PROGRAMACIÓN

MANUAL TÉCNICO

Alumno: Carlos León Vázquez

FECHA DE ENTREGA: 04/06/2021

INDICE

Introducción

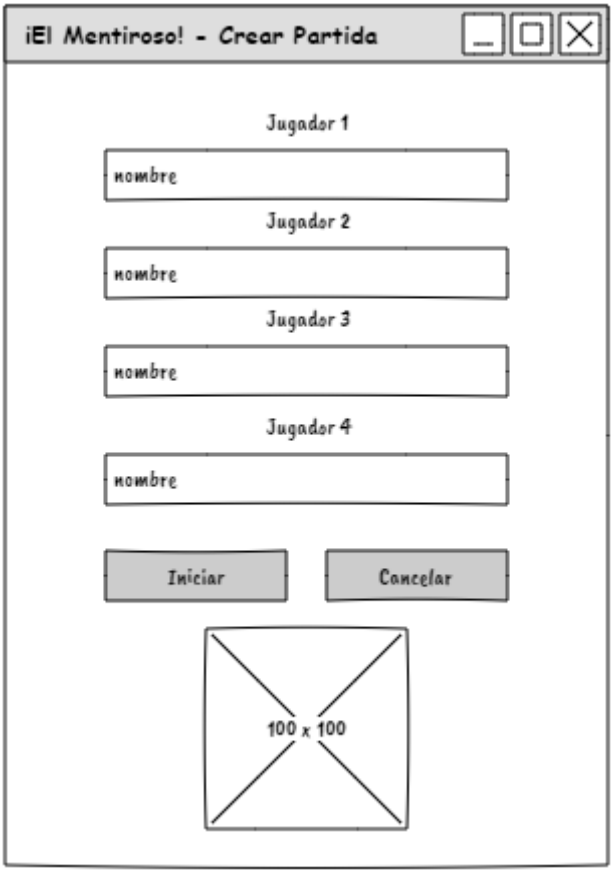
En la presente práctica se adjuntarán todas las capturas respectivas y necesarias para exponer de forma técnica el juego de cartas que he desarrollado, “El Mentiroso”. Las tecnologías que he empleado para su desarrollo han sido:

- DIA
- Pencil
- SQL Workbench
- IDE Eclipse
- Java

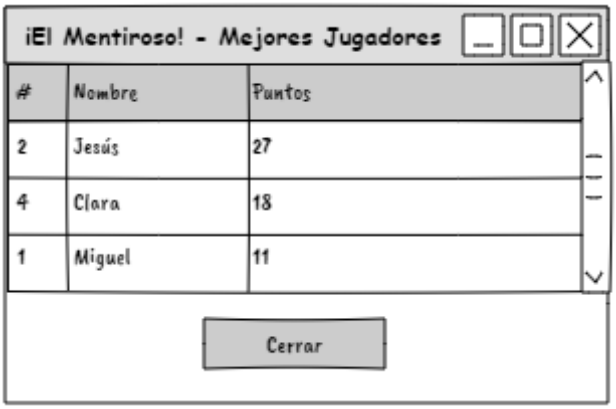
1. Capturas de pantalla – Pencil:

- ***Pantalla de inicio del juego:***

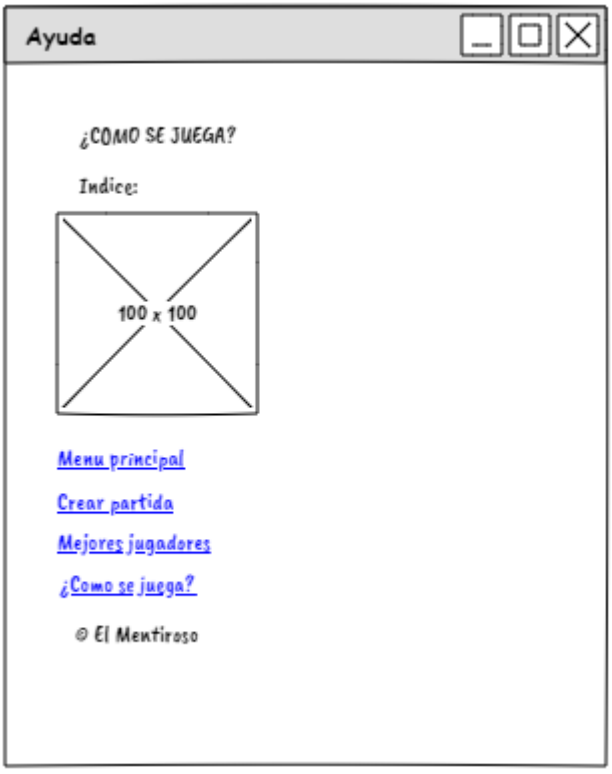
- ***Pantalla de Crear partida:***



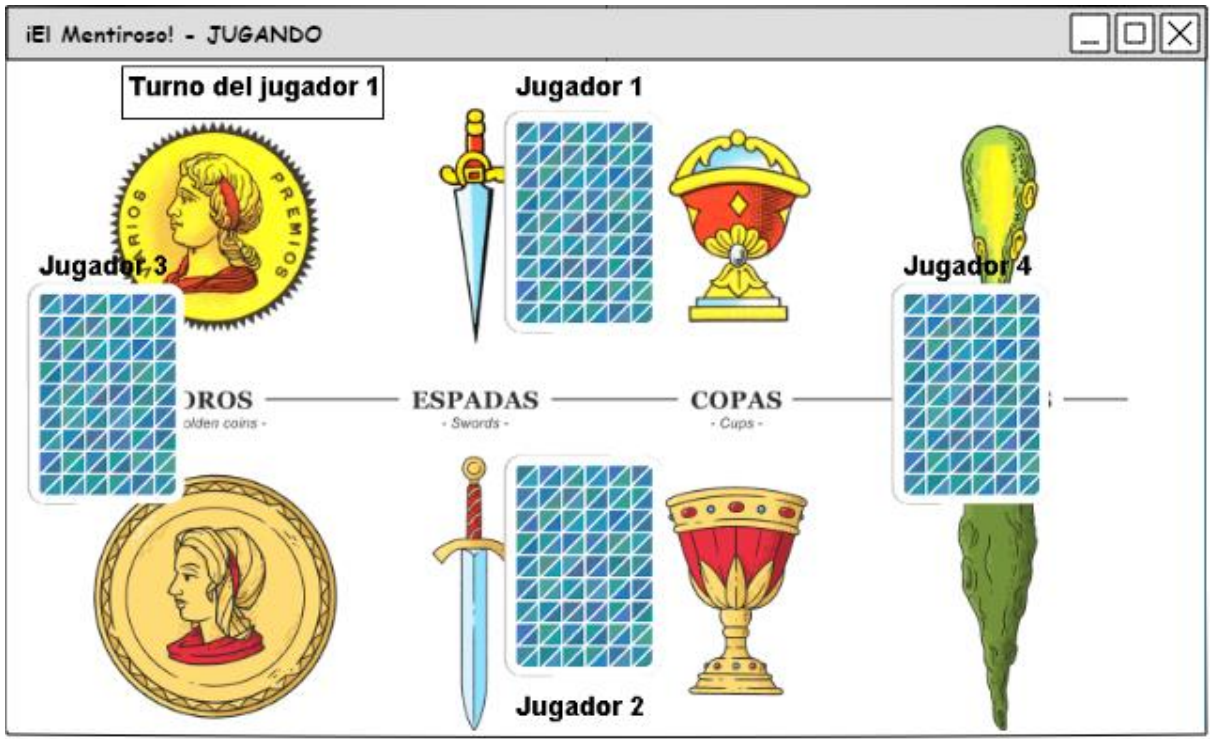
- ***Pantalla de Mejores Jugadores:***



- ***Pantalla de cómo se juega:***

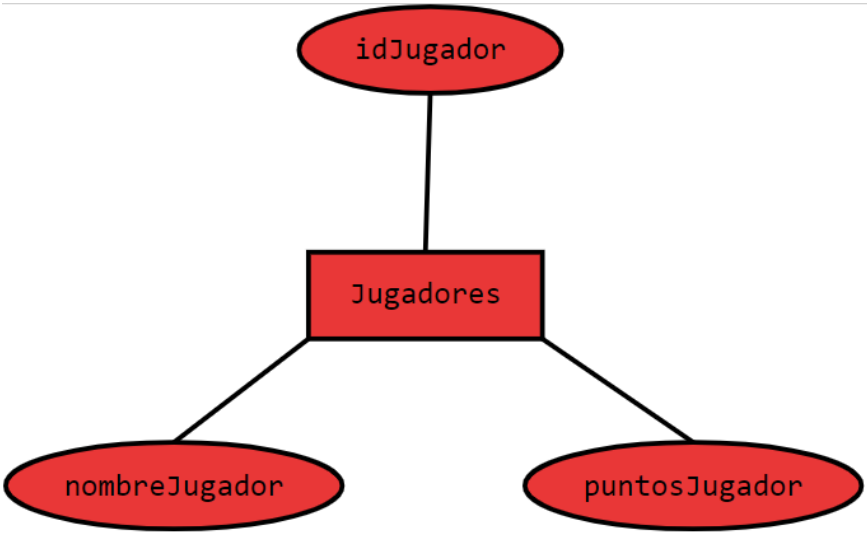


- ***Pantalla del juego durante la partida:***



2. Capturas de pantalla - ERD, ER, WORKBENCH, SQL

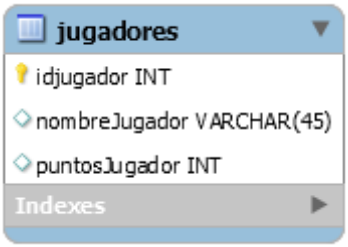
- **ERD - Diagrama Entidad Relación:**



- **ER - Entidad Relación:**

JUGADORES (#idJugadores, nombreJugador, puntosJugador)

