7-9-2017

Juego Ahorcado

Implementado en JAVA con DatagramSocket

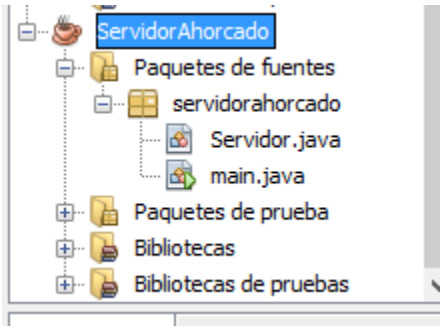
Jesús Andrés Acendra Martínez;
Juan Diego Benítez Arias
UNIVERSIDAD DE CARTAGENA

Contenido

Servidor DatagramSocket

Lo primero es crear el código del servidor DatagramSocket del juego, el servidor es el encargado de asignar a cada cliente la palabra con la que se va a jugar esa partida, así como guardar la información de este.

Se crea un proyecto **Java Application** con la siguiente estructura



Dentro de la estructura del servidor se encuentran solo dos clases “Servidor.java” donde se implementarán algunos métodos y “main.java” que es la clase principal.

1) Clase Servidor

Se crea un atributo de tipo **vector String** con el nombre de **diccionario** en el cual se tendrán todas las palabras que tendrá el juego.

```
    private String[] diccionario = {"ABEJA", "AEROPUERTO", "COMPUTADOR", "OSO",
        "JAVA", "NEVERA", "PROGRAMA", "INFORMATICA", "COMPUTACION", "COMPUTADOR","CORAZON","BANANO","PLATANO",
        "AUTOMOVIL", "PERRO", "COLOMBIA", "LONDRES", "CEPILLO", "BRAZO", "CABEZA", "CUERPO","DEPORTE","SALUD",
        "ANONYMOUS", "CUADERNO", "PANTALLA", "UBUNTU", "SEMAFORO", "LINUX", "LOBO","AMOR","MOSCA","ZANAHORIA",
        "PINGUINO", "HACKER", "SISTEMA", "ELEFANTE", "CASCADA", "JUEGOS","LARGO","BONITO","IMPOSIBLE","UNIDOS","ZOMBIE",
        "MATEMATICAS", "CALCULO", "ALGEBRA", "DICCIONARIO", "BIBLIOTECA", "COCINA","PELICULA","COMERCIAL","GRANDE","PEQUEÑO",
        "GATO", "SAPO", "JIRAFA", "FERROCARRIL", "FACEBOOK", "PERSONA","BICICLETA","CONTROL","PANTALON","AEROSOL",
        "ERROR", "COPA", "COPA", "PROGRAMADOR", "LICENCIA", "NUEVE", "PROCESADOR","GARAJE","MODERNO","TABLA","DISCOTECA",
        "LENGUAJE", "PROGRAMACION", "HERRAMIENTAS", "INTERNET", "EJECUTAR", "PROYECTO","PERIODICO","COCODRILO","TORTUGA","CABALLO"
        "APLICACION", "PERA", "SOFTWARE", "ADMINISTRACION", "VENTANA", "MANTENIMIENTO","INFORMACION","PRESIDENTE","PERSONA","GENTE
        "NARANJA", "PRUEBA", "MANZANA", "JARRA", "CELULAR", "TELEFONO","CONTAMINACION","COLOR","ROMANO","ADIVINAR","MARCADOR",
        "INSTRUCCION", "CUADERNO", "CASA", "PALA", "ARBOL", "PUENTE", "PAPEL", "HOJA","HELICOPTERO","BARCO","GOLF","CARRERA",
        "TUBERIA", "PLOMERO", "FUTBOL", "BALONCESTO", "ESTADIO", "JEAN", "FUENTE", "LEOPARDO","REGLA","PRIMERO","SEGUNDO",
        "BLUSA", "CAMISA", "AGUA", "FUEGO", "INDUSTRIA", "AIRE","TIERRA","NATURALEZA","MIERCOLES","FOTOGRAFIA","LEON",
        "TIGRE"}; //90
```

Y contamos con un solo método llamado obtenerPalabra() el cual escoge aleatoriamente una palabra del vector de palabras y la devuelve.

```
public String obtenerPalabra()  { // metodo que devuelve la palabra que va a tener el cliente
    int num = (int)(Math.random()*(diccionario.length));
    return diccionario[num];
}
```

2) Clase Main

Se importan las siguientes librerías:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import java.net.SocketException;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Después seguimos a crear los objetos que nos serán de utilidad

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

    DatagramSocket socket; // Creamos un objeto de DatagramSocket
    Servidor server = new Servidor(); // Instanciamos un objeto de la clase Servidor
    System.out.print("Inicializando servidor... ");
```

Mostramos un mensaje que indique que el servidor se está iniciando.

