



# Ruleta Americana

Java

Adrian Potenciano Vila



## Contenido

1. ¿De qué trata el proyecto? .....	2
2. Estructura .....	2
3. Funcionalidad de cada clase.....	3
3.1. Models.....	3
3.2. Controller .....	18
3.3. Dialogos .....	23
3.4. View .....	<b>¡Error! Marcador no definido.</b>
4. Interfaz Grafica .....	29
5. Test .....	44
6. Tecnologías utilizadas y referencias.....	44

## 1. ¿De qué trata el proyecto?

La aplicación es una implementación de la ruleta americana en formato de aplicación de escritorio. Ofrece una experiencia interactiva donde los usuarios pueden realizar apuestas en números individuales, grupos de números, colores o pares/impares. Además, la aplicación muestra un panel de control que permite a los usuarios realizar un seguimiento de sus apuestas, ver los números ganadores anteriores y actualizar su saldo. Con una interfaz intuitiva y visualmente atractiva, la aplicación brinda una experiencia auténtica de la ruleta americana para los usuarios desde la comodidad de su computadora sin gastar dinero real.

## 2. Estructura

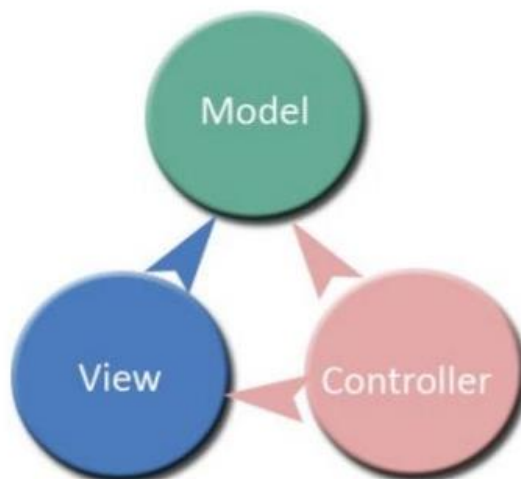
Este proyecto ha sido realizado a partir de un **MVC**.

MVC es un patrón de diseño de software que separa la aplicación en tres componentes principales: Modelo (Model), Vista (View) y Controlador (Controller). Este enfoque proporciona una clara separación de responsabilidades, lo que facilita el desarrollo, mantenimiento, escalabilidad de la aplicación y hacer los test.

**-Modelo (Model):** Representa la lógica de la aplicación y maneja directamente los datos. El modelo no tiene conocimiento de la vista ni del controlador.

**-Vista (View):** Es la interfaz de usuario. Presenta los datos del modelo al usuario y envía las acciones del usuario al controlador. La vista no tiene conocimiento directo del modelo.

**-Controlador (Controller):** Interactúa con el modelo para actualizar los datos y con la vista para reflejar esos cambios. El controlador actúa como un intermediario entre el modelo y la vista.



## 3. Funcionalidad de cada clase

### 3.1. Models

#### -ApuestaClikables

La clase **ApuesatsClikables** representa una colección de áreas clicables en la interfaz de usuario del juego de la ruleta. Permite determinar qué tipo de apuesta se realiza al hacer clic en una determinada área en la interfaz de la ruleta.

#### Atributos

**medioAncho, CuartoAncho, SegmentoAncho**: Variables que almacenan diferentes dimensiones relacionadas con los segmentos de la ruleta y se utilizan para calcular las posiciones de las áreas clicables.

**margen, imagenMargen, lineaMargen, MargenTotal**: Variables que almacenan los márgenes y dimensiones de la interfaz de la ruleta.

**apuestaClikables (List<ApuestaClikable>)**: Una lista que almacena las diferentes áreas clicables de la ruleta.

**ruleta (Ruleta)**: Una instancia de la clase Ruleta utilizada para obtener información sobre los números de la ruleta y otros tipos de apuestas.

#### Constructor

**public ApuesatsClikables ()**:

Este constructor inicializa los márgenes y dimensiones de las áreas clicables y crea todas las áreas clicables disponibles en la ruleta.

#### Métodos

**private List<ApuestaClikable> crearApuestasClikables ()**:

Método privado que inicializa y crea todas las áreas clicables disponibles en la ruleta.

**public ApuestaClikable contieneApuestaClikable (Point point)**:

Método que verifica si un punto dado se encuentra dentro de alguna de las áreas clicables de la ruleta y devuelve la apuesta correspondiente si se encuentra, o null si no se encuentra.

## -Apuesta

La clase **Apuesta** encapsula todos los detalles relacionados con una apuesta individual en el juego de la ruleta.

### Atributos

**boolean EsApuestaGanadora**: indica si la apuesta es ganadora.

**int ContadorFichas**: almacena la cantidad de fichas apostadas.

**int ValorFicha**: almacena el valor de cada ficha apostada.

**Point UbicacionBoard**: la ubicación de la apuesta en el tablero, representada como un punto (x, y).

**Jugador jugador**: el jugador que realizó la apuesta.

**String [] numerosApostados**: una lista de números en los que se ha apostado.

**TipoApuesta TipoApuesta**: el tipo de apuesta realizada.

### Constructor

**public Apuesta (Point ubicacionBoard, Jugador jugador, TipoApuesta tipoApuesta, String... numerosApostados)**: inicializa los atributos UbicacionBoard, jugador, TipoApuesta y numerosApostados con los valores proporcionados.

### Métodos

**public boolean EsMismaApuesta (Apuesta otraApuesta)**:

Compara la ubicación de dos apuestas para determinar si son la misma.

**otraApuesta**: Otra instancia de Apuesta para comparar.

Devuelve true si las ubicaciones son las mismas, de lo contrario, devuelve false.

**public void ponerCantidadApuesta (int contadorFichas, int valorFichas)**:

Establece la cantidad de fichas y el valor de las fichas apostadas.

**contadorFichas**: Cantidad de fichas apostadas.

**valorFichas:** Valor de cada ficha apostada.

**public boolean isGanador (String numero):**

Verifica si un número dado está en los números apostados.

**numero:** El número que se verifica.

Devuelve true si el número está en los números apostados, de lo contrario, devuelve false.

**public boolean isEsApuestaGanadora ():**

Devuelve el estado de la apuesta, si es ganadora o no.

Devuelve true si la apuesta es ganadora, de lo contrario, devuelve false.

**public void setEsApuestaGanadora (boolean isEsApuestaGanadora):**

Establece el estado de la apuesta como ganadora o no.

**isApuestaGanadora:** true para marcar la apuesta como ganadora, false de lo contrario.

**public Jugador getJugador ():**

Devuelve el jugador que realizó la apuesta.

Devuelve una instancia de Jugador.

**public void incrementarContadorFichas (int increment):**

Incrementa el contador de fichas en una cantidad dada.

**increment:** La cantidad para incrementar el contador de fichas.

**public int getContadorFichas ():** Devuelve el contador de fichas apostadas.

Devuelve la cantidad de fichas apostadas.

**public int getValorFicha ():** Devuelve el valor de cada ficha apostada.

Devuelve el valor de la ficha.

**public Point getUbicacionBoard ():** Devuelve la ubicación de la apuesta en el tablero.

Devuelve una instancia de Point que representa la ubicación en el tablero.

**public RuletaApp.model. TipoApuesta getTipoApuesta ():**

Devuelve el tipo de apuesta realizada. Devuelve una instancia de TipoApuesta.

## -ApuestaClicable

La clase **ApuestaClicable** representa un área interactiva en el tablero de la ruleta donde un jugador puede hacer una apuesta.

### Atributos

**limite (Rectangle):** Define los límites del área interactiva en el tablero.

**numeros (String []):** Los números que están involucrados en esta área de apuesta.

**TipoApuesta (TipoApuesta):** El tipo de apuesta que esta área representa.

### Constructor

**public ApuestaClicable (Rectangle limite, TipoApuesta tipoApuesta, String... numeros):**

Inicializa una nueva instancia de ApuestaClicable con los límites del área, el tipo de apuesta y los números involucrados.

### Métodos

**public Point getPuntoMedio ():**

Calcula y devuelve el punto medio del área interactiva.

Calcula las coordenadas x e y del punto medio del rectángulo limite.

Devuelve un objeto Point que representa el punto medio.

**public Point getPoint (int position):**

Calcula y devuelve un punto específico dentro del área interactiva basado en una posición dada.

**position:** Un entero que representa la posición dentro de un ancho de cuatro segmentos (1, 2 o 3).

Calcula las coordenadas x e y basadas en la posición proporcionada.

Devuelve un objeto Point que representa el punto calculado.

**public Rectangle getLimite ():**

Devuelve el área interactiva (rectángulo) de la apuesta.

Devuelve un objeto Rectangle que representa el límite del área interactiva.

**public String [] getNumeros ():**

Devuelve los números asociados con esta región de apuesta.

Devuelve un array de cadenas (String []) que contiene los números.

**public TipoApuesta getTipoApuesta ():**

Devuelve el tipo de apuesta asociada con esta región.

Devuelve un objeto TipoApuesta que representa el tipo de apuesta.

### -FichaRuleta

La clase **FichaRuleta** representa una ficha utilizada en la ruleta del juego.

#### Atributos

**ImagenFicha (BufferedImage):** Representa la imagen de la ficha.

**ColorFicha (Color):** El color base de la ficha.

**ColorResaltado (Color):** El color de resaltado de la ficha.

#### Constructor

**public FichaRuleta (Color colorFicha, Color ColorResaltado):**

Inicializa una nueva instancia de FichaRuleta con los colores especificados.

#### Métodos

**public BufferedImage getImagenFicha ():**

Devuelve la imagen de la ficha generada a partir de los colores base y de resaltado.

Devuelve un objeto BufferedImage que representa la imagen de la ficha.

**public Color getColorFicha ():**

Devuelve el color base de la ficha.

Devuelve un objeto Color que representa el color base.

**public Color getColorResaltado ():**

Devuelve el color de resaltado de la ficha.

Devuelve un objeto Color que representa el color de resaltado.

### -FichasRuleta

La clase **FichasRuleta** gestiona una colección de fichas utilizadas en el juego de la ruleta. Permite añadir y eliminar fichas.

