

	<b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b> <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b> <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
Aprobación: 2022/03/01	Código: GUIA-PRLE-001	Página: 1

## INFORME DE LABORATORIO

### (formato estudiante)

INFORMACIÓN BÁSICA					
ASIGNATURA:	Fundamentos de la programación 2				
TÍTULO DE LA PRÁCTICA:	TRABAJO FINAL DE LABORATORIO DE FUNDAMENTOS DE PROGRAMACIÓN 2				
NÚMERO DE PRÁCTICA:		AÑO LECTIVO:	2024	NRO. SEMESTRE:	2
FECHA DE PRESENTACIÓN	13/12/2024	HORA DE PRESENTACIÓN	16/50/00		
<b>INTEGRANTE (s)</b> Karla Miluska Bedregal Coaguila Usiel Surriel Quispe Puma Jose Manuel Morocco Saico José León Enrique Hatches Curo				<b>NOTA (0-20)</b>	
<b>DOCENTE(s):</b> Ing. Lino Jose Pinto Oppe					

RESULTADOS Y PRUEBAS
<b>CONEXIÓN A GITHUB:</b>  Link al repositorio: <a href="https://github.com/KarlaBedregal/Proyecto-Final---Fundamentos.git">https://github.com/KarlaBedregal/Proyecto-Final---Fundamentos.git</a>
<b>1. RESUMEN</b>  Para este trabajo final, se desarrolló un juego basado en el clásico juego de mesa Ludo, pero implementado con una interfaz gráfica utilizando el lenguaje de programación Java. El objetivo era recrear las funcionalidades originales del juego, como el movimiento de fichas, el lanzamiento del dado y la interacción entre los jugadores.

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 2

En el proceso de desarrollo, se utilizaron diversos conceptos de programación, especialmente en lo que respecta a la creación de interfaces gráficas de usuario (GUI) y la gestión de eventos. Esto permitió ofrecer una experiencia similar al juego tradicional, pero con la comodidad y el dinamismo de una aplicación digital.

Además, se brindó al usuario la posibilidad de personalizar el juego eligiendo la cantidad de jugadores y los nombres, y una vez comenzado, cada jugador podía lanzar el dado, mover sus fichas y ver los resultados en tiempo real. De esta manera, se logra una experiencia interactiva y entretenida para los participantes.

## 2. RESUMEN DEL VIDEOJUEGO

Este videojuego es una versión digital del popular juego de mesa Ludo, desarrollado con una interfaz gráfica utilizando el lenguaje de programación Java. El juego permite a los jugadores competir entre sí, moviendo sus fichas por el tablero según los resultados obtenidos al lanzar el dado.

Los jugadores pueden personalizar la partida al elegir la cantidad de jugadores y asignar nombres a cada uno. Durante el juego, cada jugador tiene un color diferente para sus fichas, y se puede lanzar el dado de manera interactiva, ya sea presionando un botón o usando la tecla "ENTER".

La dinámica del juego sigue las reglas clásicas de Ludo, con la ventaja de que se gestiona todo de manera automática a través de la interfaz gráfica, lo que facilita la experiencia y la hace más atractiva. El objetivo es mover todas las fichas hasta la meta antes que los demás jugadores.

