

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 '4-10-2017'. Below the banner, several thin, curved lines in dark blue and light gray extend upwards from the bottom left corner.

4-10-2017

Juego Ahorcado

Implementado en JAVA con
Componentes HTTP

Jesús Andrés Acendra Martínez; Juan Diego Benítez
Arias

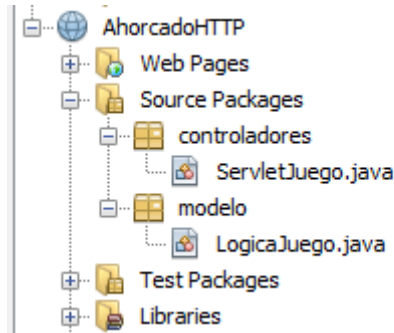
UNIVERSIDAD DE CARTAGENA

Contenido

Servidor WEB.....	2
1) Paquete Controladores.....	2
2) Paquete Modelo	3
Cliente HTTP	6
1) Lógica del juego	6
2) Método Constructor	8
3) Método para realizar la petición al servidor	10
4) Validar la letra ingresada por el cliente.....	11
5) Validar si gana el juego	12
6) Vista del juego	13

Servidor WEB

Lo primero es implementar el código del servidor web, se crea un proyecto **Java Web** con la siguiente estructura



1) Paquete Controladores

Este paquete contiene el único Servlet encargado de asignar la palabra con la que el cliente jugara.

ServletJuego.java:

Importamos las librerías necesarias para trabajar.

```
import com.google.gson.Gson;
import java.io.IOException;
import java.io.PrintWriter;
import java.io.Writer;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import modelo.LogicaJuego;
```

Se implementa el método por defecto que traen los servlet ***processRequest()***

```

L      */
      protected void processRequest(HttpServletRequest request, HttpServletResponse response)
      throws ServletException, IOException {
          response.setContentType("text/html;charset=UTF-8");
          PrintWriter out = response.getWriter();

          LogicaJuego palabra = new LogicaJuego();
          String word = palabra.Random();
          //System.out.println(word);
          String json = new Gson().toJson(word);
          System.out.println(json);
          response.setContentType("application/json");
          response.setCharacterEncoding("UTF-8");
          Writer writer = null;
          try{
              writer = response.getWriter();
              writer.write(json);
          }finally{
              try{
                  writer.close();

```

En este metodo instanciamos un objeto de la clase LogicaJuego y de este obtenemos la palabra por medio del metodo Random() y guardamos esta palabra en un variable llamada word. Utilizamos Gson() para trabajar con los objetos de Json, es decir, la variable word la convertimos a Json y esto lo guardamos en otro String pero ya como JSON. Configuramos los parametros de response para que pueda responder un JSON y por ultimo con ayuda de la clase Writer respondemos, enviando al String que contiene el JSON de la palabra como respuesta. Lo de mas son capturas de excepciones.

2) Paquete Modelo

En este paquete encontraremos las clases referente a la lógica del servidor.

