PRACTICA 2 UF 3

Comunicación entre procesos

Vamos a hacer un juego con comunicación entre PCs usando sockets e inteligencia de juegos. La práctica debe hacerse entre dos, uno hace el servidor y el otro el cliente.

El cliente representará al jugador.

El servidor ofrece la posibilidad de jugar, resolverá las jugadas y además implementará inteligencia artificial del más alto nivel para simular las decisiones del segundo jugador.

El juego que se emulará es el popular “piedra, papel o tijera”. El programa debe permitir conectar en una LAN entre dos PCs.

Programa cliente:

Empieza conectando al servidor.

Pregunta al cliente si quiere “piedra, papel o tijera”.

Envía la elección al server

Recoge del server el resultado de la mano.

Repite la jugada hasta que uno llegue a 3. Indica quien gana el total

Programa servidor:

Ofrece el servicio de juego al cliente. Cuando esté listo pondrá por pantalla “juego listo… esperando cliente”

El sistema piensa piedra papel o tijera con 30% cada una.

Acepta la petición del cliente.

Calcula el resultado y envía al cliente los datos (elección servidor, elección cliente y resultado).

Hay que controlar posibles errores.

**CODIGO CLIENTE:**

**package Pt2;**

**import java.io.DataInputStream;**

**import java.io.DataOutputStream;**

**import java.io.IOException;**

**import java.net.InetAddress;**

**import java.net.Socket;**

**import java.net.UnknownHostException;**

**import java.util.Scanner;**

**public class ClienteTCP\_PPT {**

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

**Scanner lector = new Scanner(System.in);**

**// Leemos el primer parámetro, donde debe ir la dirección**

**// IP del servidor**

**InetAddress direcc = null;**

**try {**

**direcc = InetAddress.getByName("192.168.40.189");**

**} catch (UnknownHostException uhe) {**

**System.err.println("Host no encontrado : " + uhe);**

**System.exit(-1);**

**}**

**// Puerto que hemos usado para el servidor**

**int puerto = 1235;**

**// Para cada uno de los argumentos...**

**int contadorCliente = 0;**

**int contadorServidor = 0;**

**for (int n = 1; n < 1000; n++) {**

**Socket sckt = null;**

**DataInputStream dis = null;**

**DataOutputStream dos = null;**

**try {**

**// Convertimos el texto en número**

**boolean comprobar = true;**

**int opcion = 0;**

**// COMPRUEBO QUE LA OPCION INTRODUCIDA ES CORRECTAR SIN AFECTAR AL SERVIDOR, AUNQUE EL SERVIDOR TAMBIEN LA COMPRUEBA**

**while(comprobar) {**

**System.out.println("Elige las siguentes opciones:");**

**System.out.println("1 - Piedra");**

**System.out.println("2 - Papel");**

**System.out.println("3 - Tijera");**

**if(lector.hasNextInt()) {**

**opcion = lector.nextInt();**

**comprobar = false;**

**}else {**

**System.out.println("No has escrito una opcion correcta, volviendo a las opciones...");**

**lector.next();**

**}**

**}**

**// Creamos el Socket**

**sckt = new Socket(direcc, puerto);**

**// Extraemos los streams de entrada y salida**

**dis = new DataInputStream(sckt.getInputStream());**

**dos = new DataOutputStream(sckt.getOutputStream());**

**// Lo escribimos**

**dos.writeInt(opcion);**

**// Leemos el resultado final**

**// AQUI MIRO CON LA ULTIMA PALABRA QUIEN GANO LA ULTIMA RONDA Y A SI HAGO UN CONTADOR PARA SALIR DEL BUCLE Y TERMINAR EL JUEGO**