#### Atributos



**fichasRuleta (List<FichaRuleta>):** Una lista que almacena las fichas de la ruleta.

#### Constructor

**public FichasRuleta ():**

Inicializa una nueva instancia de FichasRuleta y genera una lista de fichas de ruleta predeterminadas.

#### Métodos

**private List<FichaRuleta> GenerearFicha ():**

Genera y devuelve una lista de fichas de ruleta con colores predefinidos.

Devuelve una lista de objetos FichaRuleta con colores predefinidos.

**public void addFichaRuleta (FichaRuleta fichaRuleta):**

Añade una nueva ficha a la colección.

**fichaRuleta:** La ficha de ruleta que se va a añadir.

**public void removeFichaRuleta (FichaRuleta fichaRuleta):**

Elimina una ficha de la colección.

**fichaRuleta:** La ficha de ruleta que se va a eliminar.

**public List<FichaRuleta> getFichasRuleta ():**

Devuelve la lista de fichas de ruleta disponibles.

#### -Jugador

La clase **Jugador** representa a un jugador en el juego de la ruleta. Gestiona el balance del jugador, la cantidad de fichas que posee, las fichas que ha comprado y su interacción con las fichas de la ruleta.

#### Atributos

**balance (int):** El saldo del jugador.

**CantidadCompra (int):** La cantidad de fichas que el jugador ha comprado.

**ContadorFicha (int):** El contador de fichas que el jugador posee.

**ValorFicha (ValorFicha):** El valor de las fichas del jugador.

**FichaRuleta (FichaRuleta):** La ficha de ruleta seleccionada por el jugador.

**nombre (String):** El nombre del jugador.

#### Constructor

**public Jugador (String nombre):**

Inicializa una nueva instancia de Jugador con el nombre especificado.

Métodos

**public void compra (int CantidadCompra, ValorFicha ValorFicha):**

Permite al jugador comprar fichas.

**CantidadCompra:** La cantidad de fichas que el jugador desea comprar.

**ValorFicha:** El valor de las fichas que el jugador desea comprar.

**public void EliminarFicha (int cantidad):**

Elimina fichas del contador del jugador y ajusta su saldo en consecuencia.

**cantidad:** La cantidad de fichas que se van a eliminar.

**public void addFichas (int cantidad):**

Agrega fichas al contador del jugador y ajusta su saldo en consecuencia.

**cantidad:** La cantidad de fichas que se van a agregar.

**public int getBalance ():**

Devuelve el saldo actual del jugador.

**public int getCantidadCompra ():**

Devuelve la cantidad de fichas compradas por el jugador.

**public int getContadorFicha ():**

Devuelve la cantidad de fichas que el jugador posee actualmente.

**public int getValorFicha ():**

Devuelve el valor de las fichas del jugador.

**public Color getChipColor ():**

Devuelve el color de la ficha de ruleta seleccionada por el jugador.

**public BufferedImage getImagenFicha ():**

Devuelve la imagen de la ficha de ruleta seleccionada por el jugador.

**public String getNombre ():**

Devuelve el nombre del jugador.

**@Override public String toString ():**

Devuelve una representación de cadena de texto del objeto Jugador, que incluye su nombre, saldo, cantidad comprada, contador de fichas y valor de fichas.

### -JugadorApuestaError

La clase JugadorApuestaError representa un error relacionado con una apuesta realizada por un jugador en el juego de la ruleta.

#### Atributos

**errorCode (int):** El código de error que indica el tipo de error ocurrido.

**jugador (Jugador):** El jugador asociado con el error de apuesta.

#### Constructor

**public JugadorApuestaError (int errorCode, Jugador jugador):**

Inicializa una nueva instancia de JugadorApuestaError con el código de error y el jugador asociado.

#### Métodos

**public int getErrorCode ():**

Devuelve el código de error que indica el tipo de error ocurrido.

**public Jugador getJugador ():**

Devuelve el jugador asociado con el error de apuesta.

### -Ronda

La clase **Ronda** representa una ronda de apuestas en el juego de la ruleta. Contiene métodos para gestionar y realizar operaciones con las apuestas realizadas por los jugadores durante la ronda.

#### Atributos

**Apuesta (List<Apuesta>):** Una lista de objetos Apuesta que representan las apuestas realizadas durante la ronda.

#### Constructor

**public Ronda ():**

Inicializa una nueva instancia de Ronda con una lista vacía de apuestas.

#### Métodos

**public List<Apuesta> getApuesta ():**

Devuelve la lista de apuestas realizadas durante la ronda.

**public List<Apuesta> getApuestaGanadora (String numero):**

Devuelve una lista de apuestas que resultaron ganadoras para un número específico de la ruleta.

**numero:** El número de la ruleta para el cual se buscan las apuestas ganadoras.

**public void addApuesta (Apuesta apuesta):**

Agrega una nueva apuesta a la lista de apuestas de la ronda. Si un jugador ya ha realizado una apuesta similar en la misma posición, se incrementa el contador de fichas de esa apuesta en lugar de agregar una nueva apuesta.

**public void limpiarApuesta ():**

Elimina las apuestas no ganadoras de la lista de apuestas de la ronda y resta las fichas de los jugadores correspondientes.

**public List<Apuesta> getApuesta (Jugador jugador):**

Devuelve una lista de apuestas realizadas por un jugador específico durante la ronda.

**jugador:** El jugador para el cual se buscan las apuestas realizadas.

## -Ruleta

La clase **Ruleta** representa la ruleta del juego de la ruleta. Contiene segmentos que representan los números y colores en la ruleta, así como métodos para obtener información sobre los números y colores específicos

### Atributos

**rojo (Color):** Color rojo utilizado en la ruleta.

**verde (Color):** Color verde utilizado en la ruleta.

**segmentoRuleta (SegmentoRuleta []):** Un array de objetos **SegmentoRuleta** que representan los segmentos de la ruleta.

### Constructor

**public Ruleta ():**

Inicializa una nueva instancia de Ruleta con los colores definidos y los segmentos de la ruleta

### Métodos

**public int length ():**

Devuelve la cantidad de segmentos en la ruleta.

**public SegmentoRuleta getSegmento (int index):**

Devuelve el segmento de la ruleta en el índice especificado.

**index:** El índice del segmento que se desea obtener.

**public SegmentoRuleta [] getSegmentoRuleta ():**

Devuelve todos los segmentos de la ruleta como un array.

**public String [] getRangoNumero (int start, int fin):**

Devuelve un rango de números como un array de cadenas de caracteres, desde start hasta fin.

**start:** El número inicial del rango.

**fin:** El número final del rango.

**public String [] getImpares ():**

Devuelve un array de cadenas de caracteres que contiene todos los números impares en la ruleta.

**public String [] getPares ():**

Devuelve un array de cadenas de caracteres que contiene todos los números pares en la ruleta.

**public String [] getNumerosRojos ():**

Devuelve un array de cadenas de caracteres que contiene todos los números rojos en la ruleta.

**public String [] getNumeroNegro ():**

Devuelve un array de cadenas de caracteres que contiene todos los números negros en la ruleta.

**public Color getRojo ():**

Devuelve el color rojo utilizado en la ruleta.

**public Color getVerde ():**

Devuelve el color verde utilizado en la ruleta.

**public Color getBackgroundColor (int numero):**

Devuelve el color de fondo correspondiente al número especificado en la ruleta.

**numero:** El número para el cual se desea obtener el color de fondo.

## -RuletaModelo

La clase **RuletaModelo** representa el modelo del juego de la ruleta. Contiene la lógica de negocio para gestionar las apuestas de los jugadores, así como información sobre los jugadores, fichas, ruleta y rondas del juego.

### Atributos

**ApuestaMinima (int):** El monto mínimo requerido para realizar una apuesta.

**ApuestaMaximaInterna (int):** El monto máximo permitido para una apuesta interna.

**ApuestaMaximaExterna (int):** El monto máximo permitido para una apuesta externa.

**valorFichas (ValorFichas):** Instancia para manejar los valores de las fichas.

**apuestasClicables (ApuestasClicables):** Instancia para manejar las apuestas clicables en la interfaz.

**jugador (List<Jugador>):** Lista de jugadores en el juego.

**JugadorSeleccionado (Jugador):** Jugador seleccionado en el juego.

**random (Random):** Generador de números aleatorios.

**ronda (Ronda):** Instancia para manejar las rondas del juego.

**fichasRuleta (FichasRuleta):** Instancia para manejar las fichas de la ruleta.

**ruleta (Ruleta):** Instancia que representa la ruleta del juego

### Constructor

**public RuletaModelo ():**

Inicializa una nueva instancia de RuletaModelo con los valores predeterminados y las instancias necesarias.

### Métodos

**public ApuestaClicable contieneApuestaClicable (Point point):**

Verifica si hay una apuesta clicable en las coordenadas especificadas y devuelve la apuesta si se encuentra alguna.

**point:** Punto de coordenadas donde se realiza la verificación.

**public void addJugador (Jugador jugador):**

Agrega un jugador a la lista de jugadores.

**jugador:** Jugador que se va a agregar.

**public void removeJugador (Jugador jugador):**

Elimina un jugador de la lista de jugadores y devuelve sus fichas a la lista de fichas de la ruleta.

**jugador:** Jugador que se va a eliminar.

**public Jugador getJugador (String nombre):**

Devuelve un jugador con el nombre especificado.

**nombre:** Nombre del jugador que se desea obtener.

**public JugadorApuestaError ApuestaValida ():**

Verifica si las apuestas realizadas son válidas y devuelve un error si no lo son.

Retorna una instancia de JugadorApuestaError con un código de error que indica el estado de las apuestas.

**public Jugador getEleccionJugador ():**

Devuelve el jugador seleccionado en el juego.

**public void setJugadorSeleccionado (Jugador jugadorSeleccionado):**

Establece el jugador seleccionado en el juego.

**jugadorSeleccionado:** Jugador seleccionado que se va a establecer.

**public void addFichaRuleta (FichaRuleta fichaRuleta):**

Agrega una ficha de ruleta a la lista de fichas de la ruleta.