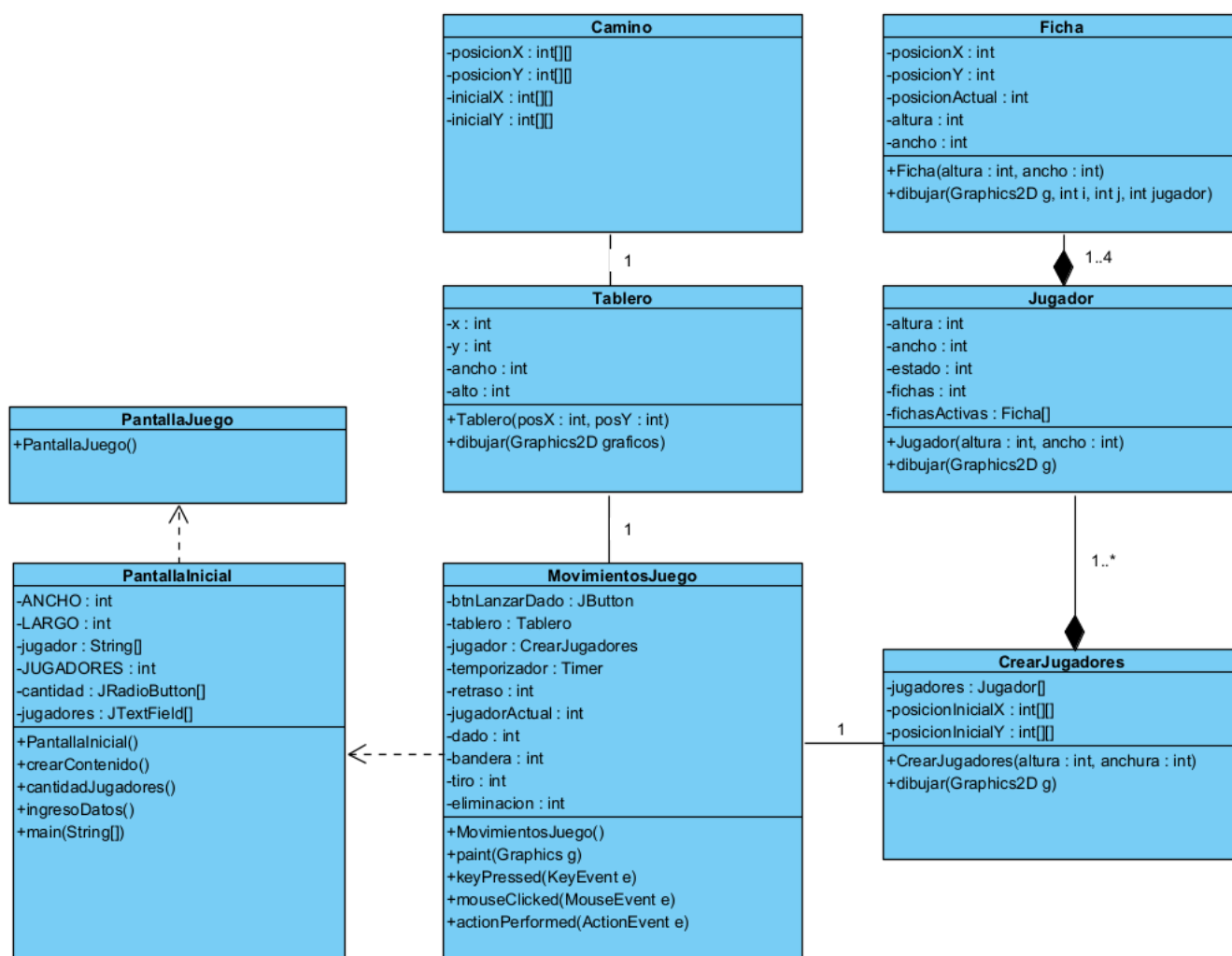
**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 3

### 3. DIAGRAMA DE CLASE UML



	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 4</p>

## 4. CÓDIGO COMPLETO

### - CLASE Camino:

```
public class Camino {
    // MATRIZ DE POSICIONES EN EL EJE X PARA CADA CASILLA DEL TABLERO
    static int[][] posicionX = {
        {1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 14, 13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 5, 6},
        {8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 14, 13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7},
        {13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 13, 12, 11, 10, 9, 8},
        {6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 14, 13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7}
    };

    // MATRIZ DE POSICIONES EN EL EJE Y PARA CADA CASILLA DEL TABLERO
    static int[][] posicionY = {
        {6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 14, 13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 7, 7, 7, 7, 7},
        {1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 1, 2, 3, 4, 5, 6},
        {8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 14, 13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 7, 7, 7, 7, 7},
        {13, 12, 11, 10, 9, 8, 8, 8, 8, 8, 7, 6, 6, 6, 6, 6, 5, 4, 3, 2, 1, 0, 0, 0, 1, 2, 3, 4, 5, 6, 6, 6, 6, 6, 6, 7, 8, 8, 8, 8, 8, 9, 10, 11, 12, 13, 14, 14, 13, 12, 11, 10, 9, 8}
    };

    // POSICIONES INICIALES EN EL EJE X PARA CADA JUGADOR
    static int[][] inicialX = {
        {1, 1, 3, 3},
        {10, 10, 12, 12},
        {10, 10, 12, 12},
        {1, 1, 3, 3}
    };

    // POSICIONES INICIALES EN EL EJE Y PARA CADA JUGADOR
    static int[][] inicialY = {
        {1, 3, 1, 3},
        {1, 3, 1, 3},
        {10, 12, 10, 12},
        {10, 12, 10, 12}
    };
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 5

**- CLASE CrearJugadores:**

