

INSTITUTO POLITECNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO (ESCOM)

PROGRAMACIÓN ORIENTADA A OBJETOS

NOMBRE DEL ALUMNO:

* SANTOS MÉNDEZ ULISES JESÚS

PRÁCTICA:

* SOCKETS CLIENTES

NÚMERO DE PRÁCTICA: 45

OPCIÓN: 1

* CHAT 3D

FECHA DE ENTREGA:

* 18/06/2021

GRUPO:

* 2CM11

**Sockets Clientes**

**Introducción**

Se desarrollo un programa haciendo uso de lo visto teóricamente en clase sobre los Sockets, consiste en el desarrollo con ayuda de Java 3D para hacer un personaje de tamagochi y que tenga diversos estados de ánimo, se hizo uso de muchas clases que permitieron hacer el cuerpo, el comportamiento, y el acceso a red.

**Marco Teórico**

Java fue pensado en ofrecer la máxima transparencia posible al trabajar en redes, a partir de abstraer los detalles específicos y diferentes de las diferentes redes involucradas en la comunicación.

Con objeto de liberar al programador Java de gran parte del trabajo de red de bajo nivel, Java proporciona dos formas básicas de trabajar en red: mediante Sockets y URL.

Los sockets son un mecanismo cliente/servidor que funciona a nivel de transporte (TCP Y UDP) y son los más utilizados. Consisten en comunicar un proceso entre dos puntos finales o puertos de dos máquinas conectadas mediante una red.

Los sockets representan una abstracción de conexión de red, que permite a dos programas, ejecutándose en dos máquinas o sistemas de comunicación diferentes, intercambiarse datos primitivos, en forma de streams o secuencias de bytes.

De esta forma, entre dos sockets, se abre un canal de comunicación, en la cual se escriben y leen datos en formas de streams.

**OutputStream**

Es una clase abstracta que representa el flujo de bytes de salida. Un output stream acepta bytes de salida que escribe en un dispositivo genérico. Un dispositivo de salida se considera apropiado mientras sea capaz de recibir o almacenar los datos que le llegan de un flujo de salida.

**InputStream**

Es una clase abstracta que representa un flujo de datos de entrada. Un input stream acepta bytes de entrada que lee de un dispositivo genérico. Un dispositivo de entrada se considera apropiado si puede trabajar como fuente de datos y coincide con los dispositivos de salida.

Diagrama

Descripción generada automáticamente

**Desarrollo**

1. Se hizo uso del paquete java.io es el encargado de gestionar las operaciones de entrada/salida. Entrada estándar sería System.in (es un objeto InputStream), salida estándar sería System.out y salida estandar de errores sería System.err (las salidas son objetos PrintStreams). Además, que a través de las clases del paquete java.net, los programas Java pueden utilizar TCP o UDP para comunicarse a través de Internet. Las clases URL, URLConnection, Socket, y SocketServer utilizan TCP para comunicarse a través de la Red.

Texto, Carta

Descripción generada automáticamente

1. Se hace uso de las clases VerySimpleCharServer y la clase anidada ClientHandler que implementa la interfaz Runneable donde se usa un ArrayList que va a contener flujos de salida para objetos (ObjectOutputStream). Además se declara un socket. Se va a implementar el método abstracto run (un hilo por cada client donde el hilo es creado por ClientHandler). En el constructor inicializamos la variable sock apartir del parámetro que recibe le constructor y luego se usa sock para obtener un flujo de entrada.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

Texto

Descripción generada automáticamente

1. Se crea el método run dentro del ClientHandler. Se hace uso de un ciclo infinito. La idea es que un objeto de tipo ClientHandler atiende a un cliente entonces en el método run vamos a estar esperando que el objeto llegue de la red. El método me va a devolver algo de tipo objeto, en la parte del reader.readObject() se va a tratar de leer un objeto de la red si esto tiene éxito entonces en obj guardamos el objeto que llegó de la red y luego el servidor manda a llamar a tellEveryone y le pasa el objeto que llegó de la red.

Texto, Carta

Descripción generada automáticamente

1. Se crea el método main, donde crea un nuevo objeto de tipo VerySimpleChatServer y lo usamos para llamar al método go.

Texto

Descripción generada automáticamente

1. Se crea el método go donde se crea un ServerSocket con el puerto 5000(que está dentro de un try porque puede llamar a una excepción. El método go crea el objeto de tipo ArrayList donde se van a almacenar los flujos de salida. También se hace uso de un ciclo infinito.Hay que llamar a accept y bloquea el programa solo se desbloquea hasta que un cliente se conecte y entonces retorna un Socket que va a servir para comunicarse con el cliente.

Texto, Carta

Descripción generada automáticamente

1. Se crea el método tellEveryone, este método recibe un objeto como parámetro. El ArrayList es un contenedor y entonces los contenedores se pueden recorrer. Obtenemos un iterador y este va a servir para recorrer el contenedor. El iterador tiene un método que devuelve cierto si es que aun quedan elementos que recorrer. La primera vez que llamemos al método next vamos a obtener el primer elemento que esté en el ArrayList. Cuando llamemos de nuevo a next obtenemos el segundo y así sucesivamente.

Texto, Carta

Descripción generada automáticamente

1. Se importan las librerías necesarias. Se ve que se está usando Java 3D.

Texto, Carta

Descripción generada automáticamente

1. Creamos la clase Tamagochi que extiende JFrame e implementa la interfaz LeeRed.



1. En el constructor de la clase Tamagochi se inicializan las variables de instancia, los botones que en este caso se trata de 5 botones para cada estado de ánimo y se agregan con el método add. Se crea un objeto tipo EventHandler y se le coloca el nombre eh. Luego con el addActionListener en los botones vamos a suscribirnos a la acción más típica de estos y como parámetro le vamos a dar. Agregamos los botones a un panel, etc. Mandamos a llamar al método setup3DGraphics.

Texto

Descripción generada automáticamente

1. Se implementa el método main.

Texto

Descripción generada automáticamente con confianza media

1. Se crea la clase Red donde se declara la variable cliente tipo Socket, un objeto de flujo de salida y un objeto de flujo de entrada. En el constructor se inicializa a la variable lr con el parámetro del constructor. Se llama al método setUpNetworking..

Texto

Descripción generada automáticamente

Imagen que contiene interior, pájaro, ave

Descripción generada automáticamente

1. Se declara el método escribeRed, este recibe un objeto como parámetro. Usamos el método WriteObject de la clase oosNet, le pasamos el objeto y luego se llama al método flush donde se vacía el buffer.

Texto, Carta

Descripción generada automáticamente

1. Creamos la clase IncomingReader que implementa interfaz Runnable, donde se declara una variable lr tipo Leered y un objeto de flujo de salida y en el constructor se le da como parámetro LeeRed lr y ObjectInputStream oisNet, Se inicializan las variables lr y oisNet.

Texto

Descripción generada automáticamente

1. Se implementa el método run donde se va a usar un ciclo infinito. Donde lr es un objeto de una clase que implementa la interfaz LeeRed.

Texto, Carta

Descripción generada automáticamente

1. Se crea la clase Body donde se va a hacer la parte del diseño del cuerpo del muñeco, esto de diseño de objetos en Java3D.

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

1. Procedemos a compilar cada una de las clases

Texto

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Le damos a aceptar y procedemos a ejecutar el servidor

Texto

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente

Interfaz de usuario gráfica

Descripción generada automáticamente

**Conclusión**

En conclusión, se logró desarrollar la conexión del cliente al colocar la dirección IP que se solicita se logra tener en la consola el mensaje que hay una conexión.