```
try {
    socket = new DatagramSocket(6000);  // Creamos el servidor en el puerto 6000
    System.out.print("OK");  //
    byte[] mensaje_bytes = new byte[256];
    String mensaje = "";
    mensaje = new String(mensaje_bytes);
    String mensajeComp = "";

    DatagramPacket paquete = new DatagramPacket(mensaje_bytes, 256); // Creamos el paquete que sera recibido
    DatagramPacket envpaquete = new DatagramPacket(mensaje_bytes, 256); //Creamos el paquete que sera enviado
    int puerto; // Capturamos el puerto
    InetAddress address; // Para guardar las direccion de donde llega el paquete
    byte[] mensaje2_bytes = new byte[256];
```

Hacemos uso del try – catch para capturar posibles excepciones, creamos el DatagramSocket que hará las veces de Servidor en el puerto 6000 y mandamos un mensaje que el servidor se inició correctamente, se crean variables que contendrán los mensajes que llegaran al servidor y se enviaran del servidor; se crean estas variables de tipo “byte” ya que al estar haciendo uso de Datagramas la información se envía en array de bytes.

Los DatagramPacket son las clases que se encargan de transportar estos paquetes, como parámetros piden el array de bytes, la longitud del mensaje, la dirección IP del destinatario y el puerto, creamos objetos de estas clases para poder enviar información entre el servidor y el cliente.

```
while(true){
    try {
        // Recibimos el paquete
        socket.receive(paquete);
    } catch (IOException ex) {
        Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
    }
    // Lo formateamos
    mensaje = new String(mensaje_bytes).trim();

    //Obtenemos IP Y PUERTO
    puerto = paquete.getPort();
    address = paquete.getAddress();
```

Dentro de un bucle while para que siempre este corriendo el servidor resolvemos la comunicación del cliente y el servidor, con “socket.receive()” recibimos el mensaje que nos envía el cliente, a este mensaje que está en formato de bytes lo formateamos para que quede en formato de dato String y podamos operar más fácil con él. Guardamos la dirección IP y el puerto del cliente que envió el mensaje en las variables correspondiente.

Por ultimo

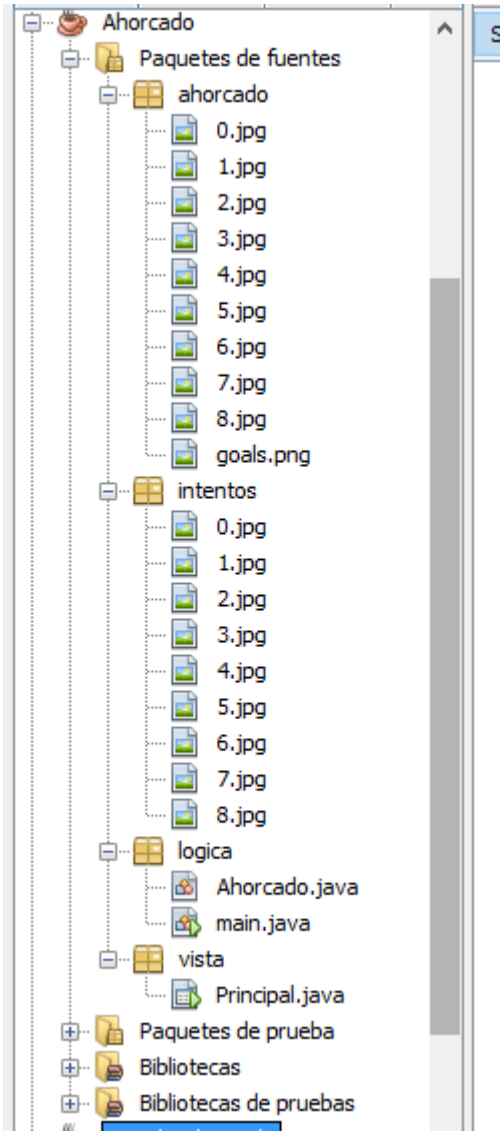
```
    if (mensaje.startsWith("palabra")) {
        mensajeComp = server.obtenerPalabra();
    }