**fichaRuleta:** Ficha de ruleta que se va a agregar.

**public void EliminarFicha (FichaRuleta fichaRuleta):**

Elimina una ficha de ruleta de la lista de fichas de la ruleta.

**fichaRuleta:** Ficha de ruleta que se va a eliminar.

**public List<Apuesta> getApuesta ():**

Devuelve la lista de apuestas realizadas en la ronda actual.

**public List<Apuesta> getGanancias (String number):**

Devuelve las apuestas ganadoras para un número específico en la ronda actual.

**number:** Número para el cual se desean obtener las apuestas ganadoras.

**public void addApuesta (Apuesta apuesta):**

Agrega una apuesta a la ronda actual.

**apuesta:** Apuesta que se va a agregar.

**public void LimpiarApuesta ():**

Limpia las apuestas realizadas en la ronda actual.

## -SegmentoRuleta

La clase **SegmentoRuleta** representa un segmento individual en la ruleta. Cada segmento tiene un color de fondo, una ubicación relativa y un número asociado en la ruleta.

### Atributos

**colorFondo (Color):** El color de fondo del segmento.

**Delta (Point):** El desplazamiento relativo del segmento con respecto al centro de la ruleta.

**NumeroRuleta (String):** El número asociado al segmento en la ruleta.

### Constructor

**public SegmentoRuleta (String NumeroRuleta, Color colorFondo, Point Delta):**

Construye un nuevo segmento de ruleta con el número asociado, el color de fondo y el desplazamiento relativo especificados.

### Métodos

**public Color getColorFondo ():**

Devuelve el color de fondo del segmento.

**public Point getDelta ():**

Devuelve el desplazamiento relativo del segmento.

**public String getNumeroRuleta ():**

Devuelve el número asociado al segmento en la ruleta.

## -TablaConConstantesDibujables

Es una clase que define un conjunto de constantes para controlar el diseño y la disposición de un tablero o tabla dentro de una aplicación gráfica.

### Atributos

**AnchuraSegmento:** la anchura de cada segmento en el tablero.

**margen:** el margen general utilizado en el diseño del tablero.

**lineaMargen:** el margen entre las líneas del tablero.

**imagenMargen:** el margen alrededor de las imágenes en el tablero.

### Constructor



**TablaDibujoConstantes ()**: es el constructor de la clase que inicializa los valores de las constantes. Estos valores son fijos y se establecen en el momento de la creación de un objeto de esta clase.

**imagenMargen** se inicializa con 10.

**margen** se inicializa con 30.

**lineaMargen** se inicializa con 3.

**AnchuraSegmento** se inicializa con 64.

### Métodos

**public int getSegmentoAncho ()**: devuelve el valor de **AnchuraSegmento**.

**public int getMargen ()**: devuelve el valor de **margen**.

**public int getLineaMargen ()**: devuelve el valor de **lineaMargen**.

**public int getImagenMargen ()**: devuelve el valor de **imagenMargen**

### -TipoApuesta

El enum **TipoApuesta** define los diferentes tipos de apuestas que se pueden realizar en el juego de la ruleta. Cada tipo de apuesta tiene asociado un pago, una indicación de si es una apuesta interna o externa, y una descripción que explica brevemente qué representa esa apuesta.

### Valores de ENUM:

```
NUMERO( pago: 36, ApuestaInterna: true, descripcion: "Numero a apostar"),
5 usages
APUESTA_DIVIDIDA( pago: 18, ApuestaInterna: true, descripcion: "Apostar entre 2 numeros"),
4 usages
TRES_VALORES_00( pago: 12, ApuestaInterna: true, descripcion: "00, 1, 2 combinacion"),
4 usages
TRES_VALORES_0( pago: 12, ApuestaInterna: true, descripcion: "0, 2, 3 combinacion"),
4 usages
APUESTA_CALLE( pago: 12, ApuestaInterna: true, descripcion: "Apuesta por cualquier fila de tres números"),
4 usages
CUATRO_ESQUINAS( pago: 9, ApuestaInterna: true, descripcion: "Apuesta 4 numeros de las esquinas"),
4 usages
APUESTA_LINEA( pago: 6, ApuestaInterna: true, descripcion: "Cualquier par de una línea"),
6 usages
APUESTA_COLUMNA( pago: 3, ApuestaInterna: false, descripcion: "Apostar la columna"),
4 usages
PRIMEROS12( pago: 3, ApuestaInterna: false, descripcion: "Los numeros del 1 al 12"),
4 usages
SEGUNDOS12( pago: 3, ApuestaInterna: false, descripcion: "Los numeros del 13 al 24"),
4 usages
TERCEROS12( pago: 3, ApuestaInterna: false, descripcion: "Los numeros del 25 al 36"),
4 usages
PRIMEROS18( pago: 2, ApuestaInterna: false, descripcion: "Los numeros del 1 al 18"),
4 usages
SEGUNDOS18( pago: 2, ApuestaInterna: false, descripcion: "Los numeros del 19 al 36"),
4 usages
IMPARES( pago: 2, ApuestaInterna: false, descripcion: "Apuesta a impares"),
7 usages
PARES( pago: 2, ApuestaInterna: false, descripcion: "Apuesta a pares"),
4 usages
ROJO( pago: 2, ApuestaInterna: false, descripcion: "Apuesta a rojos"),
4 usages
NEGRO( pago: 2, ApuestaInterna: false, descripcion: "Apuesta a negros");
```

### Atributos

**pago (int):** El pago asociado a la apuesta.

**ApuestaInterna (boolean):** Indica si la apuesta es interna (true) o externa (false).

**descripcion (String):** Una breve descripción de la apuesta.

### Métodos

**public int getPago ():**

Devuelve el pago asociado a la apuesta.

**public boolean isApuestaInterna ():**

Indica si la apuesta es interna (true) o externa (false).

**public String getDescripcion ():**

Devuelve la descripción de la apuesta.

### -ValorFicha

La clase **ValorFicha** representa el valor monetario de una ficha utilizada en el juego de la ruleta.

#### Atributos

**valor (int):** El valor numérico de la ficha

#### Constructor

**public ValorFicha (int valor):**

Este constructor crea una nueva instancia de ValorFicha con el valor especificado.

#### Métodos

**public int getValor ():**

Devuelve el valor numérico de la ficha.

**@Override public String toString ():**

Devuelve una representación en cadena del valor de la ficha, formateada como un valor monetario en dólares.

### -ValorFichas

La clase **ValorFichas** representa una colección de valores de fichas utilizadas en el juego de la ruleta. Permite definir y obtener los diferentes valores de fichas disponibles en la aplicación.

#### Atributos

**valores (List<ValorFicha>):** Una lista que almacena los diferentes valores de fichas disponibles en la aplicación.

#### Constructor

**public ValorFichas ():**

Este constructor inicializa la lista de valores de fichas con valores predeterminados: \$1, \$5 y \$25.

#### Métodos

**public List<ValorFicha> getValores ():**

Devuelve la lista de valores de fichas disponibles en la aplicación.

## 3.2. Controller

### -AñadirJugadorListener

Este código gestiona la adición de nuevos jugadores en la aplicación de ruleta, asegurándose de que los datos ingresados sean válidos (por ejemplo, que la

cantidad sea un número y que el nombre del jugador no esté duplicado). Si todo es válido, crea un nuevo jugador con las características seleccionadas y lo añade al modelo, actualizando la interfaz de usuario en consecuencia

### Atributos

**dialog (AñadirDialogoJugador):** El diálogo desde el cual se añade el jugador.

**frame (RuletaFrame):** El marco principal de la aplicación.

**model (RuletaModelo):** El modelo que gestiona la lógica de la ruleta.

### Constructor

**public AñadirJugadorListener (RuletaFrame frame, RuletaModelo model, AñadirDialogoJugador dialog)**

Este constructor inicializa los atributos frame, model, y dialog con los valores proporcionados.

### Métodos

**public void actionPerformed (ActionEvent evento)**

Este método maneja el evento de añadir un nuevo jugador cuando se hace clic en el botón "OK". Realiza las siguientes acciones:

**Obtener el Panel de Añadir Jugador:** Obtiene el panel desde el diálogo para acceder a los campos de entrada del usuario.

**Validar Cantidad de Compra:** Obtiene el texto del campo de compra y lo convierte a un double.

Si la conversión falla, marca el campo de texto en rojo y muestra un mensaje de error.

Si la conversión tiene éxito, restablece los colores y el mensaje de error.

**Validar Nombre del Jugador:** Obtiene el nombre del jugador y verifica si ya existe en el modelo.

Si el nombre ya está en uso, marca el campo de texto en rojo y muestra un mensaje de error.

Si no está en uso, restablece los colores y el mensaje de error.

**Configurar y Añadir el Jugador:**

Obtiene las selecciones de la ficha y el valor de la ficha del combo box.

Elimina la ficha seleccionada del modelo para evitar duplicados.

Crea un nuevo jugador con el nombre proporcionado, asigna las fichas y la imagen de la ficha.

Añade el nuevo jugador al modelo.

Actualiza el panel de jugadores en la interfaz de usuario.

**Cierra el diálogo.**

**double valueOf (String number):**

Este método convierte una cadena de texto en un número double. Si la conversión falla, devuelve Double.MIN\_VALUE.

### **-EliminarJugadorListener**

La clase **EliminarJugadorListener** maneja el evento de eliminación de un jugador de la ruleta. Se encarga de eliminar un jugador seleccionado del modelo y actualizar la interfaz de usuario en consecuencia.

#### Atributos

**frame (RuletaFrame):** El marco principal de la aplicación.

**model (RuletaModelo):** El modelo que gestiona la lógica de la ruleta.

**dialog (DialogoPanelEliminarJugador):** El diálogo desde el cual se realiza la eliminación del jugador.

#### Constructor

**EliminarJugadorListener (RuletaFrame frame, RuletaModelo model, DialogoPanelEliminarJugador dialog):**

Este constructor inicializa los atributos frame, model, y dialog con los valores proporcionados.

