ejecutan independientes el uno del otro, esto generalmente se hace un programa cliente y otro servidor, pero para este caso solo nos centraremos en los sockets clientes.

**Desarrollo**

Primeramente, hay que recalcar que se ha modificado el programa ubicado en la carpeta estatiProgBarSimBot\_/hombregatoact2014.

Una vez mencionado lo anterior se ha modificado el archivo Tamagochi la cual contiene a una clase llamada Tamagochi la cual extiende de la clase JFrame e implementa la interfaz LeeRed.

public class Tamagochi **extends** JFrame **implements** LeeRed **{**

Dónde la interfaz LeeRed contiene al método void leeRed el cual tiene un solo parámetro de tipo Object.

public interface LeeRed **{**

void leeRed**(**Object obj**);**

**}**

Lo siguiente es declarar nuestras variables de instancia

private Canvas3D canvas3D**;**

private Appearance ap**;**

private static Texture texture**;**

private JPanel jp1**,** jp2**;**

private JButton bcambia**,** BTNFeliz**,** BTNEnfermo**,** BTNAA2Mareado**,** BTNA4Sospechoso**,** BTNA1Feliz**;**

private EventHandler eh**;**

private String nombres**[]={**"carafeliz.jpg"**,**"caraenfermo.jpg"**,** "babeando.jpg"**,** "Avatar 4.jpg"**,** "Avatar1.jpg"**};**

private int turno**;**

private MicroChat microChat**;**

private Body body**;**

private Red r**;**

De nuestras variables de instancia, se han añadido más JButton con los cuales se realizarán los cambios de ánimo a un objeto de tipo Body el cual es un conjunto de figuras creadas a partir de Java3D solo hay que hacer hincapié en que a este objeto tipo Body de los parámetros que se le pasan, el que modificará la textura será el último parámetro de tipo String llamado img

public Body**(**float x**,** float y**,** float z**,**String nombre\_mostrar**,**boolean movimiento**,**Frame frame**,**String img**){**

Regresando a nuestras variables de instancia tenemos otro objeto de tipo EventHandler que está en la clase EventHandler e implementa a la interfaz ActionListener en la cual se modificará la textura de nuestro objeto body utilizando el método changeTextureCab, como se observa en el código de abajo, se obtiene un objeto el cual compararemos con el botón que ha sido presionado, dependiendo de cuál ha sido presionado cambiará el valor de la variable turno.

class EventHandler **implements** ActionListener **{**

public void actionPerformed**(**ActionEvent e**)** **{**

Object obj**=**e**.**getSource**();**

**if(**obj **instanceof** JButton**){**

JButton btn**=(**JButton**)**e**.**getSource**();**

**if(**btn**==**BTNFeliz**)**

turno**=**0**;**

**if(**btn**==**BTNEnfermo**)**

turno**=**1**;**

**if** **(**btn**==**BTNAA2Mareado**)**

turno**=**2**;**

**if(**btn**==**BTNA4Sospechoso**)**

turno**=**3**;**

**if(**btn**==**BTNA1Feliz**)**

turno**=**4**;**

body**.**changeTextureCab**(**texture**,** nombres**[**turno**]);**

r**.**escribeRed**(new** Icono**(**"Tamagochi"**,** turno**));**

**}**

**}**

Como ya se ha mencionado el objeto de la clase EventHandler modificará la textura del objeto tipo Body, utilizando el método changeTextureCab, el cual recibe dos parámetros, uno de tipo Texture y otro de tipo String

public void changeTextureCab**(**Texture texture**,** String image**)** **{**

loader **=** **new** TextureLoader**(**image**,** "RGB"**,**frame1**);**

texture **=** loader**.**getTexture**();**

texture**.**setBoundaryModeS**(**Texture**.**CLAMP\_TO\_BOUNDARY**);**

texture**.**setBoundaryModeT**(**Texture**.**CLAMP\_TO\_BOUNDARY**);**

texture**.**setBoundaryColor**(new** Color4f**(**0.0f**,** 1.0f**,** 0.5f**,** 0f**));**

TextureAttributes texAttr **=** **new** TextureAttributes**();**

texAttr**.**setTextureMode**(**TextureAttributes**.**REPLACE**);**

ap**.**setTextureAttributes**(**texAttr**);**

ap**.**setTexture**(**texture**);**

**}**

Como podemos observar también creamos un objeto de tipo TextureLoader llamado loader la cual es una variable de instancia de la clase Body, este objeto TextureLoader recibe tres parámetros, de los cuales le hemos dado el nombre de una imagen, le hemos indicado que será RGB y finalmente le pasamos un objeto de tipo JFrame, una vez hecho lo anterior podemos modificar la textura como mejor nos parezca y finalmente aplicarla.

