PRACTICA 4 UF 3

Programación RMI. El ahorcado

Vamos a hacer un juego con comunicación entre PCs usando RMI. 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 al ahorcado, resuelve las jugadas y lleva el control de la partida.

El juego que se emulará es el popular “ahorcado”. El programa debe permitir conectar en una LAN entre dos PCs.

Programa cliente:

Empieza conectando al servidor.

Le muestra al cliente el tamaño de la palabra

Pregunta al cliente una letra.

Envía la letra al server que resuelve la jugada.

Muestra al cliente el tamaño de la palabra con las adivinadas

Acaba si se completa la palabra

Programa servidor:

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

Cuando se conecte el cliente aparece por pantalla cliente conectado...

El servidor se piensa una palabra de una lista de 100 de un fichero o bien permite entrar otra

Comprobar letras que recibe del cliente

Comprobar si acaba la partida (8 intentos) y envia resultado al cliente

Cuando el cliente adivina indica “palabra adivinada”

Entregar el código fuente de las dos partes, comentado. Entregar un par de ejecuciones (una que gane cada uno)

CODIGO SERVIDOR

package ahorcado;

import java.io.BufferedReader;

import java.io.File;

import java.io.FileReader;

import java.net.InetAddress;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.rmi.server.UnicastRemoteObject;

import java.util.ArrayList;

public class ServidorTCP\_Ahorcado implements RMIAhorcInterface {

private ArrayList<String> palabras = new ArrayList<String>();

private static ArrayList<String> aPalabra = new ArrayList<String>();

private static ArrayList<String> aGuiones = new ArrayList<String>();

static ServidorTCP\_Ahorcado sta = new ServidorTCP\_Ahorcado();

static String palabra = null;

static String devPalabra = new String();

static int contador = 0;

@Override

public int largoPalabra() throws RemoteException {

return palabra.length() - 1;

}

@Override

public String devolverPalabra(String letra) throws RemoteException {

devPalabra = "";

for (int i = 0; i < aPalabra.size(); i++) {

if(letra.equalsIgnoreCase(aPalabra.get(i).toString())) {

aGuiones.set(i, letra);

}

}

for (int i = 0; i < aGuiones.size(); i++) {

devPalabra = devPalabra + aGuiones.get(i).toString() + " ";

if(aGuiones.toString().equalsIgnoreCase(aPalabra.toString())) {

return "Has ganado";

}

}

if(!aPalabra.contains(letra)) {

contador = contador + 1;

if(contador >= 8){

return "Has perdido";

}

}

return devPalabra;

}

@Override

public boolean finalizar() throws RemoteException {

if(contador >= 8 || aGuiones.toString().equalsIgnoreCase(aPalabra.toString())) {

return false;

}

return true;

}

public String escogerPalabra() {

String cadena = null;

try {

File f = new File("palabras.txt");

FileReader fr = new FileReader(f);

BufferedReader br = new BufferedReader(fr);

while((cadena = br.readLine()) != null) {

palabras.add(cadena);

}

System.out.println(palabras.toString());

int posicion = (int) (Math.random() \* 100 + 0);

cadena = palabras.get(posicion).toString();

System.out.println("Palabra escogida por el servidor: " + cadena);

} catch (Exception e) {

System.out.println(e);

}

return cadena;

}

public void guiones() {

for (int i = 0; i < palabra.length() - 1; i++) {

aPalabra.add(String.valueOf(palabra.charAt(i)));

}

System.out.println(aPalabra.toString());

for (int i = 0; i < palabra.length() - 1; i++) {

aGuiones.add("\_");

}

System.out.println(aGuiones.toString());

}

public static void main(String[] args) {

// CREAMOS UN REGISTRO DE OBJETOS REMOTOS

Registry reg = null;

// ABRIMOS EL REGISTRO EN EL PUERTO 5555

try {

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

reg = LocateRegistry.createRegistry(5555);

} catch (Exception e) {

System.out.println("Error: No se ha podido crear el registro");

e.printStackTrace();

}

// PONIENDO DATOS METODOS

palabra = sta.escogerPalabra();

sta.guiones();

//CREAMOS EL OBJETO SERVIDOR Y LO INSCRIBIMOS EN EL REGISTRO.

