

A dark blue vertical bar is positioned on the left side of the page. A blue arrow-shaped banner points to the right from this bar, containing the date '7-9-2017'. Below the banner, several thin, curved lines in shades of blue and grey sweep upwards and to the right, creating an abstract, organic shape.

7-9-2017

Juego Ahorcado

Implementado en JAVA con Sockets

Jesús Andrés Acendra Martínez; Juan Diego Benítez
Arias

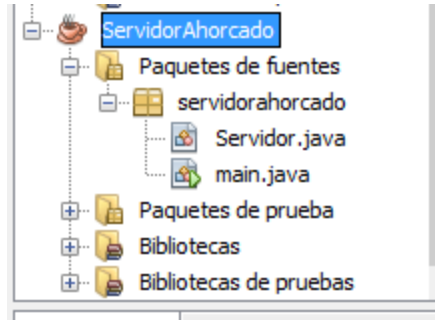
UNIVERSIDAD DE CARTAGENA

Contenido

Servidor Socket.....	2
1) Clase Servidor	2
2) Clase Main	6
Cliente Socket.....	9
1) Lógica del juego.....	9
2) Método Constructor.....	11
3) Método para realizar la petición al servidor	13
4) Validar la letra ingresada por el cliente	13
5) Validar si gano el juego	14
6) Vista del juego	15

Servidor Socket

Lo primero es crear el código del servidor socket del juego, se crea un proyecto **Java Application** con la siguiente estructura

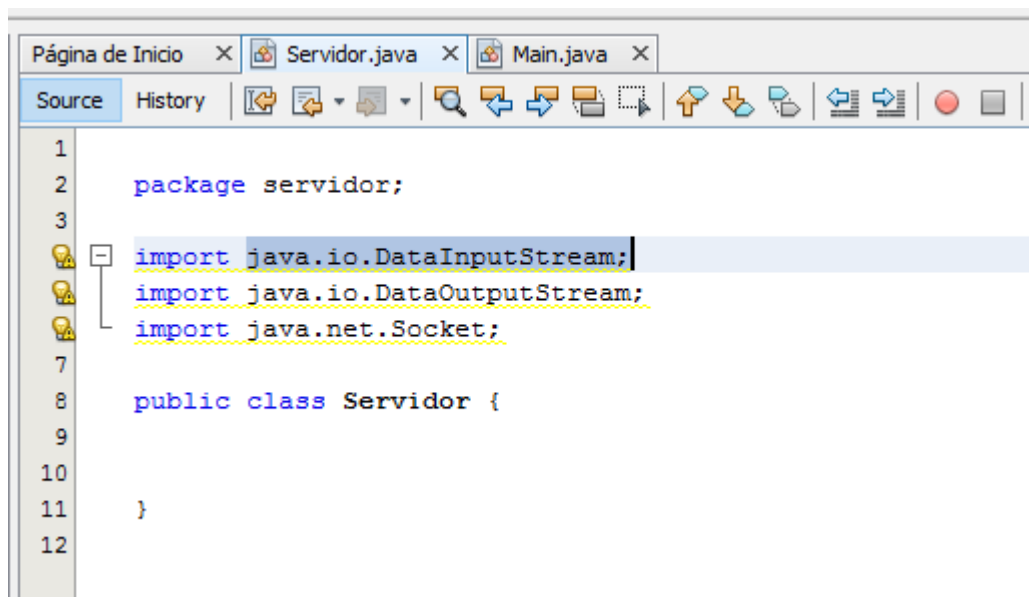


1) Clase Servidor

Se importan las librerías **java.net.Socket**, **java.io.DataInputStream** y **java.io.DataOutputStream**.

La librería **java.net.Socket** contiene las clases y métodos necesarios para crear un servidor socket.

Las librerías **java.io.DataInputStream** y **java.io.DataOutputStream** contienen las clases y métodos necesarios para enviar y recibir mensajes hacia y desde el cliente.



Ahora se crean tres objetos, **private Socket socket**, **private DataOutputStream dos** y **private DataInputStream dis**.