- **Workbench:**



- **SQL, creación de base de datos y tablas.**

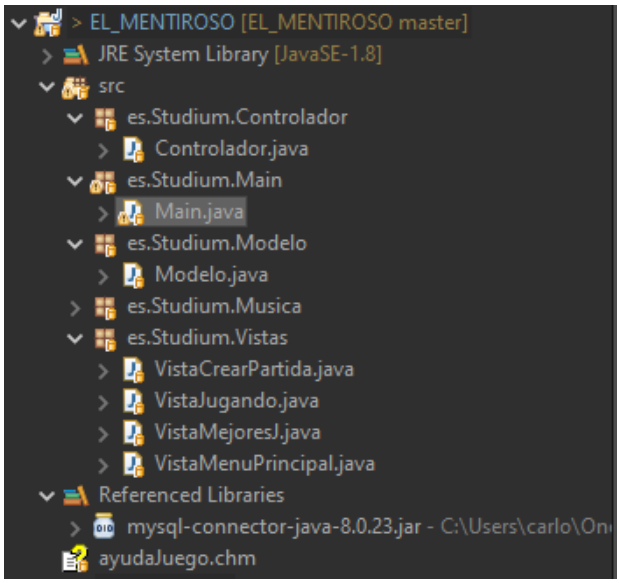
```
1    CREATE DATABASE el_mentiroso CHARACTER SET utf8mb4 COLLATE utf8mb4_spanish2_ci;
2 •  CREATE TABLE jugadores (idJugador INT AUTO_INCREMENT, PRIMARY KEY(idJugador), nombreJugador VARCHAR(25), puntosJugador INT);
3 •  SHOW TABLES;
```

Result Grid | Filter Rows: | Export: | Wrap Cell Content:

Tables_in_el_mentiroso
jugadores

3. Código del juego

➢ PAQUETES Y CLASES



➢ MENÚ PRINCIPAL

```java
package es.Studium.Main;

import es.Studium.Controlador.Controlador;
import es.Studium.Modelo.Modelo;
import es.Studium.Musica.Sonido;
import es.Studium.Vistas.VistaCrearPartida;
import es.Studium.Vistas.VistaJugando;
import es.Studium.Vistas.VistaMejoresJ;
import es.Studium.Vistas.VistaMenuPrincipal;

public class Main
{

    public static void main (String[]args)
    {
        Modelo modelo = new Modelo();
        VistaMejoresJ vistaMejoresJ = new VistaMejoresJ();
        VistaMenuPrincipal vistaMenu = new VistaMenuPrincipal();
        VistaCrearPartida vistaCrearP = new VistaCrearPartida();
        VistaJugando vistaJugando = new VistaJugando();
        new Controlador (vistaMejoresJ, vistaMenu,vistaCrearP, vistaJugando, modelo);
        Sonido sonido = new Sonido();
    }

}
```

- VISTACREARPARTIDA

```
public class VistaCrearPartida extends Frame
{
    private static final long serialVersionUID = 1L;
    public Frame ventanaCrearPartida = new Frame ("¡El Mentiroso!: Crear Partida");
    public Label labelNombreJugador1 = new Label ("Nombre Jugador 1:");
    public TextField textoNombreJugador1 = new TextField (30);
    public Label labelNombreJugador2 = new Label ("Nombre Jugador 2:");
    public TextField textoNombreJugador2 = new TextField (30);
    public Label labelNombreJugador3 = new Label ("Nombre Jugador 3:");
    public TextField textoNombreJugador3 = new TextField (30);
    public Label labelNombreJugador4 = new Label ("Nombre Jugador 4:");
    public TextField textoNombreJugador4 = new TextField (30);
    public Label labelCodigoCreadordePartida = new Label ("Introduce tu codigo:");
    public TextField textoCodigoJugadorPartida = new TextField (30);
    public Dialog dialogoMensajePartidaCreada = new Dialog(ventanaCrearPartida, "Partida creada", true);
    public Label partidaCreada = new Label ("ERROR: Faltan datos");
    public Button buttonIniciarPartida = new Button ("Iniciar");
    public Button cerrarPartida = new Button ("Cerrar");
    Toolkit herramienta;
    Image fondoCrearPartida;

    public VistaCrearPartida()
    {
        setTitle("¡El Mentiroso!: Crear Jugador");
        setLayout(new FlowLayout());
        add(labelNombreJugador1);
        add(textoNombreJugador1);
        add(labelNombreJugador2);
        add(textoNombreJugador2);
        add(labelNombreJugador3);
        add(textoNombreJugador3);
        add(labelNombreJugador4);
        add(textoNombreJugador4);
        add(buttonIniciarPartida);
        add(cerrarPartida);
        herramienta = getToolkit();
        fondoCrearPartida = herramienta.getImage("fondoCrearPartida.jpg");
        setVisible(false);
        setSize(300,455);
        setLocationRelativeTo(null);
        setResizable(false);

        dialogoMensajePartidaCreada.setLayout(new FlowLayout());
        dialogoMensajePartidaCreada.setSize(200,100);
        dialogoMensajePartidaCreada.setLocationRelativeTo(null);
        dialogoMensajePartidaCreada.setResizable(false);
        dialogoMensajePartidaCreada.add(partidaCreada);
    }
    //DIBUJAMOS
    public void paint(Graphics g)
    {
        g.drawImage(fondoCrearPartida,0,295,this);
    }
```

- VISTAMENUPRINCIPAL

```
import java.awt.Button;

public class VistaMenuPrincipal extends Frame
{
    private static final long serialVersionUID = 1L;

    //VENTANA MENU
    public Frame ventanaMenu = new Frame ("¡El Mentiroso!");
    public Label labelMenu = new Label("¡BIENVENIDO AL MENTIROSO!");
    public Button buttonCrearPartida = new Button ("Nueva Partida");
    public Button buttonMejoresJugadores = new Button ("Mejores jugadores");
    public Button buttoncomoSeJuega = new Button ("¿Como se juega?");
    public Button buttonSalirMenu = new Button ("Salir");

    //HERRAMIENTA PARA IMAGENES
    Toolkit herramienta;
    Image fondoMenu;

    public VistaMenuPrincipal()
    {
        setTitle("¡El Mentiroso");
        setLayout(new FlowLayout());
        add(labelMenu);
        add(buttonCrearPartida);
        add(buttonMejoresJugadores);
        add(buttoncomoSeJuega);
        add(buttonSalirMenu);
        herramienta = getToolkit();
        fondoMenu = herramienta.getImage("fondoMenu.jpg");
        setVisible(true);
        setSize(295,365);
        setResizable(false);
        setLocationRelativeTo(null);
    }

    //DIBUJAMOS
    public void paint(Graphics g)
    {
        g.drawImage(fondoMenu,30,140,this);
    }

}
```

- VISTAMEJORESJUGADORES

```
package es.Studium.Vistas;

import java.awt.Button;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextArea;



public class VistaMejoresJ  extends Frame
{
    private static final long serialVersionUID = 1L;

    //VENTANA MEJORES JUGADORES
    public Frame ventanaMejoresJugadores = new Frame("Mejores Mentirosos");
    public Label labelMejores = new Label ("El Mentiroso: ¡Mejores Mentirosos!");
    public TextArea listadoJugadores = new TextArea(15, 40);
    public Button cerrar = new Button("cerrar");

    //CONSTRUCTOR
    public VistaMejoresJ()
    {
        ventanaMejoresJugadores.setLayout(new FlowLayout());
        ventanaMejoresJugadores.add(labelMejores);
        ventanaMejoresJugadores.add(listadoJugadores);
        listadoJugadores.append("ID\tJUGADOR\tPUNTOS\n");
        ventanaMejoresJugadores.add(cerrar);
        ventanaMejoresJugadores.setSize(600,420);
        ventanaMejoresJugadores.setBackground(Color.white);
        ventanaMejoresJugadores.setLocationRelativeTo(null);
        ventanaMejoresJugadores.setResizable(false);
    }

}
```

- VISTAJUGANDO

```
public class VistaJugando extends Frame
{
    private static final long serialVersionUID = 1L;
    Toolkit herramientas;
    Image tapete, reverso;
    int cartaReverso = 0;
    int cargarCartas=0;
    int imagenAmostrar2 = 0;
    int imagenAmostrar3 = 0;
    int imagenAmostrar4 = 0;
    int imagenAmostrar5 = 0;
    int turno = 0;

    public Frame ventanaJuego = new Frame ("¡EL MENTIROSO!: JUGANDO");
    public Dialog dialogoTurno = new Dialog(this, "TURNO", true);
    public Dialog dialogoAcusacion1 = new Dialog(this,"ACUSACIÓN");
    public Dialog dialogoAcusacion2 = new Dialog(this,"ACUSACIÓN");
    public Dialog dialogoVictoria = new Dialog(this,"GANADOR");
    public Dialog dialogoComienzo = new Dialog(this,"INICIO");
    public Label lblTurno = new Label();
    public Label lblComienzo = new Label();
    public Label lblVictoria = new Label();
    public Label lblNoMintio = new Label();
    public Label lblMentido = new Label();
```

```
    public VistaJugando()
    {
        herramientas = getToolkit();
        tapete = herramientas.getImage("tapete2.png");
        reverso = herramientas.getImage("reverso.png");

        this.setTitle("¡EL MENTIROSO! - JUGANDO");
        this.setSize(820,490);
        this.setLocationRelativeTo(null);
        this.setResizable(false);
        this.setVisible(false);

        dialogoComienzo.setLayout(new FlowLayout());
        dialogoComienzo.setSize(200,100);
        dialogoComienzo.setLocationRelativeTo(null);
        dialogoComienzo.setResizable(false);
        dialogoComienzo.add(lblComienzo);

        dialogoComienzo.setLayout(new FlowLayout());
        dialogoComienzo.setSize(300,100);
        dialogoComienzo.setLocationRelativeTo(null);
        dialogoComienzo.setResizable(false);
        dialogoComienzo.add(lblTurno);

        dialogoVictoria.setLayout(new FlowLayout());
        dialogoVictoria.setSize(200,100);
        dialogoVictoria.setLocationRelativeTo(null);
        dialogoVictoria.setResizable(false);
        dialogoVictoria.add(lblVictoria);

        dialogoAcusacion1.setLayout(new FlowLayout());
        dialogoAcusacion1.setSize(200,100);
        dialogoAcusacion1.setLocationRelativeTo(null);
        dialogoAcusacion1.setResizable(false);
        dialogoAcusacion1.add(lblNoMintio);

        dialogoAcusacion2.setLayout(new FlowLayout());
        dialogoAcusacion2.setSize(200,100);
        dialogoAcusacion2.setLocationRelativeTo(null);
        dialogoAcusacion2.setResizable(false);
        dialogoAcusacion2.add(lblMentido);
    }
```

```
publ Close d paint(Graphics g)
{
    g.drawImage(tapete, 0, 30, this);
    Font fuente = new Font("Arial", Font.BOLD, 18);
    g.setFont(fuente);
    g.setColor(Color.black);
    g.drawString("Jugador 1", 350, 55);
    g.drawImage(reverso, 340, 65, this);
    g.drawString("Jugador 2", 350, 470);
    g.drawImage(reverso, 340, 295, this);
    g.drawString("Jugador 3", 30, 175);
    g.drawImage(reverso, 20, 180, this);
    g.drawString("Jugador 4", 610, 175);
    g.drawImage(reverso, 600, 180, this);
    g.setColor(Color.black);
    g.drawRect(85, 35, 175, 35);
    g.drawString("Turno del jugador " + turno, 90, 55);

    //POSICION DE LAS CARTAS
    switch(cartaReverso)
    {
    case 1:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 2:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 3:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 4:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 5:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 6:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 7:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 8:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 9:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 10:
        g.drawImage(reverso, 340, 165, this);
        break;
    case 11:
        g.drawImage(reverso, 340, 165, this);
```

```
            case 48:
                g.drawImage(reverso, 340, 165, this);
                break;
            }
        }

        //DIBUJAMOS
        public void reversoCarta()
        {
            cartaReverso++;
            repaint();
        }

        //DIBUJAMOS TURNOS
        public void avisoTurno(int t)
        {
            this.turno=t;
            repaint();
        }

}
```