#### Métodos

**void actionPerformed (ActionEvent event):**

Este método maneja el evento de eliminación de un jugador cuando se hace clic en el botón correspondiente. Realiza las siguientes acciones:

**Obtener el Jugador Seleccionado:** Obtiene el jugador seleccionado actualmente del modelo.

**Desseleccionar el Jugador:** Establece el jugador seleccionado en null en el modelo.

**Eliminar el Jugador:** Elimina el jugador del modelo.

**Actualizar la Interfaz de Usuario:** Llama al método updatePanelJugador en el frame para actualizar la lista de jugadores mostrada.

**Cerrar el Diálogo:** Cierra el diálogo de eliminación de jugador.

## -FichaListener

Esta clase es un **MouseListener** que escucha los eventos del mouse relacionados con las fichas en el tablero de apuestas. Sus principales funciones son agregar una apuesta cuando se hace clic con el botón izquierdo del mouse y eliminar una apuesta cuando se hace clic con el botón derecho del mouse.

### Atributos:

**frame:** Una instancia de RuletaFrame, que representa el marco principal de la aplicación.

**model:** Una instancia de RuletaModelo, que contiene la lógica del juego y almacena los datos del juego.

### Constructor:

**FichaListener (RuletaFrame frame, RuletaModelo model):** El constructor que inicializa los atributos de la clase con las instancias proporcionadas.

### Métodos

#### **mouseReleased (MouseEvent event):**

Este método se llama cuando se libera un botón del mouse después de un clic. Comprueba qué botón se soltó y ejecuta la acción correspondiente.

Si se suelta el botón izquierdo del mouse (MouseEvent.BUTTON1), llama al método **addApuesta** para agregar una apuesta.

Si se suelta el botón derecho del mouse (MouseEvent.BUTTON3), llama al método **eliminarApuesta** para eliminar una apuesta.

#### **addApuesta (Jugador jugador, Point point):**

Este método agrega una apuesta al jugador en el punto especificado en el tablero de apuestas.

Primero, verifica si hay una **apuesta clicable** en el punto especificado. Si la encuentra, crea una nueva apuesta con los detalles proporcionados por el **ApuestaClicable** y la agrega al modelo.

Luego, actualiza la tabla de apuestas en el marco principal.

#### **eliminarApuesta (Jugador jugador, Point point):**

Este método elimina una apuesta del jugador en el punto especificado en el tablero de apuestas.

Similar a **addApuesta**, busca una **apuesta clicable** en el punto especificado. Si la encuentra, busca en la lista de apuestas del modelo aquellas que coincidan con el **jugador y la ubicación** de la apuesta clicable.

Si encuentra una apuesta que coincida, la elimina de la lista de apuestas. Si la apuesta tiene más de una ficha, simplemente decrementa el contador de fichas; de lo contrario, elimina la apuesta por completo.

Finalmente, actualiza la tabla de apuestas en el marco principal.

### -RuletaListener

Esta clase implementa la interfaz **ActionListener** y se encarga de manejar los eventos relacionados con la ruleta del juego de la aplicación.

#### Atributos:

**index:** Un entero que representa el índice del segmento de la ruleta seleccionado.

**frame:** Una instancia de RuletaFrame, que representa el marco principal de la aplicación.

**model:** Una instancia de RuletaModelo, que contiene la lógica del juego y almacena los datos del juego.

**timer:** Un temporizador utilizado para controlar la rotación de la ruleta.

**SegmentoRuleta:** Un objeto SegmentoRuleta que representa el segmento de la ruleta seleccionado.

#### Constructor:

**RuletaListener (RuletaFrame frame, RuletaModelo model):** El constructor que inicializa los atributos de la clase con las instancias proporcionadas.

#### Métodos

##### **actionPerformed (ActionEvent event):**

Este método se llama cuando ocurre un evento de acción, como hacer clic en un botón.

Comprueba si la apuesta realizada por los jugadores es válida utilizando un objeto **JugadorApuestaError**. Dependiendo del código de error devuelto, muestra un mensaje de error correspondiente.

Si la apuesta es válida, genera un índice aleatorio para seleccionar un segmento de la ruleta y comienza la rotación de la ruleta utilizando un temporizador.

Cuando se completa la rotación, muestra el resultado de la ruleta y calcula las ganancias de los jugadores, actualizando sus fichas y la tabla de apuestas en consecuencia.

### Clase interna RuletasListener:

Esta clase implementa la interfaz **ActionListener** y se utiliza para controlar la rotación de la ruleta.

Al inicializar, genera un contador y un límite de rotación aleatorio.

En cada acción, rota la ruleta y comprueba si se ha alcanzado el límite de rotación. Cuando se alcanza el límite, detiene la rotación, muestra el resultado de la ruleta y calcula las ganancias de los jugadores.

### -SeleccionJugadorListener

Esta clase implementa la interfaz **ListSelectionListener** y se encarga de manejar los eventos de selección en la tabla de jugadores de la aplicación.

#### Atributos:

**frame:** Una instancia de **RuletaFrame**, que representa el marco principal de la aplicación.

**model:** Una instancia de **RuletaModelo**, que contiene la lógica del juego y almacena los datos del juego.

#### Constructor:

**SeleccionJugadorListener (RuletaFrame frame, RuletaModelo model):** El constructor que inicializa los atributos de la clase con las instancias proporcionadas.

#### Métodos

**valueChanged (ListSelectionEvent evento):** Este método se llama cuando cambia la selección en la tabla de jugadores.

Obtiene el modelo de selección (**ListSelectionModel**) y la tabla de jugadores del evento.

Verifica si la selección ha terminado y obtiene el índice de la fila seleccionada.

Si se ha seleccionado una fila, obtiene el nombre del jugador seleccionado y actualiza el jugador seleccionado en el modelo.

Si no se ha seleccionado ninguna fila, establece el jugador seleccionado en null en el modelo.

## 3.3. Diálogos

### -AñadirJugadorDialogo

Esta clase representa un diálogo para añadir un nuevo jugador a la aplicación.

#### Atributos:

**AñadirPanelJugador:** Una instancia de **AñadirPanelJugador**, que contiene los campos para ingresar los detalles del nuevo jugador.



#### Constructor:

**AñadirJugadorDialogo (RuletaFrame frame, RuletaModelo modelo, String título):**

El constructor que crea un nuevo diálogo de añadir jugador. Toma un objeto **RuletaFrame** para la ventana principal, un objeto **RuletaModelo** para el modelo de la ruleta y un título para el diálogo.

#### Métodos

**createPanelPrincipal (RuletaFrame frame, RuletaModelo modelo):** Este método crea el panel principal del diálogo.

Agrega el panel **AñadirPanelJugador** al centro del panel principal.

Crea un panel interno para los botones "OK" y "Cancelar".

Agrega un botón "OK" que, al hacer clic, llama a un **AñadirJugadorListener** para procesar la adición del jugador.

Agrega un botón "Cancelar" que, al hacer clic, cierra el diálogo.

**getAñadirPanelJugador ():**

Este método devuelve la instancia de **AñadirPanelJugador**.

#### -AñadirPanelJugador

Esta clase representa el panel para añadir un nuevo jugador en la interfaz de usuario de texto.

#### Atributos:

**modelo:** El modelo de la ruleta al que pertenece este panel.

**panel:** El panel principal que contiene todos los elementos de la interfaz para añadir un jugador.

#### Constructor:

**AñadirPanelJugador (RuletaModelo modelo):** El constructor que recibe el modelo de la ruleta y crea el panel principal.

#### Métodos

**createPanelPrincipal ():**

Este método crea el panel principal del panel de añadir jugador.

Define los componentes necesarios, como etiquetas, campos de texto y listas desplegables para ingresar la información del jugador.

Utiliza un diseño de cuadrícula (**GridBagLayout**) para organizar los componentes en el panel.

**getPanel ():** Devuelve el panel principal creado.

Métodos de acceso:

**getValorFichaComboBox ():** Devuelve el combobox para seleccionar el valor de la ficha.

**getImagenFichaComboBox ():** Devuelve el combobox para seleccionar la imagen de la ficha.

**getCompraCantidad ():** Devuelve el campo de texto para ingresar la cantidad de fichas a comprar.

**getCampoNombre ():** Devuelve el campo de texto para ingresar el nombre del jugador.

**getMensajeLabel ():** Devuelve la etiqueta para mostrar mensajes al usuario.

**Clase interna FichaCellRenderer:**

Esta clase implementa **ListCellRenderer** para personalizar la visualización de los elementos en el combobox de imagen de ficha.

Utiliza una etiqueta para mostrar la imagen de la ficha en el combobox.

**-DialogoApuestas**

Esta clase representa un diálogo que muestra las apuestas realizadas en la ruleta.

Constructor:

**DialogoApuestas (RuletaFrame frame, String título):** El constructor recibe un objeto **RuletaFrame** y un título para el diálogo. Crea el diálogo y establece su título, luego agrega el panel principal y lo muestra.

Métodos

**createPanelPrincipal ():**

Este método crea y devuelve el panel principal del diálogo.

El panel principal contiene un panel de apuestas (PanelApuestas) en el centro y un botón "OK" en la parte inferior.

El botón "OK" cierra el diálogo cuando se presiona.

**-DialogoGanancias**

Esta clase representa un diálogo que muestra las ganancias obtenidas en la ruleta.

Constructor:

**DialogoGanancias (RuletaFrame frame, List<Apuesta> ganadores, String numero, String título):** El constructor recibe un objeto RuletaFrame, una lista de objetos Apuesta que representan a los ganadores, un número específico y

un título para el diálogo. Crea el diálogo y establece su título, luego agrega el panel de ganancias y un botón "OK" para cerrar el diálogo.

#### Métodos

##### **createPanelPrincipal ():**

Este método crea y devuelve el panel principal del diálogo.

El panel principal contiene un panel de ganancias (PanelGanancias) en el centro y un botón "OK" en la parte inferior.

El botón "OK" cierra el diálogo cuando se presiona.

#### **-DialogoPanelEliminarJugador**

Esta clase representa un diálogo que permite al usuario eliminar un jugador de la ruleta.