Guía para realizar juego de El Ahorcado en JAVA con sockets

Estudiantes: Jesús Acendra – Juan Benítez

Universidad de Cartagena - 2017

```
package servidor;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.net.Socket;

public class Servidor {

    private Socket socket;
    private DataOutputStream dos;
    private DataInputStream dis;
}
```



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.

```
public class Servidor {

    private Socket socket;
    private DataOutputStream dos;
    private DataInputStream dis;
    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", "FELICULA", "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", "C",
    "APLICACION", "PERA", "SOFTWARE", "ADMINISTRACION", "VENTANA", "MANTENIMIENTO", "INFORMACION", "PRESIDENTE", "PERSONA",
    "NARANJA", "PRUEBA", "MANZANA", "JARRA", "CELULAR", "TELEFONO", "CONTAMINACION", "COLOR", "ROMANO", "ADIVINAR", "MARCADO",
    "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"};
}
```

Se crea otro atributo de tipo **vector de char** llamado **palabra_secreta**, este servirá para guardar la palabra a adivinar con cada letra separada en una posición del vector.

```
private char[] palabra_secreta;
}
```



Ahora se implementa el método constructor de la clase Servidor, este recibirá dos parámetros: un **Socket** y un **id de sesión de usuario** de tipo entero.

```

private char[] palabra_secreta;

public Servidor(Socket socket, int idSession) {
    }
}

```

Dentro del método se asigna el socket que se recibe por parámetro al socket que tenemos en la clase.

```

public Servidor(Socket socket, int idSession) {
    this.socket = socket;
}

```

Se instancian los objetos para la entrada y salida de mensajes del servidor de la siguiente forma:

```

private char[] palabra_secreta;

public Servidor(Socket socket, int idSession) {
    this.socket = socket;

    try {
        dos = new DataOutputStream(socket.getOutputStream());
        dis = new DataInputStream(socket.getInputStream());
    } catch (IOException ex) {
        Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null, ex);
    }
}

```

Para prevenir excepciones, se encierra todo el bloque con un **try-catch** en el cual se captura una excepción de tipo **IOException** y se crea un **Logger** para llevar el registro de los fallos, se deben importar las librerías: **java.io.IOException**, **java.util.logging.Level** y **java.util.logging.Logger**

Ahora se asigna una palabra al atributo **palabra_secreta**, se toma una palabra del diccionario aleatoriamente, para esto se crea un método llamado **Ramdon** el cual devuelve un String con la palabra escogida, el código usado es el siguiente:

```
private String Random() {
    int num = (int) (Math.random() * (diccionario.length));
    return diccionario[num];
}
```

Se genera un número aleatorio entre cero y el tamaño del vector **diccionario** con la clase **Math** y su método **random**, se hace casteo de este número para que sea entero y se devuelve la palabra dentro del vector en esa posición.

Con el método creado, dentro del método constructor se asigna la palabra a adivinar al atributo **palabra_secreta** llamando al método **random** y convirtiendo esta palabra en un char con el método **toCharArray()**, así:


```
private char[] palabra_secreta;

public Servidor(Socket socket, int idSession) {

    this.socket = socket;

    this.palabra_secreta = Random().toCharArray();

    try {
        dos = new DataOutputStream(socket.getOutputStream());
        dis = new DataInputStream(socket.getInputStream());
    } catch (IOException ex) {
        Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```



Ahora se imprime en consola la palabra a adivinar y se envía al cliente en forma de String con el objeto de tipo **DataOutputStream** creado usando su método **writeUTF**, así:

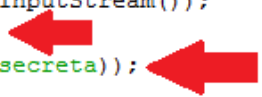
```
public Servidor(Socket socket, int idSession) {

    this.socket = socket;

    this.palabra_secreta = Random().toCharArray();

    try {
        dos = new DataOutputStream(socket.getOutputStream());
        dis = new DataInputStream(socket.getInputStream());
        System.out.println(palabra_secreta);
        dos.writeUTF(String.valueOf(palabra_secreta));


    } catch (IOException ex) {
        Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null, ex);
    }
}
```



Se implementa y sobrescribe el método **Run** para correr los hilos en la aplicación

```
@Override
public void run() {
}


private String Random() {
    int num = (int) (Math.random() * (diccionario.length));
    return diccionario[num];
}
```



Por último se extiende la clase servidor de la clase **Thread** para poder correr los hilos.

```
public class Servidor extends Thread{

    private Socket socket;
    private DataOutputStream dos;
    private DataInputStream dis;
```



2) Clase Main

Se importan las siguientes librerías:

```
package servidorahorcado;

import java.io.IOException;
import java.net.ServerSocket;
import java.net.Socket;
import java.util.logging.Level;
import java.util.logging.Logger;
```

Se creó el método **main** de la siguiente forma:

```

public class main {

    public static void main(String[] args) {
        ServerSocket ss;
        System.out.print("Iniciando servidor... ");
        try {
            ss = new ServerSocket(10578);
            System.out.println("\t[OK]");
            int idSession = 1;