**String resultado = dis.readUTF();**

**String[] vector = resultado.split(":");**

**if(vector[1].equals("Cliente")) {**

**contadorCliente++;**

**}else if(vector[1].equals("Servidor")){**

**contadorServidor++;**

**}**

**// Indicamos en pantalla**

**System.out.println(resultado);**

**// y cerramos los streams y el socket**

**dis.close();**

**dos.close();**

**// AQUI MIRO QUIEN LLEGO PRIMERO A TRES Y LE DIGO QUIEN A SIDO EL GANADOR**

**if(contadorCliente == 3) {**

**System.out.println("Eres el ganador");**

**break;**

**}else if(contadorServidor == 3) {**

**System.out.println("Te gano la maquina");**

**break;**

**}**

**} catch (Exception e) {**

**System.err.println("Se ha producido la excepción : " + e);**

**}**

**try {**

**if (sckt != null)**

**sckt.close();**

**} catch (IOException ioe) {**

**System.err.println("Error al cerrar el socket : " + ioe);**

**}**

**}**

**}**

**}**

**CODIGO SERVIDOR:**

**package Pt2;**

**import java.io.\*;**

**import java.net.\*;**

**import java.util.Random;**

**class ServidorTCP\_PPT {**

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

**int contadorServidor = 0;**

**int contadorCliente = 0;**

**// Primero indicamos la dirección IP local**

**try {**

**System.out.println("LocalHost = " + InetAddress.getLocalHost().toString());**

**} catch (UnknownHostException uhe) {**

**System.err.println("No puedo saber la dirección IP local : " + uhe);**

**}**

**// Abrimos un "Socket de Servidor" TCP en el puerto 1235.**

**ServerSocket ss = null;**

**try {**

**ss = new ServerSocket(1235);**

**} catch (IOException ioe) {**

**System.err.println("Error al abrir el socket de servidor : " + ioe);**

**System.exit(-1);**

**}**

**int entrada;**

**String salida = null;**

**// Bucle que dura hasta que acabe el juego**

**try {**

**boolean repetir = true;**

**System.out.println("Juego listo... Esperando cliente.");**

**while (repetir) {**

**// Esperamos a que alguien se conecte a nuestro Socket**

**Socket sckt = ss.accept();**

**// Extraemos los Streams de entrada y de salida**

**DataInputStream dis = new DataInputStream(sckt.getInputStream());**

**DataOutputStream dos = new DataOutputStream(sckt.getOutputStream());**

**// Podemos extraer información del socket**

**// Nº de puerto remoto**

**int puerto = sckt.getPort();**

**// Dirección de Internet remota**

**InetAddress direcc = sckt.getInetAddress();**

**// Leemos datos de la peticion**

**entrada = dis.readInt();**

**System.out.println(entrada);**

**// Generamos la eleccion del servidor y elegimos ganador**

**int eleccionServidor = generarEleccion();**

**salida = elegirGanador(entrada, eleccionServidor);**

**// Sumamos al marcador del ganador**

**String[] ganador = salida.split(":");**

**if (ganador[1].equals("Cliente")) {**

**contadorCliente++;**

**} else if (ganador[1].equals("Servidor")) {**

**contadorServidor++;**

**}**

**// Devolvemos el resultado de la jugada**

**dos.writeUTF(salida);**

**// Cerramos los streams**

**dis.close();**

**dos.close();**

**sckt.close();**

**// Registramos en salida estandar**

**System.out.println(**

**"Cliente = " + direcc + ":" + puerto + "\tEntrada = " + entrada + "\tSalida = " + salida);**

**// Cerramos el bucle si alguno de los jugadores llega a 3**

**if (contadorCliente == 3 || contadorServidor == 3) {**

**repetir = false;**

**}**

**}**

**} catch (Exception e) {**

**System.err.println("Se ha producido la excepción : " + e);**

**}**

**}**

**// Metodo que calcula el numero de eleccion de la maquina de forma aleatoria**

**public static int generarEleccion() {**

**// Piedra es 1, Papel es 2 y Tijera es 3**

**int eleccionServidor = new Random().nextInt(3) + 1;**

**return eleccionServidor;**

**}**

**// Metodo que recibe el numero de eleccion del jugador y el servidor y devuelve**

**// el ganador**

**public static String elegirGanador(int eleccionCliente, int eleccionServidor) {**

**String resultado = "Error. Eleccion no valida: ";**

**if (eleccionCliente == eleccionServidor) {**

**resultado = "Empate: ";**

**} else if (eleccionCliente == 1) {**

**if (eleccionServidor == 2) {**

**resultado = "Papel gana a piedra. Ganador:Servidor";**

**} else if (eleccionServidor == 3) {**

**resultado = "Piedra gana a tijeras. Ganador:Cliente";**

**}**

**} else if (eleccionCliente == 2) {**

**if (eleccionServidor == 1) {**

**resultado = "Papel gana a piedra. Ganador:Cliente";**

**} else if (eleccionServidor == 3) {**

**resultado = "Tijeras gana a papel. Ganador:Servidor";**

**}**

**} else if (eleccionCliente == 3) {**

**if (eleccionServidor == 1) {**

**resultado = "Piedra gana a tijeras. Ganador:Servidor";**

**} else if (eleccionServidor == 2) {**

**resultado = "Tijeras gana a papel. Ganador:Cliente";**

**}**

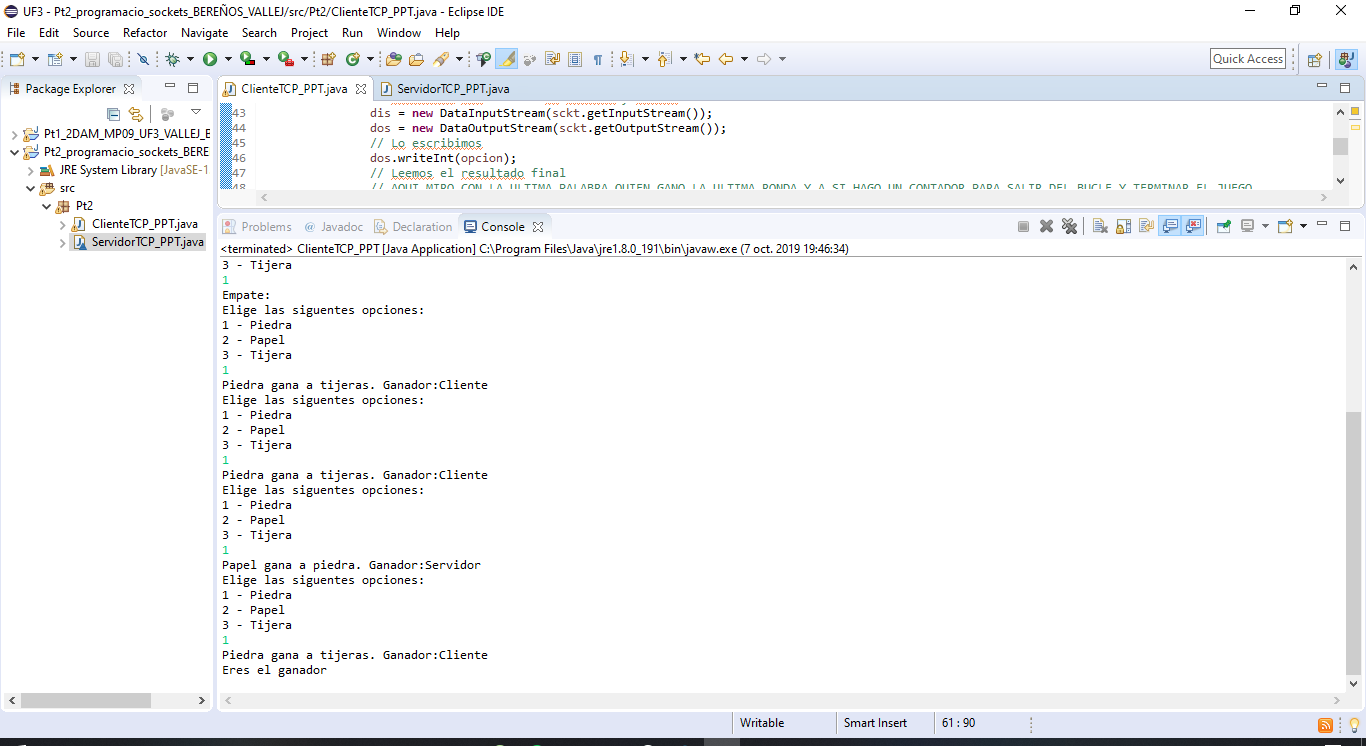
**}**

**return resultado;**

**}**

**}**

**FOTO DE COMPROBACIÓN DE FUNCIONAMIENTO.**

****