- MODELO

```
public class Modelo
{
    Random rnd = new Random();

    //BARAJAMOS LA CARTA DE LOS JUGADORES Y LOS METEMOS EN SUS ARRAYLIST CORRESPONDIENTES
    public void barajar(ArrayList<Integer> uno,ArrayList<Integer>dos, ArrayList<Integer> tres, ArrayList<Integer> cuatro)
    {
        int jugador = 0;
        int numeroArepartir = 1;
        int contador1=0, contador2=0, contador3=0, contador4=0;
        //REPARTIMOS LAS CARTAS PARA CADA JUGADOR
        for(int i = 0; i < 48; i++)
        {
            jugador = rnd.nextInt(4);
            if((jugador==0)&&(contador1<12))
            {
                uno.add(numeroArepartir);
                contador1++;
            }
            else if (contador2<12)
            {
                dos.add(numeroArepartir);
                contador2++;
            }

            else if (contador3<12)
            {
                tres.add(numeroArepartir);
                contador3++;
            }

            else if (contador4<12)
            {
                cuatro.add(numeroArepartir);
                contador4++;
            }
            if(numeroArepartir%12==0)
            {
                numeroArepartir=1;
            }
            else
            {
                numeroArepartir++;
            }
        }
    }
```

```
//LANZAMIENTO DE CARTAS Y ELIMINACIÓN DE LAS CARTAS LANZADAS DEL MAZO DEL JUGADOR
public ArrayList<Integer>lanzamientos(ArrayList<Integer>Mazo ,ArrayList<Integer>CartasLanzadas)
{
    for(int i=0; i<CartasLanzadas.size();i++)
    {
        //CON EL BOOLEAN CONTROLAMOS QUE NO SE ELIMINEN NUMEROS DUPLICADOS DEL MAZO POR ERROR
        /*
         * Es decir, si un jugador lanza un 2 y un 4, y en su mazo tiene tres 4, no se elimen
         * los tres, sino solo el que ha lanzado, es por ello que recorre el mazo y las cartas que
         * se han lanzado.
         */
        boolean noRepetir = false;
        for(int j=0;j<Mazo.size();j++)
        {

            if(CartasLanzadas.get(i)==Mazo.get(j) && (noRepetir==false))
            {
                Mazo.remove(j);
                noRepetir=true;
            }
        }
    }
    return Mazo;
}
```

```java
public ArrayList<Integer>accionJugador(ArrayList<Integer>mazoJugador)
{
    Random cantidadCarta = new Random();
    ArrayList<Integer>cartasLanzadas = new ArrayList<Integer>();
    Integer cantidadCartaUsada = cantidadCarta.nextInt(5);
    ArrayList<Integer>cartasLanzadasPosicion = new ArrayList<Integer>();

    //SI EL JUGADOR TIENE MENOS DE CINCO CARTAS
    if(mazoJugador.size()<5)
    {
        //QUE EMPIECE A LANZAR DE UNA EN UNA, EVITAR ERRORES
        cantidadCartaUsada = 1;

    }
    else
    {   //MIENTRAS CANTIDAD USADA SEA IGUAL A 0
        while(cantidadCartaUsada==0)
        {   //SE SIGUE LANZANDO CON NORMALIDAD
            cantidadCartaUsada = cantidadCarta.nextInt(5);
        }
    }

    //RECORRE EL ARRAY PARA ESCOGER LAS CARTAS
    for(int i=0; i<cantidadCartaUsada;i++)
    {
        //CONTROLA QUE NO SE REPTITA LA POSICION DE UNA CARTA ELEGIDA
        boolean controlRepiticion = false;
        while(controlRepiticion==false)
        {
            Integer posicionCartaElegida = 0;
            if (cantidadCartaUsada==1)
            {
                posicionCartaElegida=0;
            }
            else
            {
                posicionCartaElegida= cantidadCarta.nextInt(mazoJugador.size()-1);
            }

            if(cartasLanzadasPosicion.size()==0)
            {
                controlRepiticion=true;
                cartasLanzadasPosicion.add(posicionCartaElegida);
                cartasLanzadas.add(mazoJugador.get(posicionCartaElegida));
            }
```

```
            //CUANDO YA EXISTE UNA POSICION DENTRO DEL ARRAYLIST
            else
            {
                boolean controlRepiticion2=false;
                for(int j=0; j<cartasLanzadasPosicion.size();j++)
                {
                    if(cartasLanzadasPosicion.get(j)==posicionCartaElegida)
                    {
                        controlRepiticion2=true;
                    }
                }
                //SI NO COINCIDE LA POSICION
                if(controlRepiticion2==false)
                {
                    controlRepiticion=true;
                    cartasLanzadasPosicion.add(posicionCartaElegida);
                    cartasLanzadas.add(mazoJugador.get(posicionCartaElegida));
                }
            }

        }
    }
    return cartasLanzadas;
}
```

```
//METODO POR EL CUAL AÑADIMOS LAS CARTAS LANZADAS AL MAZO CENTRAL
public ArrayList<Integer>mazoCartaCentral(ArrayList<Integer>mazoCentral, ArrayList<Integer>CartasLanzadas)
{
    for(int i=0; i<CartasLanzadas.size();i++)
    {
        mazoCentral.add(CartasLanzadas.get(i));
    }

    return mazoCentral;
}

//METODO POR EL CUAL DEVOLVEMOS LA CARTA DEL MAZO CENTRAL AL JUGADOR
public ArrayList<Integer>devolverCartasAJugadores(ArrayList<Integer>mazoJugador, ArrayList<Integer>mazoCentral)
{
    for(int i=0; i<mazoCentral.size();i++)
    {
        mazoJugador.add(mazoCentral.get(i));
    }
    return mazoJugador;

}

//ESCOGEMOS UN NUMERO INICIAL UNA VEZ INICIADA LA PARTIDA, PARA QUE SEA ESE NUMERO EL QUE TENGA QUE LANZAR LOS JUGADORES
public int numeroInicialSeleccionado(int numero[])
{
    Math.random();
    int numeroAleatorio = (int) (Math.random()*12+1);
    return numeroAleatorio;
}

//SI UN JUGADOR LEVANTA A OTRO JUGADOR, SE DEBERA ESCOGER OTRO NUMERO A LANZAR
public int nuevoNumeroSeleccionado(int numero[])
{
    Math.random();
    int numeroAleatorio = (int) (Math.random()*12+1);
    return numeroAleatorio;
}
```

```java
public Connection conectar()
{
    Connection c = null;
    String driver = "com.mysql.cj.jdbc.Driver";
    String url = "jdbc:mysql://localhost:3306/el_mentiroso?serverTimezone=UTC";//ESPECIFICAMOS LA URL
    String Login = "root";
    String password = "Studium2020;";
    try
    {
        //Cargar los controladores para el acceso a la BD
        Class.forName(driver);
        //Establecer la conexión con la BD clientes
        c = DriverManager.getConnection(url, Login, password);
    }
    catch (ClassNotFoundException cnfe)
    {
        System.out.println("Error 1-"+cnfe.getMessage());
    }
    catch (SQLException sqle)
    {
        System.out.println("Error 2-"+sqle.getMessage());
    }
    return (c);
}

//METODO CERRAR LA CONEXIÓN DE LA BASE DE DATOS
public void cerrar(Connection conexion)
{
    try
    {
        if(conexion!=null)
        {
            conexion.close();
        }
    }
    catch (SQLException error)
    {
        System.out.println("Error 3-" +error.getMessage());
    }

}
```

```java
//METODO QUE ME MUESTRA CUALES SON LOS MEJORES JUGADORES SEGUN SUS PUNTOS
public String mejoresJugadores(Connection conexion)
{
    String datos ="";
    Statement statement = null;
    ResultSet rs = null;
    String sentencia = "SELECT idJugador, nombreJugador, puntosJugador FROM jugadores ORDER BY puntosJugador DESC LIMIT 10;";

    try
    {
        statement = conexion.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
        rs = statement.executeQuery(sentencia);
        while(rs.next())
        {
            datos = datos + rs.getInt("idJugador") + "\t" ;
            datos = datos + rs.getString("nombreJugador")+ "\t"+ "\t";
            datos = datos + rs.getInt("puntosJugador") + "\n";
        }
    }
    catch (SQLException e)
    {
        e.printStackTrace();
    }
    return(datos);
}

//METODO PARA CREAR UNA PARTIDA Y QUE A SU VEZ SE LE ASIGNE ESA PARTIDA AL JUGADOR QUE LA HA CREADO
public void crearPartidaNueva(Connection conexion,String nombre, int puntuacionJugador)
{

    Statement statement = null;
    String sentencia="INSERT INTO jugadores VALUES (null, '" + nombre + "','" + puntuacionJugador + "')";
    try
    {
        //CREAMOS LA SENTENCIA
        statement = conexion.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                ResultSet.CONCUR_READ_ONLY);
        statement.executeUpdate(sentencia);
        System.out.println(sentencia);
    }
    catch (SQLException e)
    {
        System.out.println("Error al introducir datos."+e.getMessage());
    }

}
```

```java
//METODO POR EL CUAL SE ABRE EL "MENU" AYUDA
public void ayuda()
{
    try
    {
        //EJECUTA EL ARCHIVO DE AYUDA
        Runtime.getRuntime().exec("hh.exe ayudaJuego.chm");
    }
    catch (IOException e)
    {
        e.printStackTrace();
    }
}
```