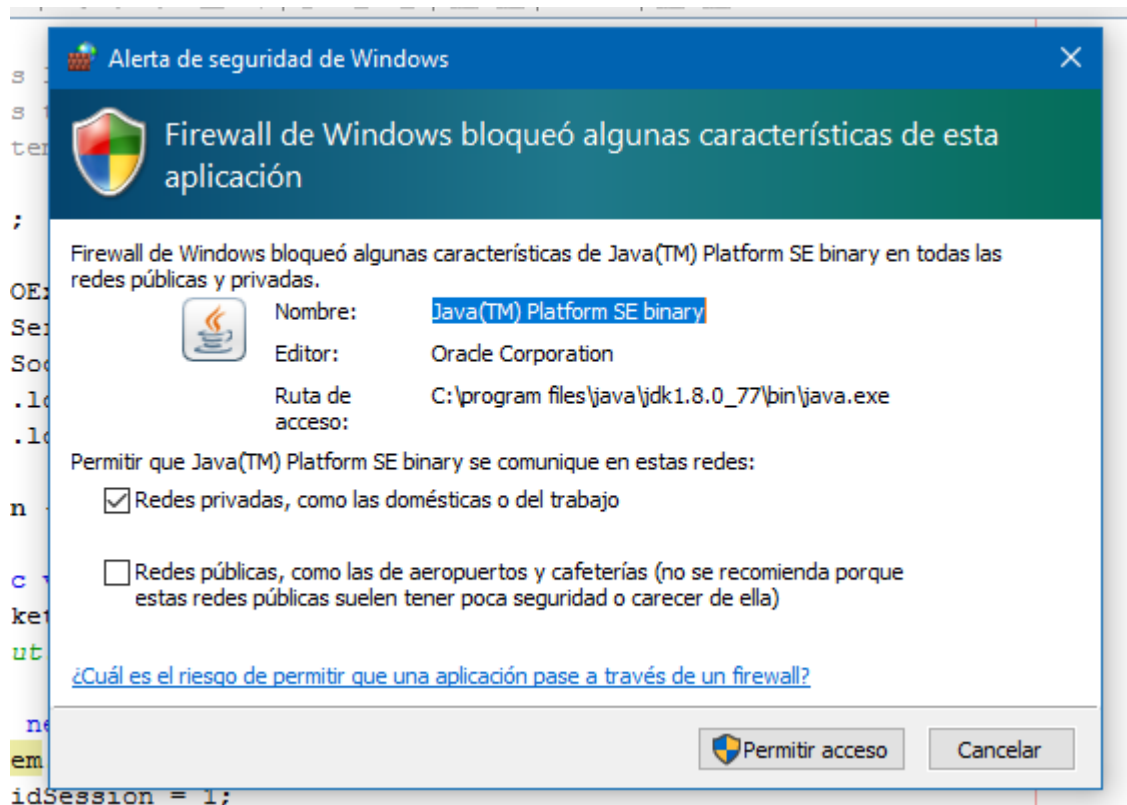
            while (true) {
                Socket socket;
                socket = ss.accept();
                System.out.println("Nueva conexión entrante: "+socket);
                ((Servidor) new Servidor(socket, idSession)).start();
                idSession++;
            }
        } catch (IOException ex) {
            Logger.getLogger(Servidor.class.getName()).log(Level.SEVERE, null, ex);
        }
    }
}

```

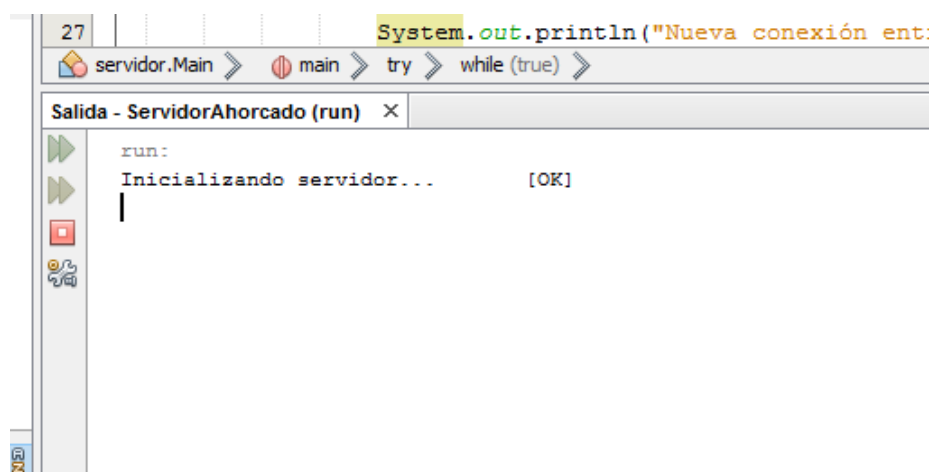
En este se realiza lo siguiente:

- Se crea un objeto de tipo **ServerSocket** llamado **ss**.
- Se imprime un mensaje en consola avisando que se está iniciando el servidor.
- Se crea un bloque **try – catch** para capturar las posibles excepciones.
- Dentro del bloque try – catch se instancia el objeto **ss** con un numero de puerto, en este caso el **10578**.
- Se imprime un mensaje para mostrar que el servidor se creó correctamente.
- Se crea un atributo llamado **idSesion** para identificar a los clientes, se le asigna el valor de **1**.
- Se crea un ciclo while que siempre se ejecute para escuchar las peticiones de los clientes.
- En el ciclo while se crea un objeto de tipo **Socket** llamado **socket**.
- Se instancia este objeto con la petición del cliente con el objeto **ss** a través del método **accept**.
- Se imprime un mensaje avisando que hay una nueva conexión y se muestra la información de este.
- Se crea un nuevo objeto de tipo servidor con el socket que se recibe de la petición y el id de sesión, luego se inicia con el método **start()**
- Se aumenta en 1 el id de sesión para el siguiente cliente.

Se corre el código y se comprueba que no tenga errores



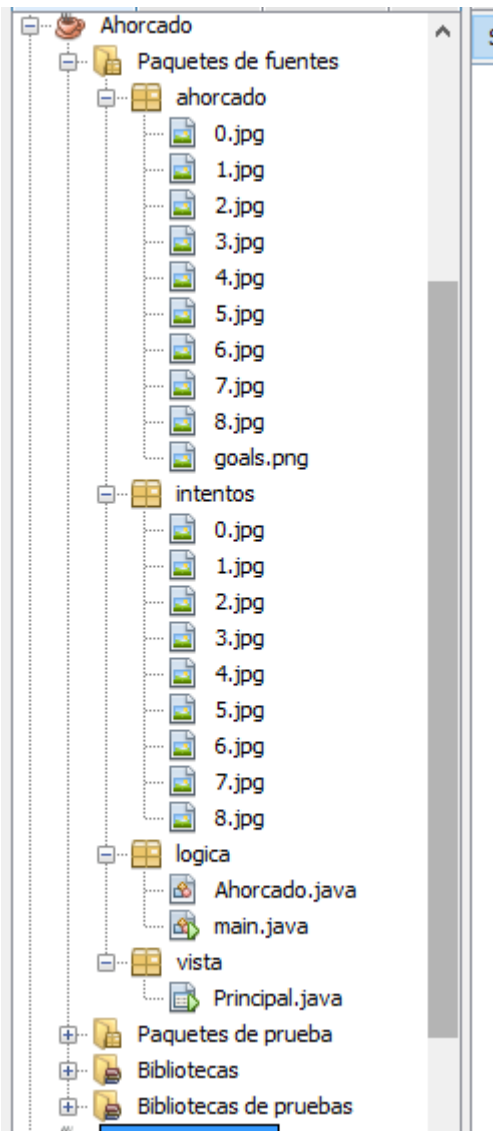
Aparece una ventana del Firewall de Windows, se permite el acceso.



Se corre el programa y se ve que el servidor se inició sin errores

Cliente Socket

Para el cliente se crea otro proyecto con la siguiente estructura



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:

```
package logica;

import java.io.DataInputStream;
import java.io.DataOutputStream;
import java.io.IOException;
import java.net.Socket;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;

public class Ahorcado {

    |
}
```

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

```
public class Ahorcado {

    protected Socket sk;
    protected DataOutputStream dos;
    protected DataInputStream dis;
    private int id;

    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 están relacionados con la comunicación entre el servidor y el cliente.

Los siguientes 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 **TextField campoPalabra**, **String palabra**, **JLabel errores**, **JLabel campoIntentos**, **JLabel largoPalabra**.

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

Estos elementos se crearon previamente en la vista del programa y son los que cambiarán 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 indicar que empezó el juego.

Se envía al JTextField 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 {
    sk = new Socket("localhost", 10578);
    dos = new DataOutputStream(sk.getOutputStream());
    dis = new DataInputStream(sk.getInputStream());
    System.out.println(id + " envía saludo");
    String palabra1 = "";
    palabra1 = dis.readUTF();
    System.out.println(palabra1);
    dis.close();
    dos.close();

    return palabra1;
}
```

En este método lo que se hace es crear el nuevo socket pasándole como parámetros la ip del servidor y el puerto, creamos las variables para poder enviar información entre el cliente y el servidor, y en la variable “*palabra1*” guardamos lo que nos envía el servidor que será la palabra que el servidor le asigno al cliente, por ultimo cerramos las variables y devolvemos 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 indicar 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 = true;
}

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 true 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.ImageIcon(getClass().getResource("/ganaste.png")));
    }
}
```

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 varios 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_Ocultas:** 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);  
    }  
}
```