    //formateamos el mensaje de salida
    mensaje2_bytes = mensajeComp.getBytes();

    //Preparamos el paquete que queremos enviar
    envpaquete = new DatagramPacket(mensaje2_bytes, mensajeComp.length(), address, puerto);

    try {
        // realizamos el envio
        socket.send(envpaquete);
    } catch (IOException ex) {
        Logger.getLogger(main.class.getName()).log(Level.SEVERE, null, ex);
    }

}
```

Verificamos que si el mensaje que nos está enviando el cliente es “palabra”, si es así obtenemos una palabra con ayuda del método obtenerPalabra() de la clase servidor y la guardamos en una variable.

En la variable *mensaje2_bytes* guardamos en valor de bytes la palabra obtenida anteriormente y posteriormente creamos el paquete que contiene el mensaje de respuesta al cliente, y finalmente lo enviamos.

Cliente DatagramSocket

Para el cliente se crea otro proyecto con la siguiente estructura



Esta aplicación del cliente contiene varios paquetes donde los paquetes “ahorcado” e “intentos” tienen recursos para las vistas de la aplicación, el paquete vista tiene la vista del juego, y el paquete lógica está compuesto por dos clases donde una de ellas es la clase main de programa.

1) Lógica del juego

En esta se crean todos los métodos necesarios para que funcione el juego

En el paquete lógica se crean dos clases: **Ahorcado** y **Main**

En la clase ahorcado se importan las siguientes librerías necesarias para el funcionamiento de la aplicación:

```
import java.io.IOException;
import java.net.DatagramPacket;
import java.net.DatagramSocket;
import java.net.InetAddress;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
```

Dentro de la clase se crean los siguientes objetos y atributos:

```
public class Ahorcado {
    JTextField campoPalabra;
    JLabel campoIntentos;
    JLabel campoErrores;
    JLabel largoPalabra;

    private boolean jugar = false;

    char[] palabra_secreta;
    private char[] palabra;

    int intentos = 0;
    boolean acerto = false;
}
```

Los primeros objetos corresponden a los elementos de la vista que modificaremos según el estado del juego.

Un atributo boolean para conocer el estado del juego.

Dos vectores de char para almacenar la palabra a adivinar y la palabra que ingresa el usuario.

Una variable entera para contar el número de errores que lleva el usuario y por ultimo un atributo boolean para saber si acertó o no a la letra dentro de la palabra.

Se implementa el método constructor por defecto

```
int intentos = 0;
boolean acerto = false;

public Ahorcado(){
}
```

2) Método Constructor

Se implementa un nuevo método constructor que recibirá por parámetros los siguientes elementos **JTextField campoPalabra, String palabra, JLabel errores, JLabel campoIntentos, JLabel largoPalabra.**

```
public Ahorcado(JTextField campoPalabra, String palabra, JLabel errores, JLabel campoIntentos, JLabel largoPalabra) {

}
```

Estos elementos se crearon previamente en la vista del programa y son los que cambiaran durante la ejecución del juego.

A cada atributo local se le asigna su igual recibido en parámetros

```
this.campoIntentos = campoIntentos;
this.campoPalabra = campoPalabra;
this.campoErrores = errores;
this.largoPalabra = largoPalabra;
```

Se guarda la palabra a adivinar, en forma de vector de char para separar cada letra.