- CONTROLADOR

```
VistaMenuPrincipal vistaMenu;
VistaCrearPartida vistaCrearP;
VistaJugando vistaJugando;
VistaMejoresJ vistaMejoresJ;
Modelo modelo;
Connection conexion = null;
String informacion ="";

//MAZOS DE LOS JUGADORES
/*  HACEMOS UN ARRAYLIST PARA LOS MAZOS DE LOS JUGADORES, PUES INTERESAN ARRAYS DINAMICOS
 *  QUE PUEDAN IR CAMBIANDO SUS VALORES Y NO ARRAYS ESTATICOS
 */
ArrayList<Integer>mazoJugador1 = new ArrayList<Integer>();
ArrayList<Integer>mazoJugador2 = new ArrayList<Integer>();
ArrayList<Integer>mazoJugador3 = new ArrayList<Integer>();
ArrayList<Integer>mazoJugador4 = new ArrayList<Integer>();
ArrayList<Integer>mazoCentral = new ArrayList<Integer>();
ArrayList<Integer>cartasLanzadas = new ArrayList<Integer>();

int numeroInicial = 0;
int cartaActual =0;
int cartaActualJugador1 = 0;
int cartaActualJugador2 = 0;
int cartaActualJugador3 = 0;
int cartaActualJugador4 = 0;
int turno = 1;
int puntuacionJugador1=0;
int puntuacionJugador2=0;
int puntuacionJugador3=0;
int puntuacionJugador4=0;
int uno,dos,tres,cuatro;
boolean controlTurno =false;
/* BOOLEAN CONTROLTURNO =
 * CONTROLAMOS LOS TURNOS DE LOS JUGADORES EN FUNCION DE SI HAN MENTIDO O NO
 * SI ES FALSE, QUIERE DECIR QUE EL ADVERSARIO HA MENTIDO, POR LO TANTO EL JUGADOR
 * QUE LEVANTA NO PIERDE EL TURNO. SI ES TRUE, EL JUGADOR QUE LEVANTA PIERDE EL TURNO
 * PUES EL ADVERSARIO NO HA MENTIDO.
 */
```

```
//COLOCAMOS LOS PARAMETROS EN EL CONTROLADOR
public Controlador(VistaMejoresJ objvistaMej, VistaMenuPrincipal objvistaM, VistaCrearPartida objvistaP, VistaJugando objvistaJug, Modelo objmodelo)
{
    //HACEMOS LAS LLAMADAS CORRESPONDIENTES
    this.vistaMejoresJ = objvistaMej;
    this.vistaMenu = objvistaM;
    this.vistaCrearP = objvistaP;
    this.vistaJugando = objvistaJug;
    this.modelo = objmodelo;

    //AGREGAMOS LOS LISTENER
    this.vistaMejoresJ.addWindowListener(this);
    this.vistaCrearP.addWindowListener(this);
    this.vistaMenu.addWindowListener(this);
    this.vistaJugando.addWindowListener(this);
    this.vistaJugando.addMouseListener(this);

    //ELEMENTOS VISTA PARTIDA
    objvistaP.addWindowListener(this);
    objvistaP.ventanaCrearPartida.addWindowListener(this);
    objvistaP.dialogoMensajePartidaCreada.addWindowListener(this);
    objvistaP.buttonIniciarPartida.addActionListener(this);
    objvistaP.cerrarPartida.addActionListener(this);
    objvistaP.dialogoMensajePartidaCreada.addWindowListener(this);

    //ELEMENTOS VISTA MENU
    objvistaM.ventanaMenu.addWindowListener(this);
    objvistaM.buttonCrearPartida.addActionListener(this);
    objvistaM.buttoncomoSeJuega.addActionListener(this);
    objvistaM.buttonMejoresJugadores.addActionListener(this);
    objvistaM.buttonSalirMenu.addActionListener(this);

    //ELEMENTOS VISTA JUGANDO
    objvistaJug.ventanaJuego.addWindowListener(this);
    objvistaJug.ventanaJuego.addMouseListener(this);
    objvistaJug.dialogoTurno.addWindowListener(this);
    objvistaJug.dialogoComienzo.addWindowListener(this);
    objvistaJug.dialogoVictoria.addWindowListener(this);
    objvistaJug.dialogoAcusacion1.addWindowListener(this);
    objvistaJug.dialogoAcusacion2.addWindowListener(this);

    //ELEMENTOS VISTA MEJORES JUGADORES
    objvistaMej.ventanaMejoresJugadores.addWindowListener(this);
    objvistaMej.cerrar.addActionListener(this);
    objvistaMej.addWindowListener(this);
    objvistaMej.setLocationRelativeTo(null);

}
```

```
public void actionPerformed(ActionEvent evento)
{

    //CREAR PARTIDA
    if(vistaMenu.buttonCrearPartida.equals(evento.getSource()))
    {
        this.vistaCrearP.setVisible(true);
        new VistaCrearPartida();

    }
    //INICIO DE LA PARTIDA Y AGREGAR NOMBRE DE LOS JUGADORES A LA BASE DE DATOS
    if(vistaCrearP.buttonIniciarPartida.equals(evento.getSource()))
    {
        if(this.vistaCrearP.buttonIniciarPartida.equals(evento.getSource()))
        {
            String texto1=this.vistaCrearP.textoNombreJugador1.getText();
            String texto2=this.vistaCrearP.textoNombreJugador2.getText();
            String texto3=this.vistaCrearP.textoNombreJugador3.getText();
            String texto4=this.vistaCrearP.textoNombreJugador4.getText();
            texto1=texto1.replaceAll(" ", "");
            texto2=texto2.replaceAll(" ", "");
            texto3=texto3.replaceAll(" ", "");
            texto4=texto4.replaceAll(" ", "");
            if(texto2.length()==0 && texto2.length()==0 && (texto3.length()==0) && (texto4.length()==0))
            {
                this.vistaCrearP.partidaCreada.setText("ERROR: FALTAN DATOS.");
                this.vistaCrearP.dialogoMensajePartidaCreada.setVisible(true);
            }
            else if(texto2.length()==0)
            {
                this.vistaCrearP.partidaCreada.setText("ERROR: FALTAN DATOS.");
                this.vistaCrearP.dialogoMensajePartidaCreada.setVisible(true);
            }
            else if(texto3.length()==0)
            {
                this.vistaCrearP.partidaCreada.setText("ERROR: FALTAN DATOS.");
                this.vistaCrearP.dialogoMensajePartidaCreada.setVisible(true);
            }
            else if(texto4.length()==0)
            {
                this.vistaCrearP.partidaCreada.setText("ERROR: FALTAN DATOS.");
                this.vistaCrearP.dialogoMensajePartidaCreada.setVisible(true);
            }
            this.vistaJugando.setVisible(false);
        }
```

```
    if(this.vistaCrearP.buttonIniciarPartida.equals(evento.getSource()))
    {
        new VistaJugando();
        this.vistaCrearP.setVisible(false);
        this.vistaJugando.setVisible(true);
        //BARAJAMOS LAS CARTAS CON EL METODO DEL MODELO
        this.modelo.barajar(mazoJugador1, mazoJugador2, mazoJugador3, mazoJugador4);
        this.vistaJugando.lblComienzo.setText("CARTAS REPARTIDAS, NUMERO A LANZAR:" + this.modelo.numeroInicialSeleccionado(null) );
        this.vistaJugando.avisoTurno(+1);
        this.vistaJugando.dialogoComienzo.setVisible(true);
        //DEL 0 AL 12 ASIGNAMOS LAS CARTAS AL MAZO DE LOS JUGADORES
        System.out.println("MAZO JUGADOR 1: " + mazoJugador1+""
                + ""+"\nMAZO JUGADOR 2: " +mazoJugador2+""
                + ""+"\nMAZO JUGADOR 3: " +mazoJugador3+""
                + ""+"\nMAZO JUGADOR 4: " +mazoJugador4);
    }
}

//COMO SE JUEGA
if(vistaMenu.buttoncomoSeJuega.equals(evento.getSource()))
{
    this.modelo.ayuda();
}

//CONSULTA MEJORES JUGADORES
else if(vistaMenu.buttonMejoresJugadores.equals(evento.getSource()))
{
    vistaMejoresJ.ventanaMejoresJugadores.setVisible(true);
    //CONECTAMOS A LA BASE DE DATOS
    conexion = this.modelo.conectar();
    //REALIZAR LA CONSULTA E INSERTAR INFORMACIÓN
    informacion = this.modelo.mejoresJugadores(conexion);
    //RELLENAMOS EL TEXTAREA
    this.vistaMejoresJ.listadoJugadores.append(informacion);
    //CERRAMOS CONEXION
    this.modelo.cerrar(conexion);
}

//BOTONES QUE CIERRAN VENTANAS
if(vistaMenu.buttonSalirMenu.equals(evento.getSource()))
{
    System.exit(0);
}


else if(vistaCrearP.cerrarPartida.equals(evento.getSource()))
{
    this.vistaCrearP.setVisible(false);

}
```

```
    else if(vistaMejoresJ.cerrar.equals(evento.getSource()))
    {
        this.vistaMejoresJ.ventanaMejoresJugadores.setVisible(false);
    }

}

//FUNCIONALIDAD DEL CLOSING
public void windowClosing(WindowEvent evento)
{
    if (vistaMenu.isActive())
    {
        vistaMenu.setVisible(false);
    }

    if(vistaCrearP.isActive())
    {
        this.vistaCrearP.setVisible(false);
    }

    else if(vistaJugando.isActive())
    {
        this.vistaJugando.setVisible(false);
    }

    else if(vistaMejoresJ.ventanaMejoresJugadores.isActive())
    {
        this.vistaMejoresJ.ventanaMejoresJugadores.setVisible(false);
    }


    //DIALOGOS
    else if(vistaJugando.dialogoAcusacion1.isActive())
    {
        vistaJugando.dialogoAcusacion1.setVisible(false);
    }

    else if(vistaJugando.dialogoAcusacion2.isActive())
    {
        vistaJugando.dialogoAcusacion2.setVisible(false);
    }

    else if(vistaJugando.dialogoComienzo.isActive())
    {
        vistaJugando.dialogoComienzo.setVisible(false);
    }

    else if(vistaJugando.dialogoVictoria.isActive())
    {
        vistaJugando.setVisible(false);
    }
```

```
        else if(vistaCrearP.dialogoMensajePartidaCreada.isActive())
        {
            vistaCrearP.dialogoMensajePartidaCreada.setVisible(false);
        }
    }

    //FUNCIONALIDAD DEL JUEGO
    public void mouseClicked(MouseEvent evento)
    {
        int x = evento.getX();
        int y = evento.getY();

        //TURNOS
        /* JUGADOR 1==1
         * JUGADOR 2==2
         * JUGADOR 3==3
         * JUGADOR 4==4
         */

        //TURNO JUGADOR 1
        if((x>=340)&&(x<=449)&&(y>=65)&&(y<=215)&&(turno==1))

        {
            controlTurno=false;
            cartasLanzadas = this.modelo.accionJugador(mazoJugador1);//MAZO ACTUALIZADO
            System.out.println("LANZA CARTAS EL JUGADOR 1 " + cartasLanzadas);
            mazoJugador1 =this.modelo.lanzamientos(mazoJugador1, cartasLanzadas);//LANZA CARTA
            System.out.println("LE QUEDAN AL JUGADOR 1: " +mazoJugador1);
            mazoCentral=this.modelo.mazoCartaCentral(mazoCentral, cartasLanzadas);//LAS CARTAS DEL JUGADOR PASAN AL MAZO CENTRAL
            this.vistaJugando.reversoCarta();
            turno = 2;
            if(turno==2)
            {
                this.vistaJugando.avisoTurno(+2);
            }
            System.out.println("JUGADOR 2 ¿LANZAS O LEVANTAS?");

        }

        if((x>=340)&&(x<=449)&&(y>=165)&&(y<=315)&&(turno==1))
        {
            if(controlTurno==false)
            {
                boolean controlMentira=false;