Retornamos a nuestras variables de instancia, y ahora tenemos un arreglo de Strings, en el cual están contenidos los nombres de las imágenes que utilizaremos para las texturas, después tenemos otra variable tipo int que será la posición en la cual nos ubicamos en nuestro arreglo de Strings, continuando tenemos un Objeto de tipo MicroChat la cual es un objeto de la clase MicroChat que extiende de la clase JPanel e implementa a la interfaz ActionListener

public class MicroChat **extends** JPanel **implements** ActionListener**{**

Y en pocas palabras como el nombre bien dice solo recibiremos mensajes.

Continuando con nuestras variables de instancia tenemos un objeto tipo Body el cual ya se ha explicado y finalmente un objeto tipo Red, el cual está en la clase Red y creo que es el más importante de todos los objetos ya que si el no podemos establecer un vínculo entre un cliente y un servidor.

public class Red **{**

private Socket cliente**;**

private ObjectInputStream oisNet**;**

private ObjectOutputStream oosNet**;**

private int puerto**=**5000**;**

private LeeRed lr**;**

**...**

**}**

En esta clase (Red) tenemos diversas variables de instancia, primeramente un objeto tipo Socket, el cual es llamado cliente, posteriormente tenemos dos objetos uno tipo ObjectInputStream y otro tipo ObjectOutputStream, los cuales utilizaremos para nuestros flujos de entrada y salida. También tenemos dos objetos más uno tipo LeeRed el cual ya he explicado y otro que es el puerto al cual se conectará nuestro Socket cliente.

public Red**(**LeeRed lr**)** **{**

**this.**lr**=**lr**;**

setUpNetworking**();**

**}**

Una vez mencionado lo anterior tenemos el constructor de la clase Red, el cual recibe un objeto tipo LeeRed el cual será igualado a nuestra variable de instancia de tipo LeeRed y se llama al método setUpNetworking();

public void setUpNetworking**()** **{**

int i**=**0**;**

String host **=** JOptionPane**.**showInputDialog**(**"Escriba dir.IP"**,** "localhost"**);**

**while(**i**==**0**){**

System**.**out**.**println**(**"Esperando por el servidor . . ."**);** i**=**1**;**

**try** **{**

cliente**=new** Socket**(**host**,** puerto**);**

**}** **catch** **(** IOException e**)** **{**

System**.**out**.**println**(**"Fallo creacion Socket"**);** i**=**0**;**

**}**

**}**

System**.**out**.**println**(**"Connectado al servidor."**);**

**try** **{**

oisNet **=** getOISNet**(**cliente**.**getInputStream**());**

oosNet **=** getOOSNet**(**cliente**.**getOutputStream**());**

**}** **catch** **(**IOException e**)** **{**

e**.**printStackTrace**();**

System**.**out**.**println**(**"Error al crear los fujos de objeto"**+**e**);**

**}**

**(new** Thread**(** **new** IncomingReader**(**lr**,** oisNet**)** **)).**start**();**

**}**

Como es posible observar en el código de arriba tenemos la implementación del método setUpNetworking() en la cual primeramente se pide ingresar una dirección IP y el host, una vez hecho lo anterior, se inicia un ciclo while mientras i == 0 mientras tanto se intenta realizar la conexión con el servidor y se repite este ciclo hasta que se pueda conectar si no se logra conectar se envía un mensaje a consola, por otro lado si la conexión ha sido exitosa se crea un objeto tipo Socket llamado cliente el cual es una variable de instancia de la clase Red, una vez que se ha creado el objeto satisfactoriamente se obtienen los flujos de entrada y salida de nuestro Socket cliente utilizando nuestros objetos ObjectInputStream y ObjectOutputStream llamados oisNet y oosNet respectivamente, si no se han obtenido los flujos como se debería de haber hecho se manda un mensaje de error. También hay que recalcar que todo lo anterior o bien la mayoría está implementada dentro de un try, ya que tenemos llamadas a métodos que pueden lanzar una excepción.

Finalmente, una vez ocurrido lo anterior se crea un objeto tipo Thread, el cual como parámetro tiene un nuevo objeto tipo IncomingReader que recibe un objeto LeeRed y ObjectInputStream, finalmente nuestro Thread se deja en estado listo.

public static void main**(**String**[]** args**)** **{**

**new** Tamagochi**();**

**}**

Retornando a nuestra clase Tamagochi solo nos resta ejecutar nuestro programa esto se realiza solo creando un objeto de la clase Tamagochi

**Conclusión:**

Para concluir el uso de Sockets no es complicado, solo hay saber cómo es que funcionan los Sockets, como es que funcionan los flujos de entrada y salida, además de que hay que conocer un poco sobre Excepciones ya que las llamadas a métodos en los sockets pueden lanzar alguna excepción. Además, el uso de sockets nos permite crear diferentes programas en diferentes computadoras y que sea posible que interactúen entre sí.