```
//guardamos la palabra secreta
this.palabra_secreta = palabra.toCharArray();
```

```
//guardamos la palabra secreta
this.palabra_secreta = palabra.toCharArray();

String linea = "";

//colocamos lineas segun el tamaño de la palabra
for (int i = 0; i <= palabra_secreta.length - 1; i++) {
    linea += "_";
}

//convertimos la linea en vector de char y guardamos en la variable palabra
this.palabra = linea.toCharArray();

//colocamos las lineas a adivinar en el JtextField
this.campoPalabra.setText(linea);

//colocamos el icono de los intentos restantes
this.campoIntentos.setIcon(new javax.swing.ImageIcon(getClass().getResource("/intentos/0.jpg")));

//colocamos la imagen del ahorcado
this.campoErrores.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ahorcado/0.jpg")));

this.jugar = true;

this.largoPalabra.setText("La palabra tiene " + this.palabra_secreta.length + " letras.");
```

Se crea la línea con la cantidad de letras de la palabra a adivinar, se convierte en un vector de char y se guarda para validar posteriormente.

Se envía al JTextField campoPalabra de la vista la línea con el número de letras de la palabra a adivinar.

Se envía al JLabel campoIntentos la imagen del número de intentos restantes.

Se envía al JLabel campoErrores la imagen del ahorcado.

La variable de juego se coloca en true para indiciar que empezó el juego.

Se envía al JTextFiled largoPalabra el texto que le dice al usuario el número de letras de la palabra a adivinar

3) Método para realizar la petición al servidor

```
public String conectarse() throws IOException {
    String palabra1 = "";
    DatagramSocket socket;
    InetAddress address;
    byte[] mensaje_bytes = new byte[256];
    String mensaje = "";
    mensaje_bytes = mensaje.getBytes();
```

Este método conectarse() crea la conexión entre el cliente y el servidor y nos devuelve un String que sea la palabra con que el cliente jugara. Se declaran las variables con las que se trabajara posteriormente.

```
//Paquete
DatagramPacket paquete;
String cadenaMensaje = "";
DatagramPacket servPaquete;
```

Se crean objetos DatagramPacket para poder comunicarse entre el cliente y el servidor.

```
byte[] RecogerServidor_bytes = new byte[256];
socket = new DatagramSocket(); // Creamos el socket del cliene
address = InetAddress.getByName("localhost"); // Direccion del servidor
```

Instanciamos el objeto DatagramSocket que hará las veces de cliente y guaramos la dirección IP del servidor en una variable.

```
mensaje = "palabra"; // Mensaje que se enviara al servidor
mensaje_bytes = mensaje.getBytes(); //Pasamos el mensaje a bytes
paquete = new DatagramPacket(mensaje_bytes, mensaje.length(), address, 6000); // Creamos el paquete
socket.send(paquete); // Enviamso el paquete
```

Se crea el paquete que el cliente enviara al servidor, se pasa como parámetros el mensaje que es “palabra” pero en bytes, la longitud del mensaje, la dirección que será la del servidor y el puerto, por último se envía el paquete.

```
RecogerServidor_bytes = new byte[256]; // variable que recoge la respuesta del servidor

//Esperamos a recibir un paquete
servPaquete = new DatagramPacket(RecogerServidor_bytes, 256);
socket.receive(servPaquete);
```

Creamos el paquete que va a contener la respuesta del servidor, y finalmente recibimos el mensaje que nos envía el servidor y lo guardamos en “servPaquete”.

```
        //Convertimos el mensaje recibido en un string
        cadenaMensaje = new String(RecogerServidor_bytes).trim();

        //Imprimimos el paquete recibido
        System.out.println(cadenaMensaje);
        palabra1 = cadenaMensaje;

        return palabra1;
    }
```

Para finaliza con el método, pasamos a formato String el mensaje de respuesta que nos envió el servidor el cual es la palabra con que se jugara y finalmente se retorna la palabra.

4) Validar la letra ingresada por el cliente

```
public void validarPalabra(char letraIngresada) {
    if (jugar) {

        String letrasAcertadas = "";

        //validamos los intentos restantes
        if (this.intentos == 7) {

            this.campoErrores.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ahorcado/8.jpg")));

            String respuesta = "";

            for (int i = 0; i <= this.palabra_secreta.length - 1; i++) {
                respuesta = respuesta + this.palabra_secreta[i];
            }

            JOptionPane.showMessageDialog(null, "Lo siento perdiste, la palabra era " + respuesta, "Perdiste!", JOptionPane
        } else {
            //evaluamos si ha adivinado la palabra comparando cada letra
            for (int i = 0; i <= palabra_secreta.length - 1; i++) {