                System.out.println("JUGADOR 1 LLAMA MENTIROSO AL JUGADOR 4. LEVANTA SUS CARTAS.");
                for(int i=0;i<cartasLanzadas.size();i++)
                {
                    if(numeroInicial!=cartasLanzadas.get(i))
                    {
                        controlMentira=true; //SI ES TRUE, ADVERSARIO HA MENTIDO, PASA AL PRÓXIMO ELSE
                    }
                }
```

```
        if(controlMentira==false)
        {
            mazoJugador1 = this.modelo.devolverCartasAJugadores(mazoJugador1, mazoCentral);
            this.vistaJugando.lblNoMintio.setText("EL JUGADOR 4 NO MINTIÓ");
            this.vistaJugando.dialogoAcusacion1.setVisible(true);
            turno=2;
            controlTurno=true;
            this.vistaJugando.avisoTurno(+2);
            System.out.println("EL JUGADOR 2, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }

        else
        {
            mazoJugador4 = this.modelo.devolverCartasAJugadores(mazoJugador4, mazoCentral);
            this.vistaJugando.lblMentido.setText("EL JUGADOR 4 MINTIÓ");
            this.vistaJugando.dialogoAcusacion2.setVisible(true);
            turno=1;
            System.out.println("EL JUGADOR 1, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }
        mazoCentral = new ArrayList<Integer>();

    }

}
```

```
//TURNO JUGADOR 2
else if ((x>=340)&&(x<=449)&&(y>=295)&&(y<=445)&&(turno==2))
{
    controlTurno=false;
    cartasLanzadas = this.modelo.accionJugador(mazoJugador2);//MAZO ACTUALIZADO
    System.out.println("LANZA CARTAS EL JUGADOR 2 " + cartasLanzadas);
    mazoJugador2 =this.modelo.lanzamientos(mazoJugador2, cartasLanzadas);//LANZA CARTA
    System.out.println("LE QUEDAN AL JUGADOR 2: " +mazoJugador2);
    mazoCentral=this.modelo.mazoCartaCentral(mazoCentral, cartasLanzadas);//LAS CARTAS DEL JUGADOR PASAN AL MAZO CENTRAL
    this.vistaJugando.reversoCarta();
    turno = 3;
    if(turno==3)
    {
        this.vistaJugando.avisoTurno(+3);
    }
    System.out.println("JUGADOR 3 ¿LANZAS O LEVANTAS?");
}
if((x>=340)&&(x<=449)&&(y>=165)&&(y<=315)&&(turno==2))
{
    if(controlTurno==false)
    {
        boolean controlMentira=false;
        System.out.println("JUGADOR 2 LLAMA MENTIROSO AL JUGADOR 1. LEVANTA SUS CARTAS.");
        for(int i=0;i<cartasLanzadas.size();i++)
        {

            if(numeroInicial!=cartasLanzadas.get(i))
            {
                controlMentira=true;//SI ES TRUE, ADVERSARIO HA MENTIDO, PASA AL PRÓXIMO ELSE
            }
        }
        if(controlMentira==false)
        {
            mazoJugador2 = this.modelo.devolverCartasAJugadores(mazoJugador2, mazoCentral);
            this.vistaJugando.lblNoMintio.setText("EL JUGADOR 1 NO MINTIÓ");
            this.vistaJugando.dialogoAcusacion1.setVisible(true);
            turno=3;
            controlTurno=true;
            this.vistaJugando.avisoTurno(+3);
            System.out.println("EL JUGADOR 3, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }
        else
        {
            mazoJugador1 = this.modelo.devolverCartasAJugadores(mazoJugador1, mazoCentral);
            this.vistaJugando.lblMentido.setText("EL JUGADOR 1 MINTIÓ");
            this.vistaJugando.dialogoAcusacion2.setVisible(true);
            turno=2;
            System.out.println("EL JUGADOR 2, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }

        mazoCentral = new ArrayList<Integer>();
    }
```

```
//TURNO JUGADOR 3
else if((x>=60)&&(x<=169)&&(y>=180)&&(y<=330)&&(turno==3))//JUGADOR 3 LANZA CARTA
{
    controlTurno=false;
    this.vistaJugando.avisoTurno(+1);
    cartasLanzadas = this.modelo.accionJugador(mazoJugador3);//MAZO ACTUALIZADO
    System.out.println("LANZA CARTAS EL JUGADOR 3 " + cartasLanzadas);
    mazoJugador3 =this.modelo.lanzamientos(mazoJugador3, cartasLanzadas);
    System.out.println("LE QUEDAN AL JUGADOR 3: " +mazoJugador3);
    mazoCentral=this.modelo.mazoCartaCentral(mazoCentral, cartasLanzadas);
    this.vistaJugando.reversoCarta();
    turno = 4;
    if(turno==4)
    {
        this.vistaJugando.avisoTurno(+4);
    }
    System.out.println("JUGADOR 4 ¿LANZAS O LEVANTAS?");
}
//JUGADOR 3 LEVANTA CARTAS DEL JUGADOR 2
if((x>=340)&&(x<=419)&&(y>=165)&&(y<=315)&&(turno==3))
{
    if(controlTurno==false)
    {
        boolean controlMentira=false;
        System.out.println("JUGADOR 3 LLAMA MENTIROSO AL JUGADOR 2. LEVANTA SUS CARTAS.");

        for(int i=0;i<cartasLanzadas.size();i++)
        {
            if(numeroInicial!=cartasLanzadas.get(i))
            {
                controlMentira=true;//SI ES TRUE, ADVERSARIO HA MENTIDO, PASA AL PRÓXIMO ELSE
            }
        }
        if(controlMentira==false)
        {
            mazoJugador3 = this.modelo.devolverCartasAJugadores(mazoJugador3, mazoCentral);
            this.vistaJugando.lblNoMintio.setText("EL JUGADOR 2 NO MINTIÓ");
            this.vistaJugando.dialogoAcusacion1.setVisible(true);
            turno=4;
            controlTurno=true;
            this.vistaJugando.avisoTurno(+4);
            System.out.println("EL JUGADOR 4, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }
        else
        {
            mazoJugador2 = this.modelo.devolverCartasAJugadores(mazoJugador2, mazoCentral);
            this.vistaJugando.lblMentido.setText("EL JUGADOR 2 MINTIÓ");
            this.vistaJugando.dialogoAcusacion2.setVisible(true);
            turno=3;
            System.out.println("EL JUGADOR 3, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }
        mazoCentral = new ArrayList<Integer>();
    }
```

```
//TURNO JUGADOR 4
else if((x>=650)&&(x<=759)&&(y>=180)&&(y<=330)&&(turno==4))
{
    controlTurno=false;
    this.vistaJugando.avisoTurno(+1);
    cartasLanzadas = this.modelo.accionJugador(mazoJugador4);//MAZO ACTUALIZADO
    System.out.println("LANZA CARTAS EL JUGADOR 4 " + cartasLanzadas);
    mazoJugador4 =this.modelo.lanzamientos(mazoJugador4, cartasLanzadas); //JUGADOR LANZA CARTAS
    System.out.println("LE QUEDAN AL JUGADOR 4: " + mazoJugador4);
    mazoCentral=this.modelo.mazoCartaCentral(mazoCentral, cartasLanzadas);//LAS CARTAS DEL JUGADOR PASAN AL MAZO CENTRAL
    this.vistaJugando.reversoCarta();
    turno = 1;
    System.out.println("JUGADOR 1 ¿LANZAS O LEVANTAS?");
}

//JUGADOR 4 LEVANTA CARTAS DEL JUGADOR 3
if((x>=340)&&(x<=449)&&(y>=165)&&(y<=315)&&(turno==4))
{
    if(controlTurno==false)
    {
        boolean controlMentira=false;
        System.out.println("JUGADOR 4 LLAMA MENTIROSO AL JUGADOR 3. LEVANTA SUS CARTAS.");
        for(int i=0;i<cartasLanzadas.size();i++)
        {
            if(numeroInicial!=cartasLanzadas.get(i))
            {
                controlMentira=true;//SI ES TRUE, ADVERSARIO HA MENTIDO, PASA AL PRÓXIMO ELSE
            }
        }

        if(controlMentira==false)
        {
            mazoJugador4 = this.modelo.devolverCartasAJugadores(mazoJugador4, mazoCentral);
            this.vistaJugando.lblNoMintio.setText("EL JUGADOR 3 NO MINTIÓ");
            this.vistaJugando.dialogoAcusacion1.setVisible(true);
            turno=1;
            controlTurno=true;
            this.vistaJugando.avisoTurno(+1);
            System.out.println("EL JUGADOR 1, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }
        else
        {
            mazoJugador3 = this.modelo.devolverCartasAJugadores(mazoJugador3, mazoCentral);
            this.vistaJugando.lblMentido.setText("EL JUGADOR 3 MINTIÓ");
            this.vistaJugando.dialogoAcusacion2.setVisible(true);
            turno=4;
            System.out.println("EL JUGADOR 4, SELECCIONA UN NUEVO NUMERO EL:" + (numeroInicial=this.modelo.nuevoNumeroSeleccionado(null)));
        }
        mazoCentral = new ArrayList<Integer>();

    }

}
```

```
//JUGADOR GANADOR
if (mazoJugador1.isEmpty())
{
    puntuacionJugador1++;
    this.vistaJugando.lblVictoria.setText("EL JUGADOR 1 " + this.vistaCrearP.textoNombreJugador1.getText() + " HA GANADO");
    this.vistaJugando.dialogoVictoria.setVisible(true);
    //CONECTAMOS A LA BASE DE DATOS
    conexion = this.modelo.conectar();
    //INSERTAR INFORMACIÓN
    this.modelo.crearPartidaNueva(conexion,this.vistaCrearP.textoNombreJugador1.getText(),puntuacionJugador1);
    //CERRAMOS CONEXION
    this.modelo.cerrar(conexion);
}
else if (mazoJugador2.isEmpty())
{
    puntuacionJugador2++;
    this.vistaJugando.lblVictoria.setText("EL JUGADOR 2 " + this.vistaCrearP.textoNombreJugador2.getText() + " HA GANADO");
    this.vistaJugando.dialogoVictoria.setVisible(true);
    //CONECTAMOS A LA BASE DE DATOS
    conexion = this.modelo.conectar();
    //INSERTAR INFORMACIÓN
    this.modelo.crearPartidaNueva(conexion,this.vistaCrearP.textoNombreJugador2.getText(),puntuacionJugador2);
    //CERRAMOS CONEXION
    this.modelo.cerrar(conexion);
}
else if (mazoJugador3.isEmpty())
{
    puntuacionJugador3++;
    this.vistaJugando.lblVictoria.setText("EL JUGADOR 3 " + this.vistaCrearP.textoNombreJugador3.getText() + " HA GANADO");
    this.vistaJugando.dialogoVictoria.setVisible(true);
    //CONECTAMOS A LA BASE DE DATOS
    conexion = this.modelo.conectar();
    //INSERTAR INFORMACIÓN
    this.modelo.crearPartidaNueva(conexion,this.vistaCrearP.textoNombreJugador3.getText(),puntuacionJugador3);
    //CERRAMOS CONEXION
    this.modelo.cerrar(conexion);
}
else if (mazoJugador4.isEmpty())
{
    puntuacionJugador4++;
    this.vistaJugando.lblVictoria.setText("EL JUGADOR 4 " + this.vistaCrearP.textoNombreJugador4.getText() + " HA GANADO");
    this.vistaJugando.dialogoVictoria.setVisible(true);
    //CONECTAMOS A LA BASE DE DATOS
    conexion = this.modelo.conectar();
    //INSERTAR INFORMACIÓN
    this.modelo.crearPartidaNueva(conexion,this.vistaCrearP.textoNombreJugador4.getText(),puntuacionJugador4);
    //CERRAMOS CONEXION
    this.modelo.cerrar(conexion);
}
```