#### Constructor:

**DialogoPanelEliminarJugador (RuletaFrame frame, RuletaModelo modelo, String título):** El constructor recibe un objeto RuletaFrame, un objeto RuletaModelo, y un título para el diálogo. Crea el diálogo y establece su título, luego agrega el panel para eliminar jugador (PanelEliminarJugador) y dos botones: "OK" para confirmar la eliminación y "Cancelar" para cerrar el diálogo sin realizar ninguna acción.

#### Métodos

##### **createMainPanel ():**

Este método crea y devuelve el panel principal del diálogo.

El panel principal contiene el panel para eliminar jugador en el centro y dos botones: "OK" y "Cancelar" en la parte inferior.

El botón "OK" ejecuta la acción de eliminar jugador al hacer clic.

El botón "Cancelar" cierra el diálogo sin realizar ninguna acción.

#### **-InstruccionesDialogo**

Esta clase representa un diálogo que muestra instrucciones o información sobre cómo jugar a la ruleta.

#### Constructor:

**InstruccionesDialogo (RuletaFrame Vista, String Titulo):** El constructor recibe un objeto RuletaFrame, que es la vista principal de la aplicación, y un título para el diálogo. Crea el diálogo con un diseño BorderLayout y lo hace modal, lo que significa que bloquea la interacción con la ventana principal hasta que se cierre el diálogo. Llama a dos métodos para crear el panel principal y el panel de botones.

#### Métodos

### **createPanelPrincipal ():**

Este método crea y devuelve el panel principal del diálogo.

El panel contiene un JEditorPane que muestra el contenido HTML de un archivo de instrucciones. El archivo se carga desde la ruta **"/instrucciones.html" en el directorio de recursos del proyecto.**

El contenido HTML no es editable y se muestra en un JScrollPane para permitir el desplazamiento si es necesario.

### **Método createButtonPanel ():**

Este método crea y devuelve el panel de botones del diálogo.

El panel contiene un botón "Ok" que, al hacer clic, cierra el diálogo.

### **Clase interna CancelarAccion:**

Esta clase interna extiende AbstractAction y representa la acción de cerrar el diálogo cuando se hace clic en el botón "Ok".

Al implementar actionPerformed, simplemente llama al método dispose () para cerrar el diálogo.

### **-PanelApuestas**

Esta clase representa un panel que muestra los diferentes tipos de apuestas disponibles en el juego de la ruleta, junto con sus correspondientes pagos.

#### Constructor:

**PanelApuestas ():** El constructor crea el panel de apuestas llamando al método createPanelApuesta ().

#### Métodos

##### **createPanelApuesta ():**

Este método crea y devuelve el panel que muestra los tipos de apuestas y sus pagos.

Utiliza un diseño de cuadrícula (GridBagLayout) para organizar los elementos en filas y columnas.

Cada fila contiene dos etiquetas: una para el tipo de apuesta y otra para el pago asociado.

El panel tiene un borde vacío con márgenes de 5 píxeles en todos los lados.

Define dos fuentes diferentes: una para el título ("Apuestas") y otra para los tipos de apuestas y sus pagos.

Itera sobre los valores del **enum TipoApuesta**, obteniendo la descripción de cada tipo de apuesta y su pago asociado.

Crea una etiqueta para cada tipo de apuesta y otra para su pago, y las agrega al panel en la posición correspondiente en la cuadrícula.

#### **Método `getPanel ()`:**

Este método devuelve el panel creado por `createPanelApuesta ()`.

#### **-PanelEliminarJugador**

Esta clase representa un panel que se utiliza para eliminar un jugador y retirar su dinero de la aplicación de la ruleta.

#### Constructor:

**PanelEliminarJugador (RuletaFrame frame, RuletaModelo modelo):** El constructor recibe una instancia de RuletaFrame y RuletaModelo. Verifica si se ha seleccionado un jugador. Si no se ha seleccionado ningún jugador, muestra un mensaje de error y no crea el panel.

#### Métodos

##### **createPanelPrincipal ():**

Este método crea y devuelve el panel principal para eliminar un jugador y retirar su dinero.

Utiliza un diseño de cuadrícula (GridBagLayout) para organizar los elementos en filas y columnas.

Crea etiquetas y campos de texto para mostrar el nombre del jugador y la cantidad de dinero que se retirará.

El nombre del jugador y la cantidad de dinero se obtienen del modelo.

Los campos de texto son de solo lectura (`setEditable(false)`) para mostrar la información sin permitir su modificación.

El panel tiene un borde vacío con márgenes de 5 píxeles en todos los lados.

Define una fuente para el título del panel ("Retirar Dinero").

#### **Método `getPanel ()`:**

Este método devuelve el panel creado por `createPanelPrincipal ()`.

#### **-PanelGanancias**

Esta clase representa un panel que muestra las apuestas ganadoras en la ruleta, incluyendo el número ganador y las apuestas realizadas por los jugadores que resultaron ganadores.

#### Constructor:

**PanelGanancias (List<Apuesta> ganancias, String numero):** El constructor recibe una lista de apuestas ganadoras (ganancias) y el número ganador

(numero). Utiliza esta información para crear el panel que muestra las apuestas ganadoras.

### Métodos

#### **createPanelPrincipal ():**

Este método crea y devuelve el panel principal para mostrar las apuestas ganadoras.

Utiliza un diseño de cuadrícula (GridBagLayout) para organizar los elementos en filas y columnas.

Crea etiquetas para mostrar el título del panel ("Apuestas Ganadoras"), el número ganador y los encabezados de las columnas ("Nombre", "Tipo Apuesta", "Apuesta").

Para cada apuesta ganadora en la lista, crea etiquetas con el nombre del jugador, el tipo de apuesta y la cantidad de pago.

El panel tiene un borde vacío con márgenes de 5 píxeles en todos los lados.

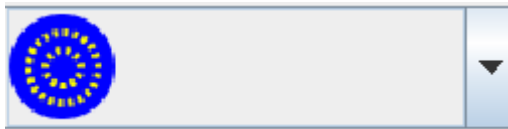
Define fuentes para el título del panel ("Apuestas Ganadoras") y los encabezados de las columnas.

#### **getPanel ():**

Este método devuelve el panel creado por createPanelPrincipal ().

## 4. Interfaz Gráfica

### -ImagenFicha



La clase ImagenFicha en el paquete RuletaApp.view es responsable de crear una imagen que representa una ficha de ruleta, utilizando colores específicos para el exterior y el interior de la ficha.

### Atributos

**radio:** Define el radio de la ficha.

**imagenFicha:** Un BufferedImage que contiene la imagen de la ficha.

**colorFicha:** Color exterior de la ficha.

**colorInterior:** Color interior de la ficha.

### Constructor:

**ImagenFicha(Color colorFicha, Color colorInterior):** Inicializa los colores y el radio de la ficha, y crea la imagen de la ficha llamando al método `creatImagenFicha()`.

#### Métodos

##### **creatImagenFicha():**

Crea una imagen circular con el color exterior y dos círculos concéntricos con el color interior.

Utiliza `Graphics2D` para dibujar la ficha, configurando el antialiasing y las métricas fraccionarias para mejorar la calidad del dibujo.

La ficha tiene un radio de 21 píxeles, y la imagen tiene un fondo transparente.

Dibuja un círculo exterior y dos círculos interiores con diferentes radios y estilos de trazado.

**getImagenFicha():** Devuelve la imagen de la ficha (`BufferedImage`).

**getColorFicha():** Devuelve el color exterior de la ficha.

#### -ImagenTablaRuleta

00	0	3	6	9	12	15	18	21	24	27	30	33	36	3 para 13 para 1
	2	5	8	11	14	17	20	23	26	29	32	35		3 para 13 para 1
	1	4	7	10	13	16	19	22	25	28	31	34		3 para 13 para 1
1 a 12					13 a 24					25 a 36				
1 a 18		PAR		♦		♦		IMPAR		19 a 36				

La clase `ImagenTablaRuleta` en el paquete `RuletaApp.view` es responsable de generar la imagen de la tabla de ruleta, incluyendo tanto los números como las áreas de apuestas externas.

#### Atributos:

**Ancho, Alto, AnchuraSegmento, margen, lineaMargen:** Son parámetros relacionados con el tamaño de la imagen y el diseño de la tabla.

**imagen:** Es la imagen de la tabla de ruleta que se crea.

**ColorVerde:** Color de fondo de la tabla.

**ruleta:** Instancia de la clase `Ruleta` que maneja la lógica de la ruleta.

#### Constructor:

Inicializa los parámetros de diseño (**margen**, **lineaMargen**, **AnchuraSegmento**) usando una instancia de **TablaConConstantesDibujables**.

Calcula Ancho y Alto de la imagen en base a estos parámetros.

Inicializa el color de fondo y la instancia de Ruleta.

Crea la imagen de la tabla llamando a **createTablaRuleta**.

### Métodos

#### **createTablaRuleta:**

Crea una imagen en blanco y configura los Graphics2D para suavizado de bordes y métrica fraccionada.

Pinta el fondo verde y llama a métodos para dibujar los números y líneas.

Establece un borde negro alrededor de la imagen.

Rota la imagen 90 grados en sentido antihorario y llama a **PintarApuestaExterna** para dibujar las apuestas externas.

Devuelve la imagen rotada.

#### **PintarNumeros:**

Dibuja los números de la ruleta y sus fondos de color.

Usa **PintarNumero** para dibujar cada número en su posición correspondiente.

#### **PintarNumero:**

Calcula la posición y dibuja un número en su celda correspondiente con el color de fondo adecuado.

#### **PintarApuestaExterna:**

Dibuja las áreas de apuestas externas (1 a 12, 13 a 24, etc.).