                //validamos que la letra este en la palabra
                if (this.palabra_secreta[i] == letraIngresada) {
                    this.palabra[i] = letraIngresada;
                    this.acerto = true;
                }

                letrasAcertadas += this.palabra[i]; //guardamos la palabra con las letras acertadas
```

Se valida si el juego no ha terminado, si es así, se crea un atributo de tipo String llamado **letrasAcertadas** en el que se guardan todas las letras que el usuario ha adivinado.

Se valida si el usuario ha perdido, si es así, se muestra la imagen del ahorcado y una ventana mostrando cual era la palabra a adivinar.

Si no ha perdido, se hace un for donde se compara que la letra que ha ingresado el cliente este en la palabra, si es así, se cambia la variable acerto a true para indiciar que adivino la letra, luego se guardan todas las letras acertadas en la variable letrasAcertadas.

```
//si no acerto, mostramos el intento fallido y el ahorcado

if (this.acerto == false) {

    intentos += 1; //Numero de errores

    campoIntentos.setIcon(new javax.swing.ImageIcon(getClass().getResource("/intentos/" + this.intentos + ".jpg")));
    campoErrores.setIcon(new javax.swing.ImageIcon(getClass().getResource("/ahorcado/" + this.intentos + ".jpg")));

    //Mostramos un mensaje avisando que erró la letra
    if (intentos < 8) {
        JOptionPane.showMessageDialog(null, "Fallaste! \n\n\t Te quedan " + ((8) - this.intentos) + " intentos.");
    }

} else {
    this.acerto = false;
}

this.campoPalabra.setText(letrasAcertadas);
//comprobamos el estado del juego

validarGano();
```

Si no acertó la letra se aumenta el contador de errores y se muestran las imágenes del ahorcado y los numero de errores cometidos, además se muestra una ventana diciendo el número de intentos restantes, si acertó, se cambia la variable a false para realizar las validaciones con las nuevas letras ingresadas.

Al final se envía a la vista la palabra con las letras que ha acertado el usuario y se valida si ya ganó el juego.

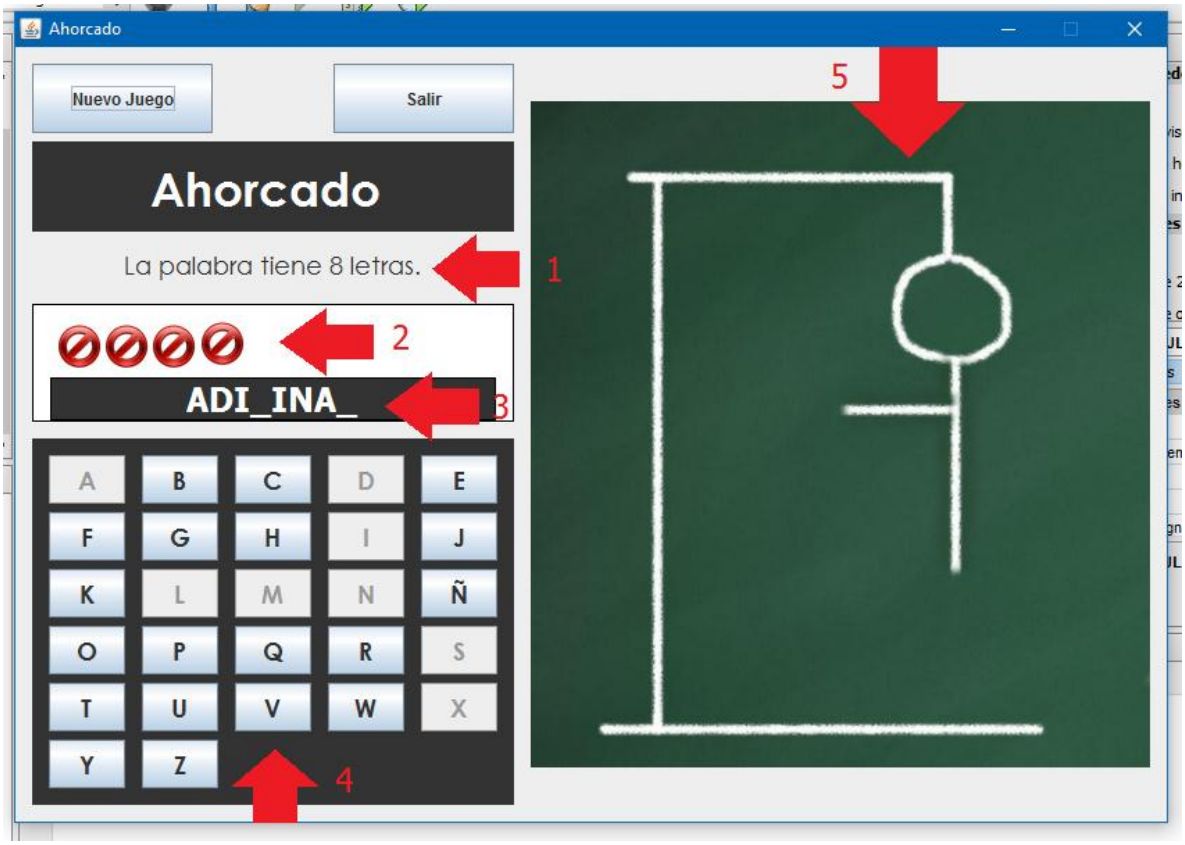
5) Validar si gano el juego