- Sonido

```
public class Sonido
{
    public Sonido()
    {
        File sf = new File("Cantina band.wav");
        AudioFileFormat aff;
        AudioInputStream ais;
        try
        {
            aff = AudioSystem.getAudioFileFormat(sf);
            ais = AudioSystem.getAudioInputStream(sf);
            AudioFormat af = aff.getFormat();
            DataLine.Info info = new DataLine.Info(Clip.class, ais.getFormat(),
                    ((int) ais.getFrameLength() * af.getFrameSize()));
            Clip ol = (Clip) AudioSystem.getLine(info);
            ol.open(ais);
            ol.loop(0);
            // Damos tiempo para que el sonido sea escuchado
            Thread.sleep(150000);
            ol.close();
        }
        catch(UnsupportedAudioFileException ee)
        {
            System.out.println(ee.getMessage());
        }
        catch(IOException ea)
        {
            System.out.println(ea.getMessage());
        }
        catch(LineUnavailableException LUE)
        {
            System.out.println(LUE.getMessage());
        }
        catch (InterruptedException e)
        {
            e.printStackTrace();
        }
    }
}
```

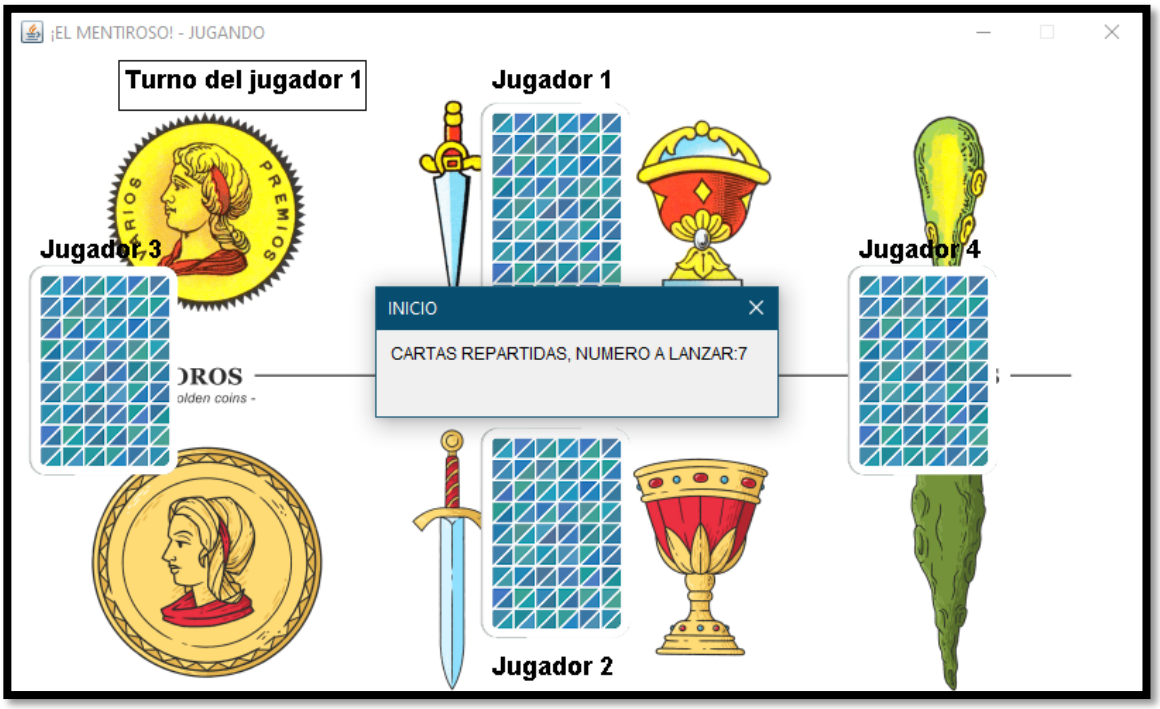
4. Funcionalidad del juego y ventanas