#### **Métodos auxiliares:**

**pintarTexto:** Dibuja texto centrado en un segmento.

**drawDiamond:** Dibuja un diamante para las apuestas de color rojo/negro.

**pintarLineasHorizontales y pintarLineaVertical:** Dibuja las líneas horizontales y verticales de la tabla.




**PintarSegmentosLineaHorizontal y PintarSegmentosLineaVertical:** Dibuja segmentos específicos de líneas.

**RotarImagen:** Rota la imagen 90 grados en sentido antihorario.

**getImagen:** Devuelve la imagen de la tabla de ruleta generada



## -JugadorPanel

Nombre Jugador	Ficha jugador	Cantidad	Valor Ficha	Cantidad actual	Numero de fichas
Jugador 1		\$40.00	\$5.00	\$40.00	8
Jugador 2		\$55.00	\$5.00	\$55.00	11
Jugador 3		\$25.00	\$5.00	\$25.00	5

La clase **JugadorPanel** es responsable de mostrar una tabla con información sobre los jugadores en una aplicación de ruleta. Utiliza JTable para mostrar datos como el nombre del jugador, su imagen de ficha, cantidad comprada, valor de la ficha, balance actual y número de fichas.

### Atributos

**ModeloTablaJugador:** Clase interna que extiende DefaultTableModel para definir el modelo de datos de la tabla.

**Renderizadores Personalizados:** Clases internas que implementan TableCellRenderer para personalizar la representación de las celdas de la tabla.

**ImagenRenderer:** Renderiza una celda con una imagen.

**CantidadRenderer:** Renderiza una celda con cantidades en formato de moneda.

**ContadorRenderer:** Renderiza una celda con un contador.

**StringRenderer:** Renderiza una celda con texto.

### Constructor

**JugadorPanel(RuletaFrame frame, RuletaModelo modelo):** Constructor que inicializa el panel y la tabla.

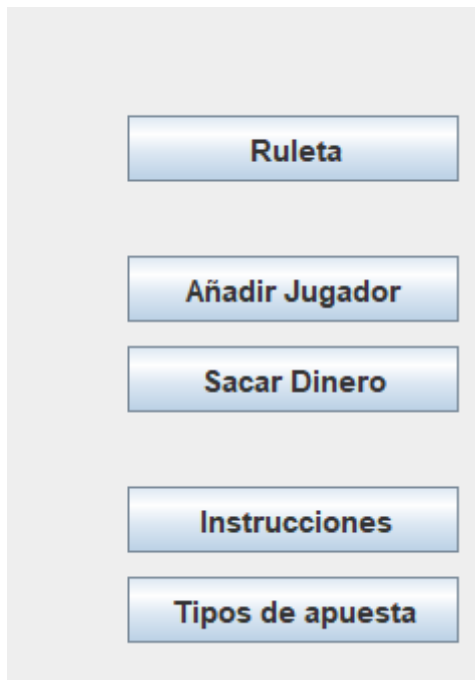
### Métodos

**createPanelJugador():** Configura y devuelve el panel principal que contiene la tabla.

**ActualizarModeloTabla():** Actualiza el modelo de la tabla con los datos actuales de los jugadores.

**addJugador(Jugador jugador):** Añade un jugador al modelo de la tabla.

## -MenuPanel



La clase MenuPanel es un componente visual que presenta un conjunto de botones para realizar diversas acciones dentro de la aplicación. Utiliza un GridBagLayout para organizar los botones dentro del panel.

### Atributos

**RuletaFrame y RuletaModelo:** Referencias al marco principal de la aplicación y al modelo de datos, respectivamente.

**JPanel panel:** Panel principal que contiene los botones del menú.

### Constructor

**MenuPanel(RuletaFrame frame, RuletaModelo modelo):** Constructor que inicializa el panel de menú.

### Métodos

**createMenuPanel():** Configura y devuelve el panel principal con los botones.

**getPanel():** Devuelve el panel principal para que pueda ser añadido a la interfaz gráfica principal.

### Botones y Funcionalidades

#### **Ruleta:**

Inicia el juego de ruleta.

Acción manejada por RuletaListener.

#### **Añadir Jugador:**

Permite añadir un nuevo jugador al juego.

Muestra un diálogo de error si ya hay 8 jugadores.

Acción manejada por un ActionListener anónimo que crea un nuevo AñadirJugadorDialogo.

### Sacar Dinero:

Permite sacar el dinero de un jugador y eliminarlo

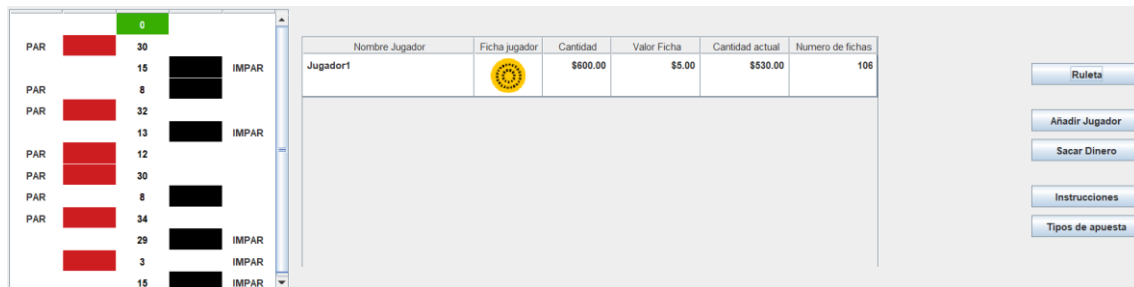
### Instrucciones

Permite ver las instrucciones escritas en un html llamado instrucciones.html

### Tipos de apuesta

Permite ver un panel que muestra todos los tipos de apuesta que hay y sus respectivas ganancias

## -PanelDeControl



El código de la clase **PanelDeControl** se encarga de crear y organizar los diferentes componentes visuales del panel de control de la aplicación de ruleta. Esta clase actúa como un contenedor que agrupa el menú, el panel de jugadores y el panel de llamadas, y proporciona métodos para interactuar con estos subcomponentes.

La clase PanelDeControl organiza y gestiona **tres subpaneles** principales:

**PanelDeLlamadas:** Muestra las llamadas realizadas.

**MenuPanel:** Contiene los botones de menú para realizar diversas acciones en la aplicación.

**JugadorPanel:** Muestra información sobre los jugadores.

### Atributos

**PanelDeLlamadas panelDeLlamadas:** Subpanel que gestiona y muestra las llamadas.

**MenuPanel menuPanel:** Subpanel que contiene los botones de menú.

**JugadorPanel JugadorPanel:** Subpanel que muestra la tabla de jugadores.

**JPanel panel:** Panel principal que organiza los subpaneles utilizando un BorderLayout.

#### Constructor

**PanelDeControl(RuletaFrame frame, RuletaModelo modelo):** Constructor que inicializa los subpaneles y el panel principal.

#### Métodos

**createPaneldeControl():** Configura y devuelve el panel principal organizando los subpaneles con un BorderLayout.

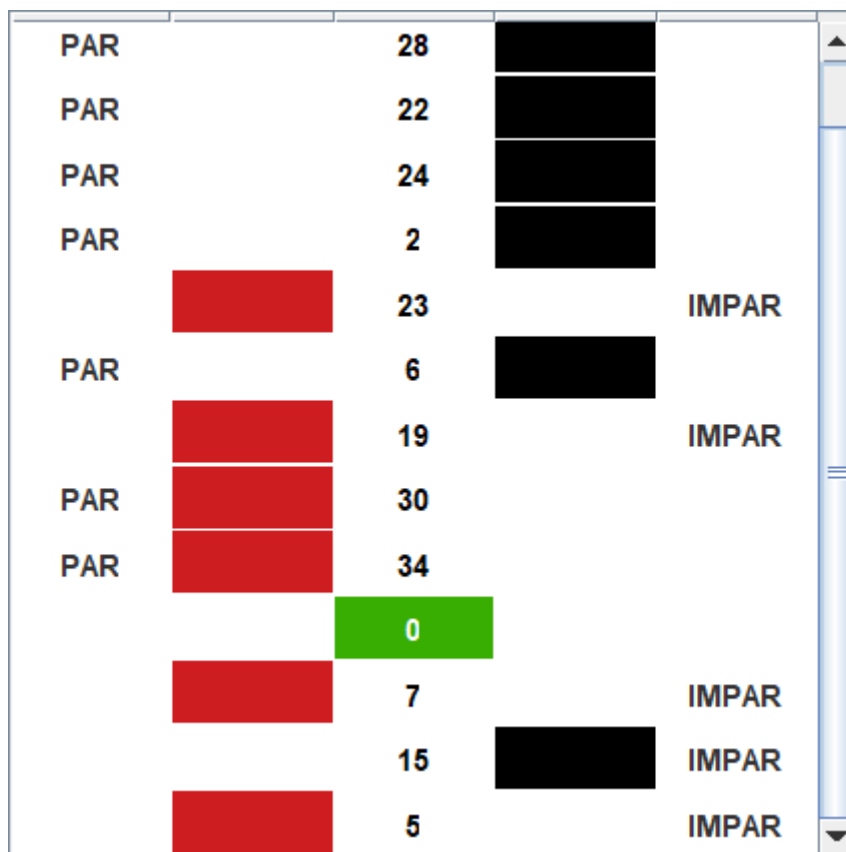
**getPanel():** Devuelve el panel principal para que pueda ser añadido a la interfaz gráfica principal.

**addLlamada(String numeroString, Color colorFondo, boolean isNegro):** Añade una llamada al PanelDeLlamadas.

**ActualizarJugadorPanel():** Actualiza el modelo de la tabla en el JugadorPanel.

**getTablaJugador():** Devuelve la tabla de jugadores del JugadorPanel.

#### -PanelDeLlamadas



PAR	28		
PAR	22		
PAR	24		
PAR	2		
	23		IMPAR
PAR	6		
	19		IMPAR
PAR	30		
PAR	34		
	0		
	7		IMPAR
	15		IMPAR
	5		IMPAR

La clase PanelDeLlamadas en la aplicación de ruleta está diseñada para mostrar las llamadas (números ganadores) en una tabla. Esta tabla incluye

información sobre si el número es par, impar, rojo, negro o verde, y cada celda está coloreada adecuadamente para facilitar la visualización.