LogicaJuego.java:

```

public class LogicaJuego {

    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",
        "MATEMATICAS", "CALCULO", "ALGEBRA", "DICCIONARIO", "BIBLIOTECA", "COCINA", "PELICULA", "COMERCIAL", "GATO",
        "SAPO", "JIRAFa", "FERROCARRIL", "FACEBOOK", "PERSONA", "BICICLETA", "CONTROL", "PANTALON", "AEROSOL",
        "ERROR", "COPA", "COPA", "PROGRAMADOR", "LICENCIA", "NUEVE", "PROCESADOR", "GARAJE", "MODERNO", "TABLA", "LUGAR",
        "LENGUAJE", "PROGRAMACION", "HERRAMIENTAS", "INTERNET", "EJECUTAR", "PROYECTO", "PERIODICO", "COCODRILO",
        "APLICACION", "PERA", "SOFTWARE", "ADMINISTRACION", "VENTANA", "MANTENIMIENTO", "INFORMACION", "PRESIDENTE",
        "NARANJA", "PRUEBA", "MANZANA", "JARRA", "CELULAR", "TELEFONO", "CONTAMINACION", "COLOR", "ROMANO", "ADIVINADOR",
        "INSTRUCCION", "CUADERNO", "CASA", "PALA", "ARBOL", "PUENTE", "PAPEL", "HOJA", "HELICOPTERO", "BARCO", "GENTE",
        "TUBERIA", "PLOMERO", "FUTBOL", "BALONCESTO", "ESTADIO", "JEAN", "FUENTE", "LEOPARDO", "REGLA", "PRIMERO",
        "BLUSA", "CAMISA", "AGUA", "FUEGO", "INDUSTRIA", "AIRE", "TIERRA", "NATURALEZA", "MIERCOLES", "FOTOGRAFIA",
        "TIGRE"};

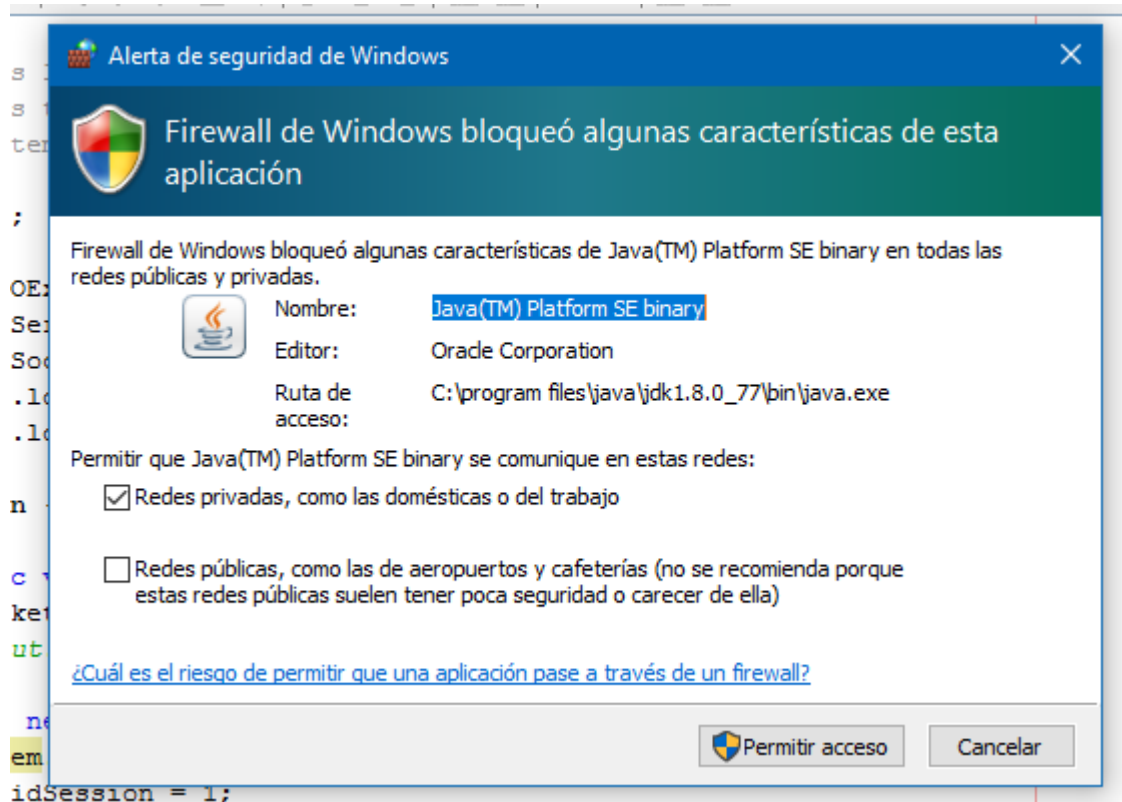
    public String Random(){
        int num = (int) (Math.random() * (diccionario.length));
        return diccionario[num];
    }
}