- Menú principal

- **Nueva Partida**

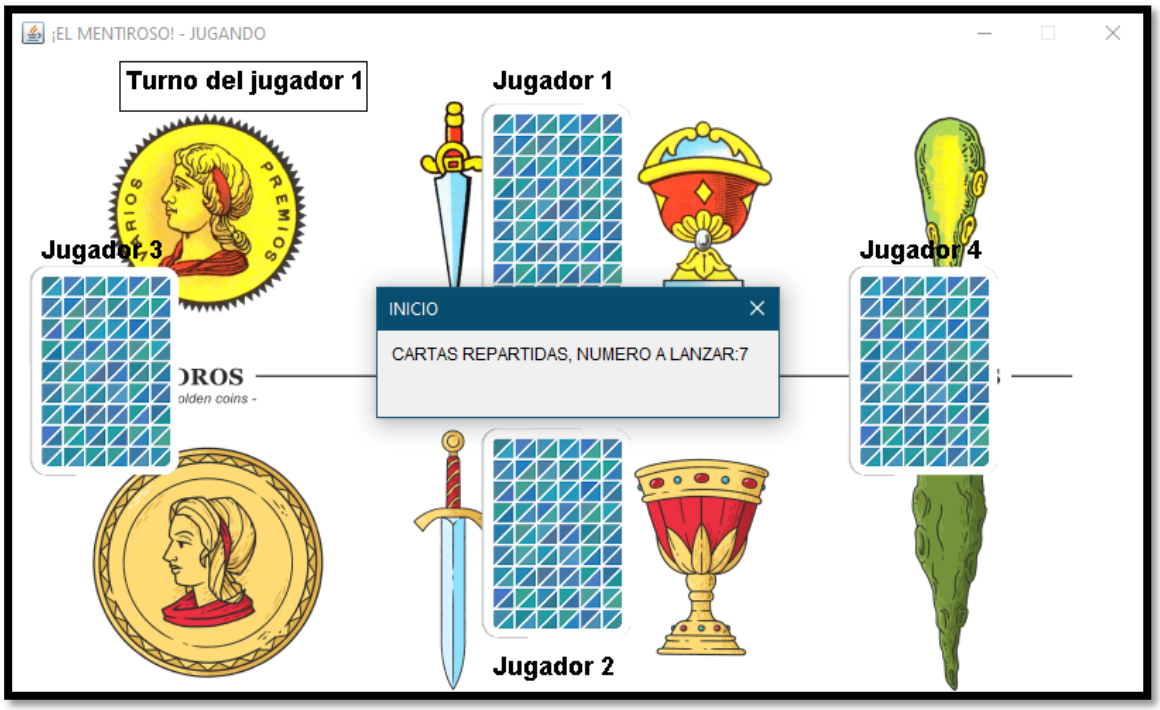
1. Campos vacíos

2. Campos rellenados



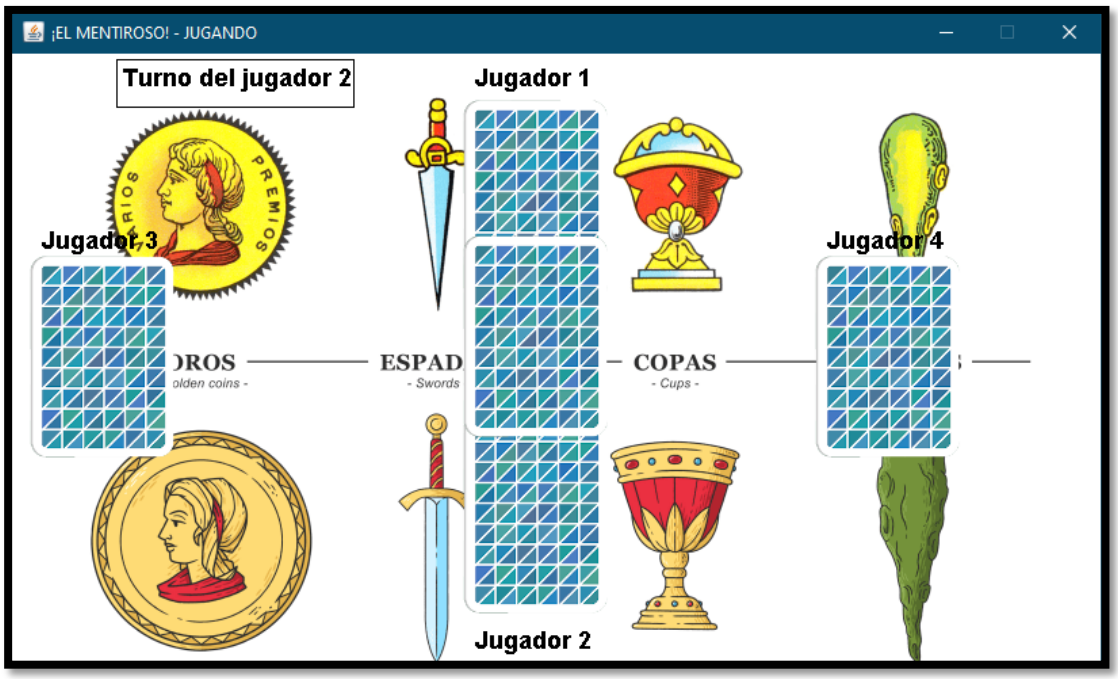
- Jugando

1. Inicio del juego



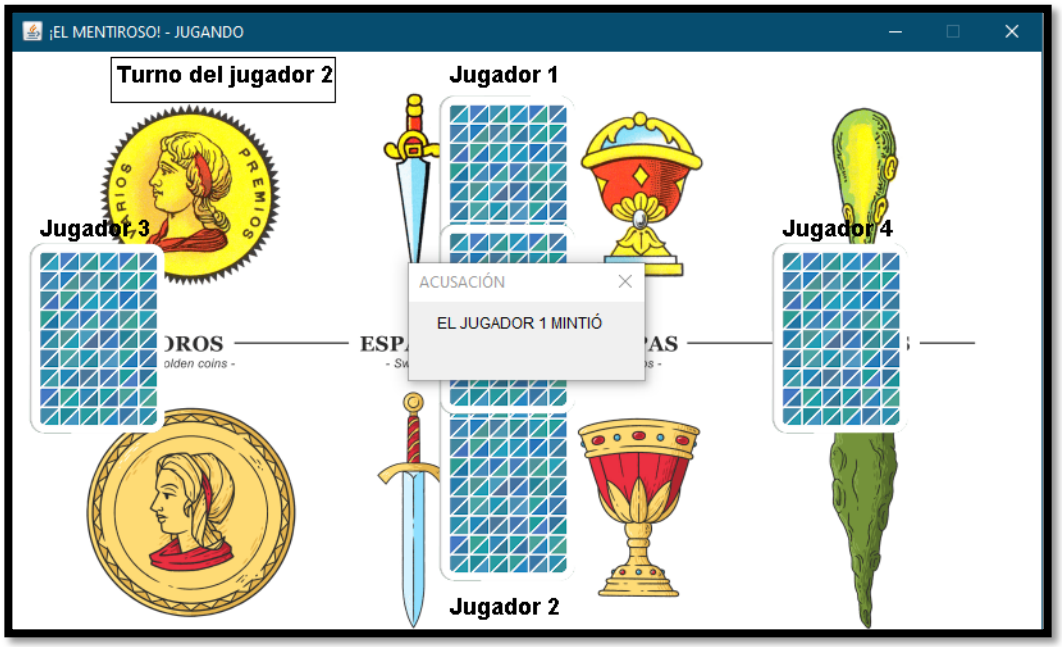

```
MAZO JUGADOR 1: [2, 3, 7, 9, 7, 9, 1, 4, 5, 7, 10, 11]
MAZO JUGADOR 2: [1, 4, 5, 6, 8, 10, 11, 12, 1, 2, 3, 4]
MAZO JUGADOR 3: [5, 6, 8, 10, 11, 12, 2, 3, 6, 8, 9, 12]
MAZO JUGADOR 4: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
```

2. Lanzamiento de carta/s



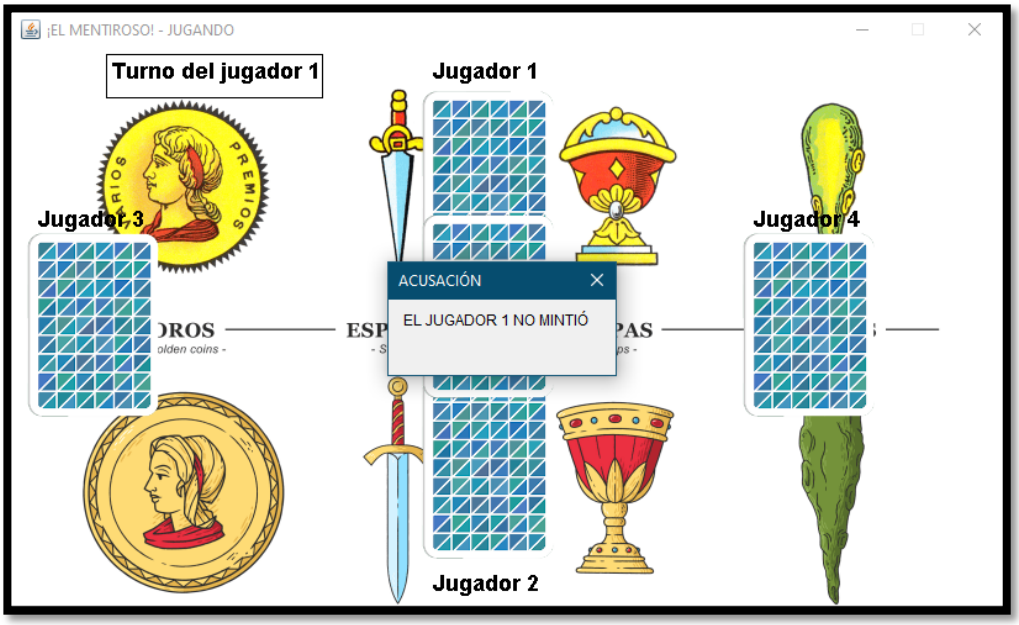

```
MAZO JUGADOR 1: [2, 3, 7, 9, 7, 9, 1, 4, 5, 7, 10, 11]
MAZO JUGADOR 2: [1, 4, 5, 6, 8, 10, 11, 12, 1, 2, 3, 4]
MAZO JUGADOR 3: [5, 6, 8, 10, 11, 12, 2, 3, 6, 8, 9, 12]
MAZO JUGADOR 4: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12]
LANZA CARTAS EL JUGADOR 1 [7, 7, 7]
LE QUEDAN AL JUGADOR 1: [2, 3, 9, 9, 1, 4, 5, 10, 11]
JUGADOR 2 ¿LANZAS O LEVANTAS?
```

3. Levantar carta de jugador + Mintió



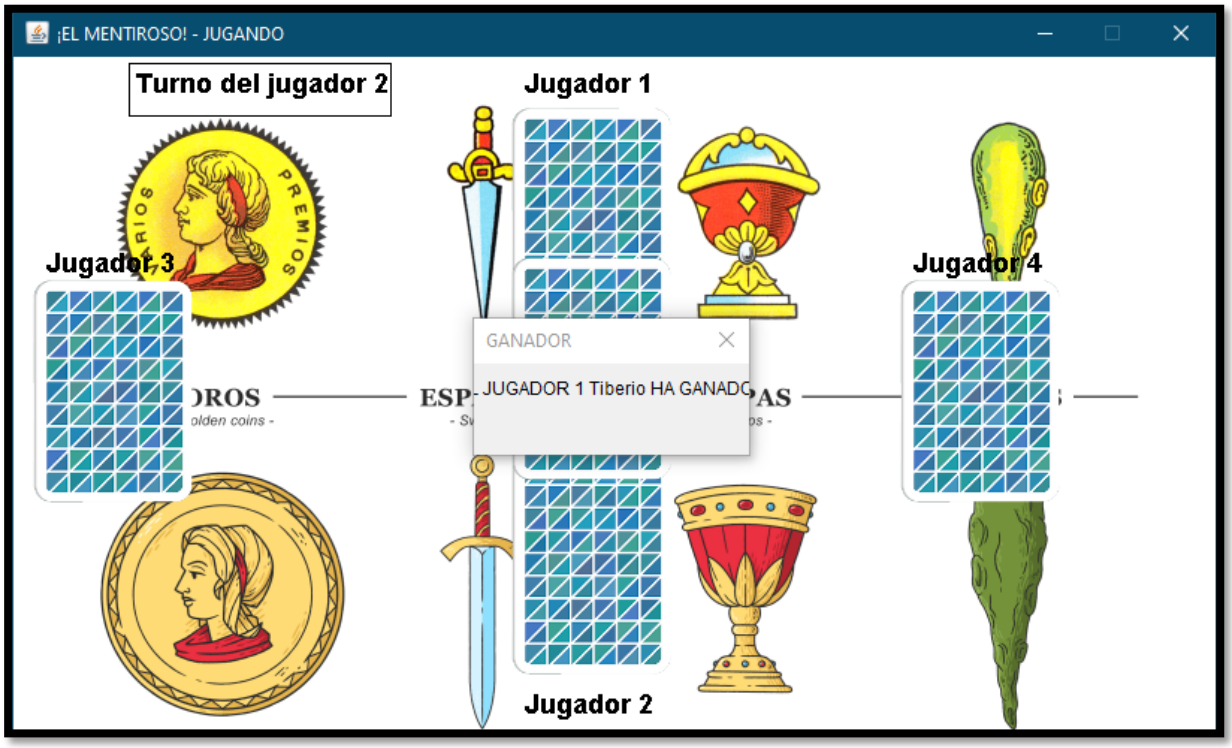

```
LANZA CARTAS EL JUGADOR 1 [1, 11, 8, 5]
LE QUEDAN AL JUGADOR 1: [3, 1, 7, 10, 11, 12, 4, 5]
JUGADOR 2 ¿LANZAS O LEVANTAS?
JUGADOR 2 LLAMA MENTIROSO AL JUGADOR 1. LEVANTA SUS CARTAS.
EL JUGADOR 2, SELECCIONA UN NUEVO NUMERO EL:1
```