System.out.println("Creando el objeto servidor e inscribiendolo en el registro...");

System.out.println("Juego listo");

ServidorTCP\_Ahorcado serverObject = new ServidorTCP\_Ahorcado();

//FINALMENTE LE DAMOS UN NOMBRE AL REGISTRO "Ahorcado" POR EL CUAL EL CLIENTE PODRA ENTRAR Y RESOLVER SUS OPERACIONES.

try {

reg.rebind("Ahorcado", (RMIAhorcInterface) UnicastRemoteObject.exportObject(serverObject, 0));

} catch (Exception e) {

System.out.println("Error: No se ha podido inscribir el objeto servidor");

e.printStackTrace();

}

}

}

CODIGO CLIENTE

package ahorcado;

import java.rmi.RemoteException;

import java.rmi.registry.LocateRegistry;

import java.rmi.registry.Registry;

import java.util.Scanner;

public class ClienteAhorcado {

public static void main(String[] args) throws RemoteException {

Scanner sc = new Scanner(System.in);

String letra = new String();

RMIAhorcInterface ahorcado = null;

try {

System.out.println("Localizando registro de objetos remotos...");

Registry registry = LocateRegistry.getRegistry("192.168.40.44", 5555);

System.out.println("Obteniendo el stub del objeto remoto...");

ahorcado = (RMIAhorcInterface) registry.lookup("Ahorcado");

} catch (Exception e) {

e.printStackTrace();

}

if (ahorcado != null) {

System.out.println("Realizando operaciones con el objeto remoto...");

try {

System.out.println("La palabra tiene " + ahorcado.largoPalabra() + " letras.");

boolean repetir = true;

while (repetir) {

System.out.println("Introduce una letra:");

letra = sc.next();

letra = String.valueOf(letra.charAt(0));

System.out.println(ahorcado.devolverPalabra(letra) + "\n");

repetir = ahorcado.finalizar();

}

} catch (Exception e) {

e.printStackTrace();

}

System.out.println("Terminado");

}

}

}

FOTO SERVIDOR

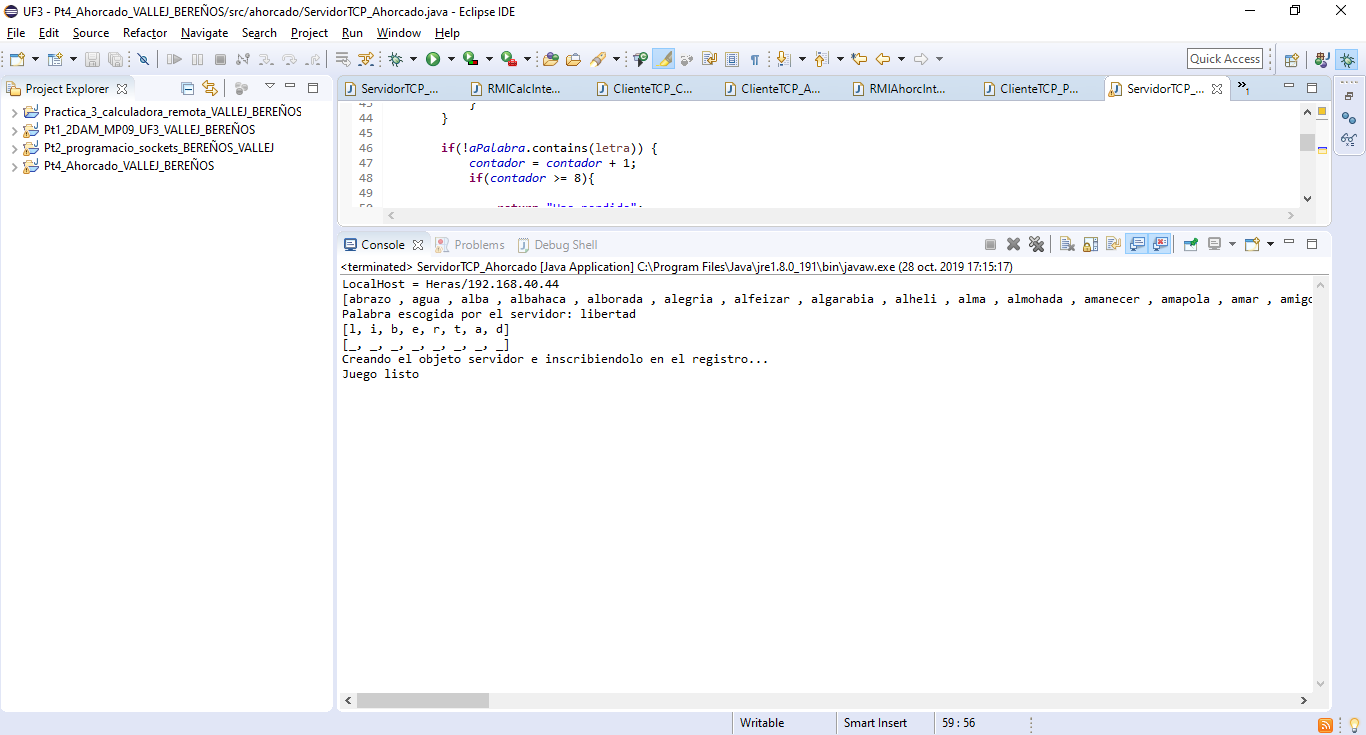


FOTO CLIENTE