```

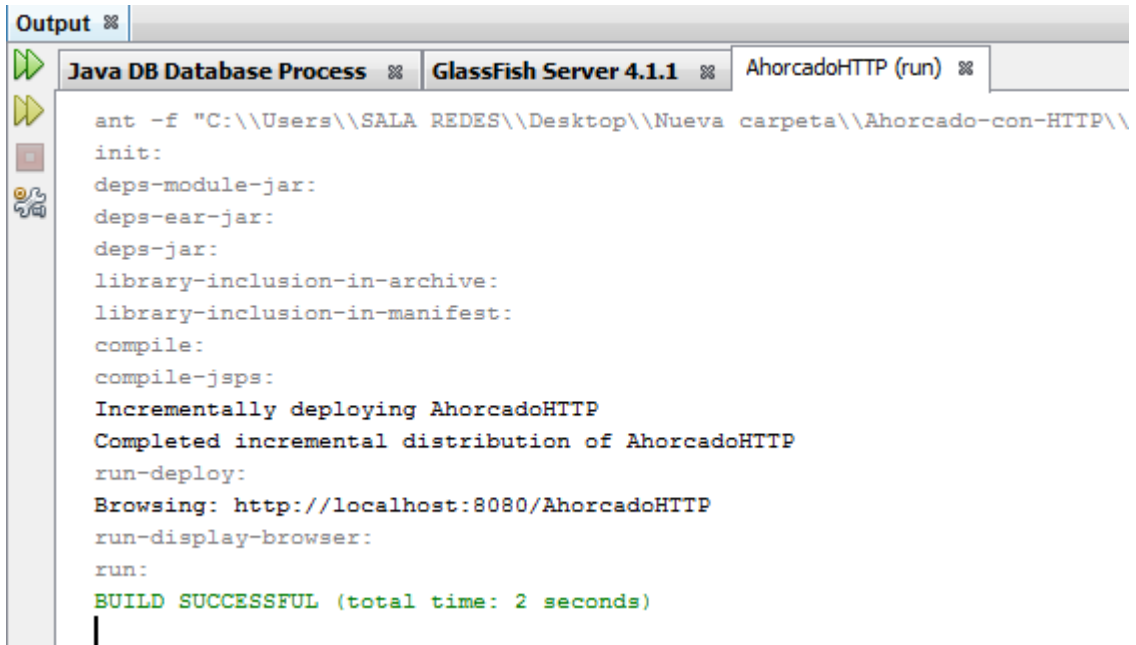
Se crea un atributo de tipo **vector String** con el nombre de **diccionario** en el cual se tendrán todas las palabras que tendrá el juego.

Y se implementa el método **Random()** el cual nos devolverá una palabra aleatoriamente de todas las contenidas en el vector.