### Atributos

**ModeloTablaLlamada:** Un modelo de tabla personalizado para manejar los datos de las llamadas.

**RuletaModelo:** El modelo principal de la ruleta que contiene la lógica del juego.

**JPanel:** El panel que contiene la tabla de llamadas.

**JTable:** La tabla que muestra las llamadas y sus atributos.

### Constructor

**PanelDeLlamadas(RuletaModelo modelo):** Inicializa el modelo de la tabla y crea el panel de llamadas.

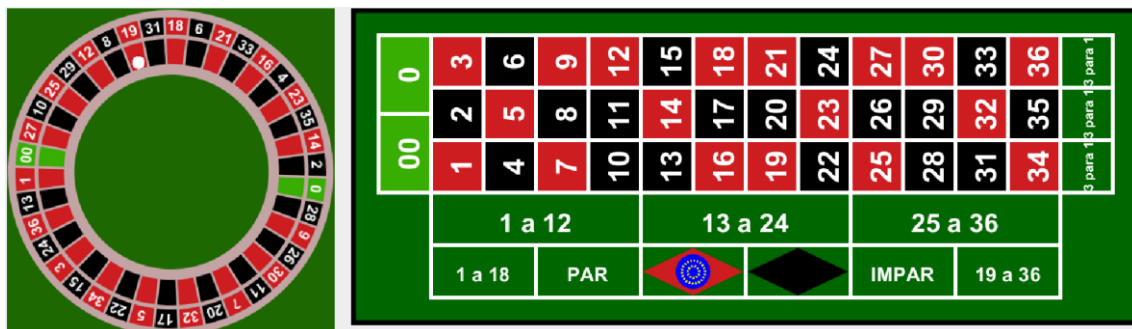
### Métodos

**createPanelDeLlamada():** Crea y configura el panel que muestra las llamadas, incluyendo la tabla y sus renderizadores personalizados.

**addLlamada(String StringNumero, Color colorFondo, boolean isNegro):**  
Agrega una nueva llamada a la tabla y gestiona la eliminación de las filas excedentes para mantener un máximo de 20 filas.

**getPanel():** Devuelve el panel de llamadas para su inclusión en la interfaz de usuario.

## -PanelRuleta



La clase **PanelRuleta** en la aplicación de ruleta está diseñada para manejar la visualización y control de la ruleta, incluyendo tanto la imagen de la ruleta como la tabla de apuestas. Este panel es fundamental para la interacción del usuario con la ruleta y la visualización de los resultados de los giros.

### Atributos

**ImagenTablaRuleta:** Un componente que maneja la imagen de la tabla de ruleta.

**PanelTablaRuleta:** Un panel que contiene la tabla de apuestas.

**RuletaImagen:** Un componente que maneja la imagen de la ruleta.

**RuletaPanel:** Un panel que contiene la lógica para la visualización y control de la ruleta.

**JPanel:** El panel principal que organiza y dispone los componentes anteriores.

#### Constructor

**PanelRuleta(RuletaFrame frame, RuletaModelo modelo):** Inicializa los componentes necesarios y crea el panel principal de control para dar vueltas a la ruleta.

#### Métodos

**createPanelDeControlParaDarVueltas():** Crea y configura el panel principal de control que contiene la imagen de la ruleta y la tabla de apuestas.

**inicializarRuleta():** Inicializa la ruleta en su estado inicial.

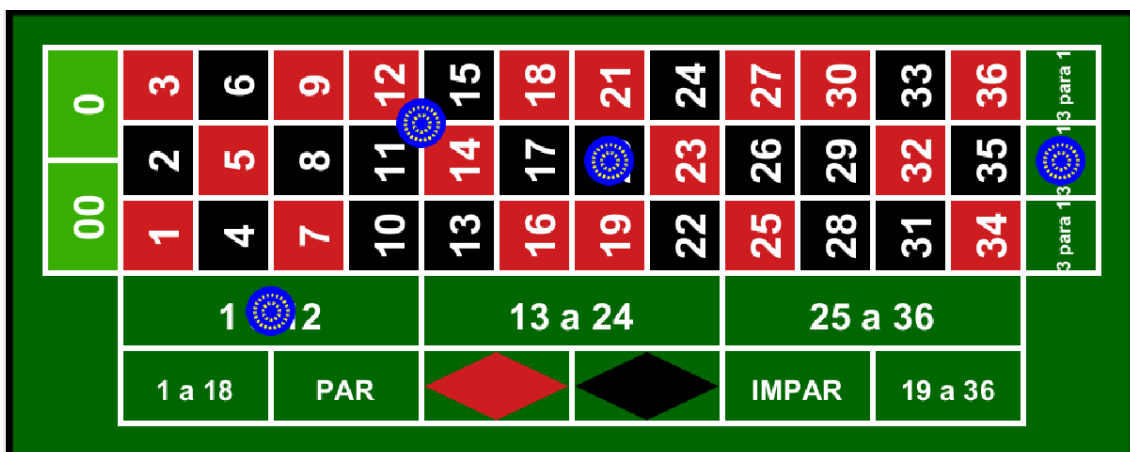
**reposicionarRuleta(SegmentoRuleta SegmentoRuleta, int index):** Reposiciona la ruleta en base al segmento y al índice proporcionado.

**rotarRuleta(double angulo):** Rota la ruleta en base al ángulo proporcionado.

**repintarRuleta():** Redibuja la tabla de apuestas.

**getPanel():** Devuelve el panel principal para su inclusión en la interfaz de usuario.

#### -PanelTablaRuleta



La clase PanelTablaRuleta es una extensión de JPanel y se encarga de dibujar la imagen de la mesa de ruleta y las apuestas realizadas por los jugadores. A continuación, se proporciona una explicación detallada de la implementación y su funcionalidad.

#### Atributos Principales

margen: Define el margen alrededor de la imagen de la mesa de ruleta.

modelo: Referencia al modelo de datos RuletaModelo, que contiene las apuestas y la información de los jugadores.

imagen: Referencia a la imagen de la mesa de ruleta, encapsulada en la clase ImagenTablaRuleta.

### Constructor

El constructor PanelTablaRuleta inicializa los componentes necesarios y configura el tamaño preferido del panel basado en la imagen de la mesa de ruleta. También añade un MouseListener para gestionar las interacciones del usuario.

### Método paintComponent

El método paintComponent sobrescribe el método de JPanel para dibujar la imagen de la mesa de ruleta y las fichas de apuestas de los jugadores.

### Funcionalidad

#### Configuración del Panel:

En el constructor, se configura el tamaño del panel basándose en el tamaño de la imagen de la mesa de ruleta, añadiendo márgenes.

Se añade un MouseListener (FichaListener) para gestionar las interacciones del usuario, como la colocación de fichas.

#### Dibujo de la Mesa y Fichas:

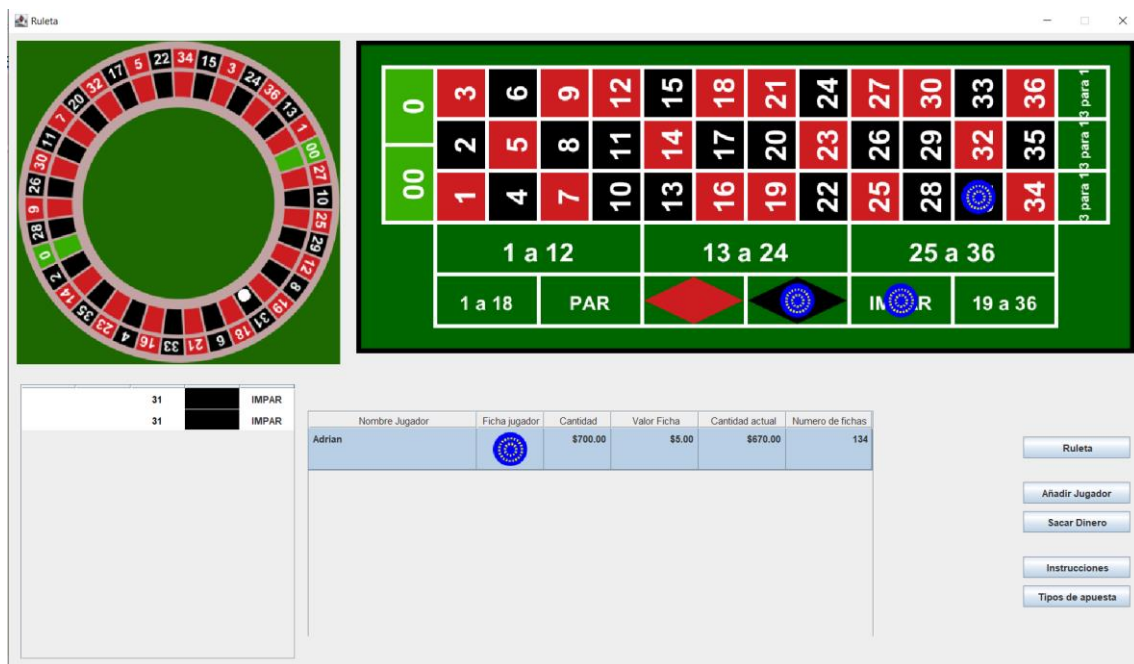
En el método paintComponent, primero se dibuja la imagen de la mesa de ruleta centrada en el panel, tomando en cuenta el margen.

Luego, para cada apuesta en el modelo, se dibuja la ficha correspondiente en la ubicación especificada. La posición de la ficha se ajusta para que esté centrada correctamente en el punto de apuesta.

#### Interacción del Usuario

El FichaListener se utiliza para manejar los clics del usuario en el panel, permitiendo que el usuario coloque apuestas en la mesa.

## -RuletaFrame



La clase RuletaFrame es una clase de la interfaz gráfica de usuario (GUI) que configura y muestra la ventana principal de la aplicación de la ruleta. Esta clase se encarga de inicializar los componentes principales, gestionar la ventana principal (JFrame) y proporcionar métodos para interactuar con estos componentes. A continuación, se desglosan las partes más importantes de esta clase.

### Atributos Principales

**PanelDeControl:** Panel que maneja los controles y la visualización de llamadas de la ruleta.

**PanelRuleta:** Panel que maneja la visualización de la ruleta y las apuestas.

**JFrame:** La ventana principal de la aplicación.

### Constructor

El constructor de RuletaFrame inicializa los paneles y crea y muestra la GUI llamando al método crearMostrarGUI.