```
private void validarGano() {

    boolean gano = false;

    for (int i = 0; i <= this.palabra_secreta.length - 1; i++) {
        if (this.palabra[i] == this.palabra_secreta[i]) {
            gano = true;
        } else {
            gano = false;
            break;
        }
    }

    if (gano) {
        JOptionPane.showMessageDialog(null, "!! Felicidades !! \n Adivinaste la palabra", "Ganaste", 0, new javax.swing.I
    }
}
```

Se crea una variable booleana para conocer el estado del juego, luego se compara letra por letra si la ingresada por el usuario es igual a la palabra a adivinar, si es así la variable gano se vuelve true, si hay algún error es false y se rompe el ciclo, indicando que no ha ganado.

Si ganó, se muestra un mensaje mostrando al usuario que ganó el juego.

6) Vista del juego



La vista del usuario cuenta con variaros elementos:

1) **JTextField largoPalabra:** En este se le muestra al cliente cuantas letras tiene la palabra a adivinar.
2) **JLabel errores:** En este se muestran el número de errores se han cometido.
3) **JTextField palabra_Oculta**: En este se muestran las letras que ha adivinado el cliente.
4) **JPanel con Botones:** Un botón para cada letra del alfabeto, cuando se presionan se deshabilitan porque solo se puede usar cada letra una vez.
5) **JLabel ahorcado:** En este se tiene la imagen que muestra el dibujo del ahorcado según los errores que lleve el cliente.

Se crea un Objeto de la clase ahorcado.

```
public class Principal extends javax.swing.JFrame {

    Ahorcado juego ;
```

Se crea un método llamado nuevoJuego en el que se instancia el objeto juego y se llama al método conectarse para realizar la petición al servidor y se guarda la palabra que este envía, con esto se llama al método constructor de la clase Ahorcado que recibe por parámetros los elementos de la vista que serán modificados durante la ejecución del juego.

Al final se llama al método habilitarLetra que habilita todos los botones de la vista para el nuevo juego.

```
public void nuevoJuego(){

    juego = new Ahorcado();

    try {
        String palabra = juego.conectarse();
        juego = new Ahorcado(Palabra_oculta, palabra, ahorcado, errores, largoPalabra);
    } catch (IOException ex) {
        Logger.getLogger(Principal.class.getName()).log(Level.SEVERE, null, ex);
    }
    habilitarLetra();
}
```

En los botones se llama al método de la Clase Ahorcado que valida si la letra está o no en la palabra, comprobando cada pulsación el estado del juego.

```
private void BActionPerformed(java.awt.event.ActionEvent evt) {
    juego.validarPalabra('B');
    B.setEnabled(false);
}
```

Por último, se crea la clase main en el paquete lógica y se crea un objeto de la ventana del juego y se muestra al cliente.

```
public class main {

public static void main(String[] args) {

    Principal ventanita = new Principal();
    ventanita.setLocationRelativeTo(null);
    ventanita.setVisible(true);

}

}
```