Se corre el código y se comprueba que no tenga errores



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



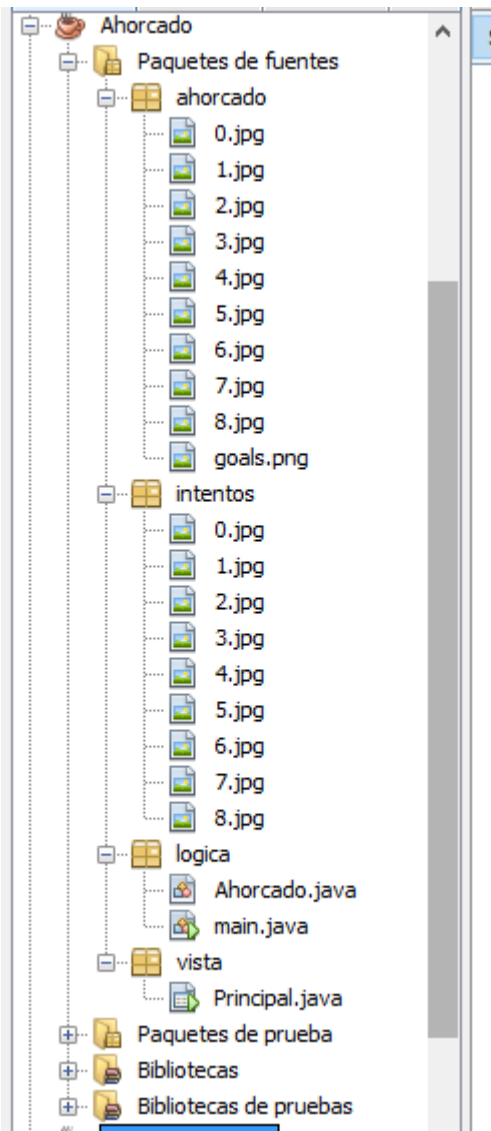
The screenshot shows an IDE's Output window with three tabs: "Java DB Database Process", "GlassFish Server 4.1.1", and "AhorcadoHTTP (run)". The "AhorcadoHTTP (run)" tab is active, displaying the output of an Ant build. The output text is as follows:

```
ant -f "C:\\Users\\SALA REDES\\Desktop\\Nueva carpeta\\Ahorcado-con-HTTP\\  
init:  
deps-module-jar:  
deps-ear-jar:  
deps-jar:  
library-inclusion-in-archive:  
library-inclusion-in-manifest:  
compile:  
compile-jsp:  
Incrementally deploying AhorcadoHTTP  
Completed incremental distribution of AhorcadoHTTP  
run-deploy:  
Browsing: http://localhost:8080/AhorcadoHTTP  
run-display-browser:  
run:  
BUILD SUCCESSFUL (total time: 2 seconds)
```

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

Cliente HTTP

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:

```
import com.google.gson.Gson;
import java.io.IOException;
import java.io.InputStream;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JTextField;
import org.apache.http.HttpEntity;
import org.apache.http.client.methods.CloseableHttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClients;
import org.apache.http.util.EntityUtils;
```

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

```

L  */
    public class Ahorcado {

        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 recuperarPalabra() throws IOException {
    String IP = "http://localhost:8080/AhorcadoHTTP/juego";
    CloseableHttpClient httpClient = HttpClients.createDefault();
    HttpGet httpGet = new HttpGet(IP);
    String palabra1 = "";
    CloseableHttpResponse response1 = httpClient.execute(httpGet);
    try {
        System.out.println(response1.getStatusLine());
        HttpEntity entity = response1.getEntity();
        if (entity != null) {
            String json = EntityUtils.toString(entity);
            System.out.println(json);
            palabra1 = new Gson().fromJson(json, String.class);
        }
        EntityUtils.consume(entity);
    } finally {
        response1.close();
    }

    System.out.println(palabra1);

    return palabra1;
}
```

En este método lo que se hace es crear una instancia de la clase **CloseableHttpClient** y un objeto de la clase **HttpGet** y le pasamos como parámetro la dirección donde se encuentra el servlet, además creamos un objeto de la clase **CloseableHttpResponse** el cual contendrá la respuesta que nos envía el servlet.

Dentro de la sentencia Try – Catch obtenemos lo que nos envía el servlet y se lo asignamos a un objeto de **HttpEntity** para trabajar con este. Por último obtenemos este objeto de **String** que contiene la palabra y de **JSON** lo pasamos a **String** con ayuda de **Gson()**, finalmente se retorna este **String**.

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 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 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.ImageIcon(getClass().getResource("/img/ahorcado_ganador.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) **TextField largoPalabra:** En este se le muestra al cliente cuantas letras tiene la palabra a adivinar.
- 2) **Label errores:** En este se muestran el número de errores se han cometido.
- 3) **TextField palabra_Oculto:** En este se muestran las letras que ha adivinado el cliente.
- 4) **Panel con Botones:** Un botón para cada letra del alfabeto, cuando se presionan se deshabilitan porque solo se puede usar cada letra una vez.
- 5) **Label 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);  
  
    }  
  
}
```