### Método crearMostrarGUI

Este método configura y muestra el JFrame con los paneles de control y de ruleta.

### Métodos Públicos



Estos métodos proporcionan formas de interactuar con los paneles y la ruleta desde fuera de la clase RuletaFrame.

addLlamada: Añade una llamada a la tabla de llamadas en PanelDeControl.

updatePanelJugador: Actualiza el panel de jugador en PanelDeControl

getTablaJugador: Devuelve la tabla de jugador de PanelDeControl.

inicializarRuleta: Inicializa la ruleta en PanelRuleta.

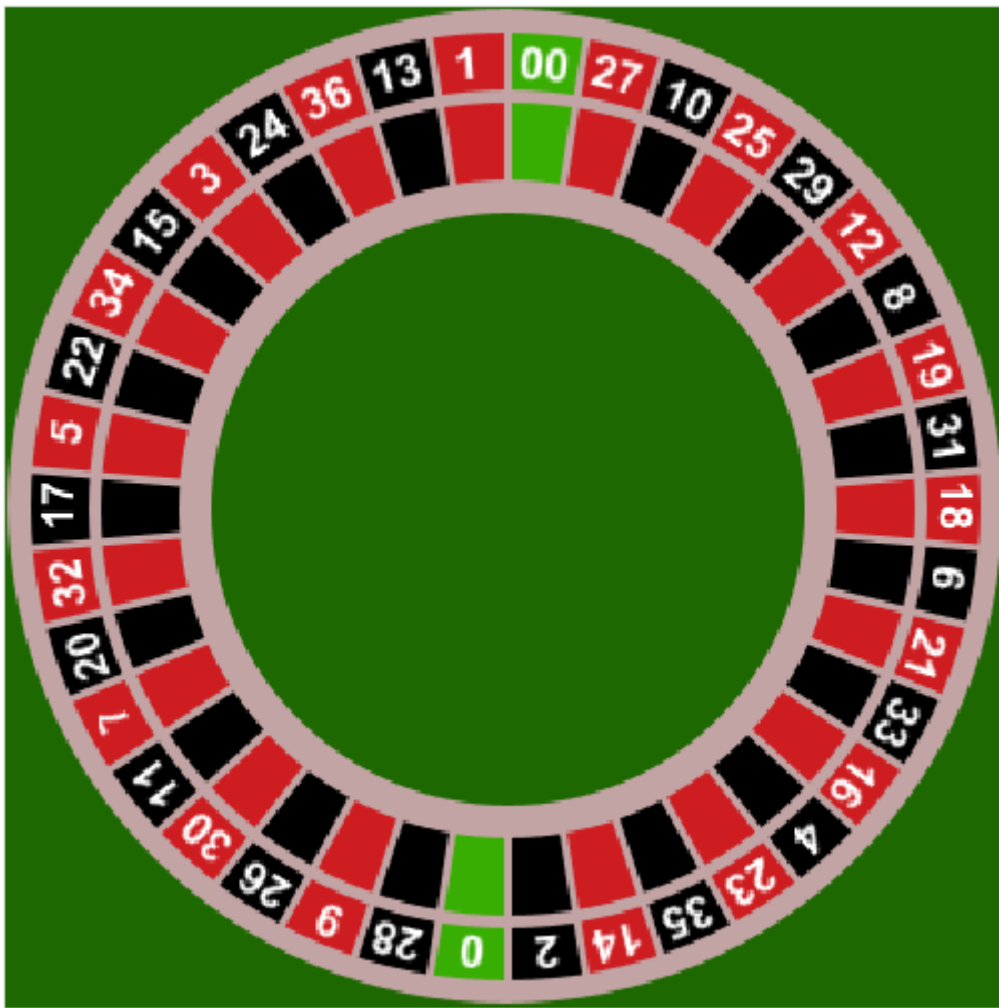
reposicionarRuleta: Reposiciona la ruleta en PanelRuleta.

RotarRuleta: Rota la ruleta en PanelRuleta

rehacerTabladeApuestas: Repinta la tabla de apuestas en PanelRuleta.

getFrame: Devuelve el JFrame principal.

### -RuletaImagen



La clase RuletaImagen es crucial para la generación de la visualización gráfica de la ruleta. Crea una imagen que representa la ruleta, segmenta y colorea

cada parte según el modelo, y posiciona los números en los segmentos correspondientes. Este enfoque modular permite una fácil personalización y mantenimiento del código, asegurando que la imagen de la ruleta se actualice y se pinte correctamente en la interfaz gráfica de la aplicación

### Atributos

BufferedImage imagen: Almacena la imagen de la ruleta.

Color ColorSueloRuleta: Color del fondo de la ruleta.

Color ColorHierroRuleta: Color del borde de la ruleta.

SegmentoRuleta[] SegmentosRuleta: Array de segmentos que componen la ruleta.

### Constructor

El constructor inicializa los colores y los segmentos de la ruleta desde el modelo (RuletaModelo) y luego crea la imagen de la ruleta llamando a createRuleta().

### Método createRuleta

Este método crea la imagen principal de la ruleta, configurando el gráfico y dibujando los segmentos y los bordes.

### Método pintarSegmentos

Este método se encarga de dibujar cada segmento de la ruleta, asignando colores y números a cada uno.

### Método pintarPoligono

Este método dibuja un polígono representando cada segmento de la ruleta.

### Método createNumeroRuleta

Este método crea una imagen con el número y el color de fondo para cada segmento de la ruleta.

### Método getImagen

Este método devuelve la imagen de la ruleta.

## -RuletaPanel



La clase `RuletaPanel` es una subclase de `JPanel` que maneja la visualización y rotación de la ruleta en la aplicación `RuletaApp`. Aquí se describen los componentes y funcionalidades principales de esta clase:

### Atributos

double AcumuladordeAngulos: Acumula los ángulos de rotación aplicados a la ruleta.

int margen: Margen para el panel de la ruleta.

BufferedImage ImagenOriginal: Imagen original de la ruleta.

BufferedImage rotarImagen: Imagen de la ruleta después de aplicar rotaciones.

Color colorFondo: Color de fondo del panel de la ruleta.

### Constructor

El constructor inicializa los atributos y establece el tamaño preferido del panel basado en el tamaño de la imagen de la ruleta más los márgenes.

## Métodos

rotarRuleta(double angulo)

Este método rota la ruleta sumando el ángulo proporcionado al acumulador de ángulos y llamando a rotarRuletaPrivada para aplicar la rotación.

reposicionarRuleta(SegmentoRuleta wheelSegment, int index)

Este método posiciona la ruleta para que un segmento específico esté alineado correctamente. Crea una nueva imagen basada en la imagen original y dibuja una marca blanca en el segmento especificado.

inicializarRuleta()

Este método reinicializa la ruleta aplicando la rotación acumulada a la imagen original.

Cartesiano(double angulo, double radio, Point centerPoint)

Este método convierte un ángulo y un radio en coordenadas cartesianas relativas a un punto central.

rotarRuletaPrivada(BufferedImage bufferedImage, double angulo)

Este método rota la imagen de la ruleta internamente utilizando una transformación afín. Aplica la rotación y actualiza la imagen rotada

Sobrescritura del Método paintComponent

Este método sobrescrito de JPanel se encarga de pintar el componente. Dibuja la imagen rotada de la ruleta en el panel.

## 5. Test

▼ RuletaApp	70% (44/...	58% (166/...	59% (950/1...
▼ controller	83% (5/6)	50% (8/16)	17% (27/152)
AñadirJugadorListener	100% (1/1)	66% (2/3)	21% (7/33)
EliminarJugadorListener	100% (1/1)	100% (2/2)	100% (9/9)
FichaListener	100% (1/1)	50% (2/4)	17% (5/29)
RuletaListener	50% (1/2)	20% (1/5)	4% (3/68)
SeleccionJugadorListener	100% (1/1)	50% (1/2)	23% (3/13)
▼ model	93% (14/...	64% (80/1...	77% (382/4...
ApuesatsClikables	100% (1/1)	93% (14/15)	96% (162/1...
Apuesta	100% (1/1)	66% (8/12)	80% (17/21)
ApuestaClikable	100% (1/1)	16% (1/6)	30% (4/13)
FichaRuleta	100% (1/1)	50% (2/4)	71% (5/7)
FichasRuleta	100% (1/1)	40% (2/5)	80% (12/15)
Jugador	100% (1/1)	78% (11/14)	53% (17/32)
JugadorApuestaError	0% (0/1)	0% (0/3)	0% (0/5)
Ronda	100% (1/1)	83% (5/6)	55% (19/34)
Ruleta	100% (1/1)	100% (13/1...	98% (80/81)
RuletaModelo	100% (1/1)	23% (6/26)	26% (17/65)
SegmentoRuleta	100% (1/1)	100% (4/4)	100% (7/7)
TablaConConstantesDibujab	100% (1/1)	100% (5/5)	100% (9/9)
TipoApuesta	100% (1/1)	83% (5/6)	96% (24/25)
ValorFicha	100% (1/1)	66% (2/3)	75% (3/4)
ValorFichas	100% (1/1)	100% (2/2)	100% (6/6)
▼ view	62% (25/...	54% (78/1...	56% (541/9...
> Texto	0% (0/15)	0% (0/43)	0% (0/314)
ImagenFicha	100% (1/1)	75% (3/4)	96% (27/28)
ImagenTablaRuleta	100% (1/1)	100% (14/1...	100% (160/...
JugadorPanel	100% (6/6)	58% (10/17)	54% (51/93)
MenuPanel	100% (5/5)	63% (7/11)	87% (42/48)
PanelDeControl	100% (1/1)	100% (6/6)	100% (17/17)
PanelDeLlamadas	100% (6/6)	86% (13/15)	82% (80/97)
PanelRuleta	100% (1/1)	85% (6/7)	88% (15/17)
PanelTablaRuleta			

Cobertura del 70%

## 6. Tecnologías utilizadas y referencias

-JAVA

-SWING

-INTELLIJ

-TRELLO