4. Levantar carta de jugador + No mintió



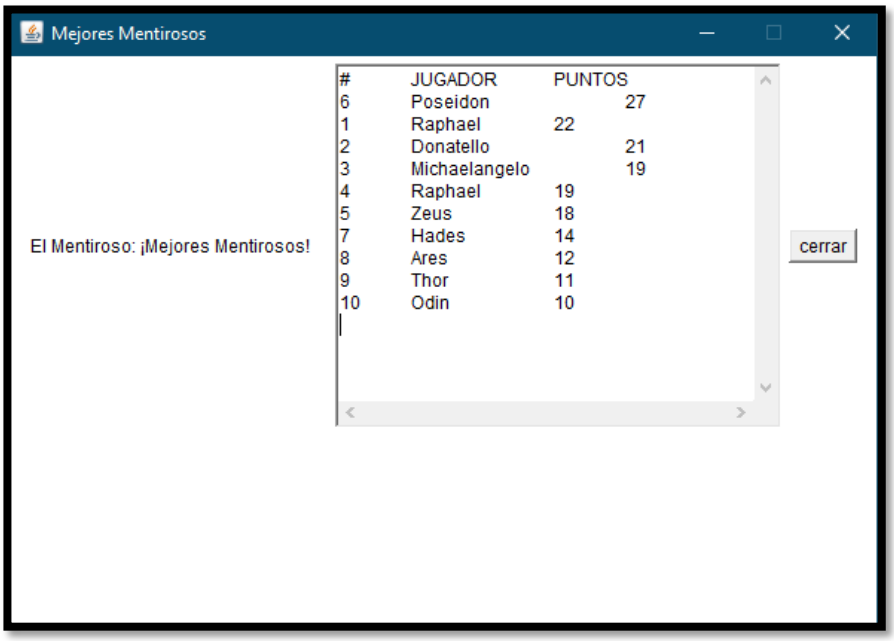

```
JUGADOR 2 ¿LANZAS O LEVANTAS?
JUGADOR 2 LLAMA MENTIROSO AL JUGADOR 1. LEVANTA SUS CARTAS.
EL JUGADOR 3, SELECCIONA UN NUEVO NUMERO EL:12
```

5. Victoria e inserción en la base de datos.



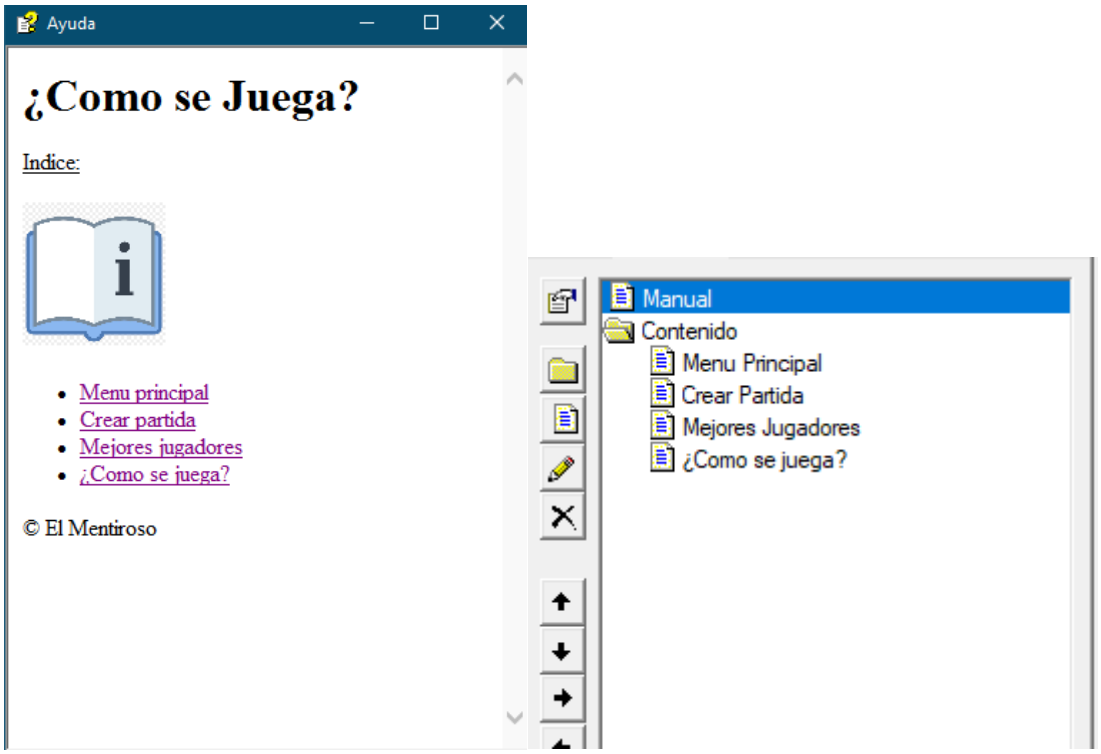

```
LANZA CARTAS EL JUGADOR 1 [8]
LE QUEDAN AL JUGADOR 1: []
JUGADOR 2 ¿LANZAS O LEVANTAS?
INSERT INTO jugadores VALUES (null, 'Tiberio','1')
```

- **Ventana de mejores jugadores**



#	JUGADOR	PUNTOS
6	Poseidon	27
1	Raphael	22
2	Donatello	21
3	Michaelangelo	19
4	Raphael	19
5	Zeus	18
7	Hades	14
8	Ares	12
9	Thor	11
10	Odin	10

- **Manual de ayuda + tabla de contenidos**



5. Imports

- **vistaCrearPartida y vistaMejoresJ**

```java
package es.Studium.Vistas;

import java.awt.Button;
import java.awt.Dialog;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Label;
import java.awt.TextField;
import java.awt.Toolkit;

public class VistaCrearPartida ext
```

```java
package es.Studium.Vistas;


import java.awt.Button;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextArea;



public class VistaMejoresJ  extends Frame
```

- **vistaMenuPrincipal y vistaJugando**

```java
package es.Studium.Vistas;

import java.awt.Button;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Label;
import java.awt.Toolkit;



public class VistaMenuPrincipal extends Frame
```

```java
package es.Studium.Vistas;

import java.awt.Color;
import java.awt.Dialog;
import java.awt.FlowLayout;
import java.awt.Font;
import java.awt.Frame;
import java.awt.Graphics;
import java.awt.Image;
import java.awt.Label;
import java.awt.Toolkit;

public class VistaJugando extends Frame
```

- **vistaMejoresJugadores**

```java
package es.Studium.Vistas;

import java.awt.Button;
import java.awt.Color;
import java.awt.FlowLayout;
import java.awt.Frame;
import java.awt.Label;
import java.awt.TextArea;
```

- **Modelo y controlador**

```java
import java.io.IOException;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.Random;
```

```java
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;
import java.awt.event.WindowEvent;
import java.awt.event.WindowListener;
import java.sql.Connection;
import java.util.ArrayList;

import es.Studium.Modelo.Modelo;
import es.Studium.Vistas.VistaCrearPartida;
import es.Studium.Vistas.VistaJugando;
import es.Studium.Vistas.VistaMejoresJ;
import es.Studium.Vistas.VistaMenuPrincipal;
```

- **Sonido**

```
import java.io.File;
import java.io.IOException;
import javax.sound.sampled.AudioFileFormat;
import javax.sound.sampled.AudioFormat;
import javax.sound.sampled.AudioInputStream;
import javax.sound.sampled.AudioSystem;
import javax.sound.sampled.Clip;
import javax.sound.sampled.DataLine;
import javax.sound.sampled.LineUnavailableException;
import javax.sound.sampled.UnsupportedAudioFileException;

public class Sonido
{
```

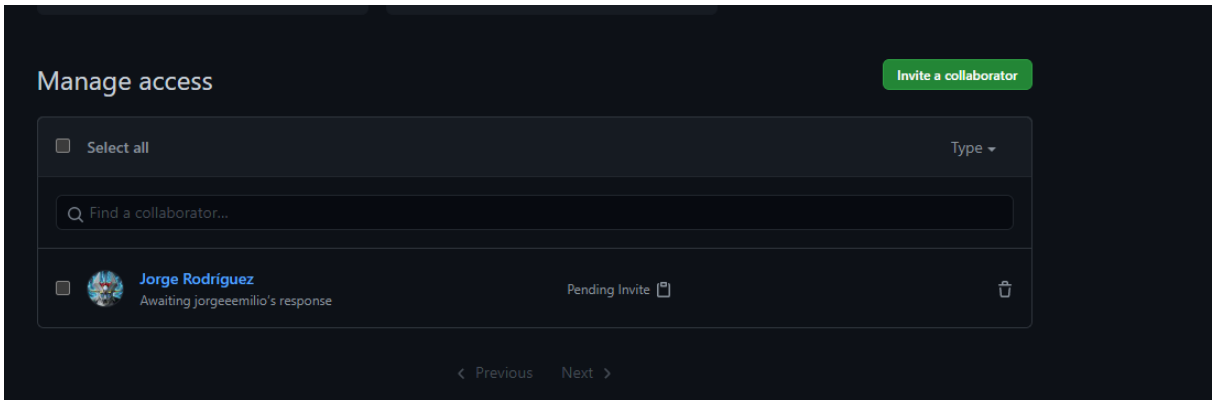
- **Main** (Insertada en su propio pantallazo del código)

6. Conclusión final

Considero esta última práctica como la más complicada que he realizado en toda la asignatura de programación. Y he entendido perfectamente el objetivo de la misma, pues aglutina en un todo, prácticamente la totalidad del conocimiento que hemos ido aprendiendo a lo largo de la asignatura, desde el primer tema hasta el final, strings, arrays, iteradores, bucles, condiciones, frames, dibujos, listeners, MVC, etc…Y a pesar de haberme quedado bloqueado en alguna que otra ocasión dentro de la práctica, en lugar de frustrarme, me obligaba a buscar alternativas, investigar y dar con soluciones posiblemente mucho más productivas que las que estaba intentando con anterioridad. Por lo que, si ese era el objetivo de la asignatura, además de aprender a programar, puedo decir, o creo que puedo decir, que lo he conseguido. En conclusión, una práctica compleja (dependiendo por supuesto también del juego), pero disfrutable que favorece mucho el asentar el aprendizaje y aprender todavía más.

7. Github

- https://github.com/carleovaz/Programacion_Juego



8. Bibliografía:

- https://www.javatpoint.com/array-vs-arraylist-in-java#:~:text=Array%20is%20a%20fixed%20length,it%20can%20only%20store%20objects.
- https://docs.oracle.com/javase/8/docs/api/java/util/Random.html
- https://www.arquitecturajava.com/java-arraylist-for-y-sus-opciones/