```
import java.awt.Graphics2D;

public class CrearJugadores {

    // ARREGLO DE OBJETOS JUGADOR
    Jugador[] jugadores = new Jugador[PantallaInicial.JUGADORES];
    // MATRIZ DE POSICIONES INICIALES EN EJE X PARA CADA JUGADOR
    int[][] posicionInicialX = {
        {1, 1, 3, 3},
        {10, 10, 12, 12},
        {10, 10, 12, 12},
        {1, 1, 3, 3}
    };

    // MATRIZ DE POSICIONES INICIALES EN EJE Y PARA CADA JUGADOR
    int[][] posicionInicialY = {
        {1, 3, 1, 3},
        {1, 3, 1, 3},
        {10, 12, 10, 12},
        {10, 12, 10, 12}
    };

    // CONSTRUCTOR DE LA CLASE CREARJUGADORES
    public CrearJugadores(int altura, int anchura) {
        for (int i = 0; i < PantallaInicial.JUGADORES; i++) {
            // INICIALIZA CADA OBJETO JUGADOR CON ALTURA Y ANCHURA DADAS
            jugadores[i] = new Jugador(altura, anchura);
        }
    }

    // METODO PARA DIBUJAR LOS JUGADORES EN EL TABLERO
    public void dibujar(Graphics2D g) {
        for (int i = 0; i < PantallaInicial.JUGADORES; i++) {
            for (int j = 0; j < 4; j++) {
                // DIBUJA CADA FICHA DEL JUGADOR EN SU POSICION INICIAL
                jugadores[i].fichasActivas[j].dibujar(g, posicionInicialX[i][j], posicionInicialY[i][j], i);
            }
        }
    }
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 6

### - Clase Ficha:

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Graphics2D;

public class Ficha {
    int posicionX, posicionY; // COORDENADAS DE LA FICHA
    int posicionActual; // POSICION ACTUAL EN EL TABLERO
    int altura, ancho; // DIMENSIONES DE LA FICHA

    // CONSTRUCTOR DE LA CLASE FICHA
    public Ficha(int altura, int ancho) {
        posicionActual = -1; // INICIALIZA POSICION ACTUAL A -1 (FUERA DEL TABLERO)
        posicionX = -1; // INICIALIZA POSICION X A -1
        posicionY = -1; // INICIALIZA POSICION Y A -1
        this.altura = altura; // ASIGNA ALTURA
        this.ancho = ancho; // ASIGNA ANCHO
    }

    // METODO PARA DIBUJAR LA FICHA
    public void dibujar(Graphics2D graficos, int i, int j, int jugador) {
        if (posicionActual == -1) { // SI LA FICHA ESTE FUERA DEL TABLERO
            int temp1 = 80 + (altura / 2), temp2 = 50 + (ancho / 2); // CALCULA LAS COORDENADAS TEMPORALES
            posicionX = i; // ASIGNA LA POSICION X
            posicionY = j; // ASIGNA LA POSICION Y
            if (jugador == 0) {
                graficos.setColor(Color.RED); // ASIGNA COLOR ROJO SI ES EL JUGADOR 0
            } else if (jugador == 1) {
                graficos.setColor(Color.GREEN); // ASIGNA COLOR VERDE SI ES EL JUGADOR 1
            } else if (jugador == 2) {
                graficos.setColor(Color.YELLOW); // ASIGNA COLOR AMARILLO SI ES EL JUGADOR 2
            } else if (jugador == 3) {
                graficos.setColor(Color.BLUE); // ASIGNA COLOR AZUL SI ES EL JUGADOR 3
            }
            graficos.fillOval(temp1 + 5 + (i * ancho), temp2 + 5 + (j * altura), ancho - 10, altura - 10); // DIBUJA LA FICHA
            graficos.setStroke(new BasicStroke(2)); // ESTABLECE EL GROSOR DEL BORDE
            graficos.setColor(Color.BLACK); // ASIGNA EL COLOR NEGRO PARA EL BORDE
            graficos.drawOval(temp1 + 5 + (i * ancho), temp2 + 5 + (j * altura), ancho - 10, altura - 10); // DIBUJA EL BORDE DE LA FICHA
        } else { // SI LA FICHA ESTA EN EL TABLERO
            int temp1 = 80, temp2 = 50; // COORDENADAS TEMPORALES
            posicionX = Camino.posicionX[jugador][posicionActual]; // ACTUALIZA LA POSICION X BASADO EN EL CAMINO DEL JUGADOR
            posicionY = Camino.posicionY[jugador][posicionActual]; // ACTUALIZA LA POSICION Y BASADO EN EL CAMINO DEL JUGADOR
            if (jugador == 0) {
                graficos.setColor(Color.RED); // ASIGNA COLOR ROJO SI ES EL JUGADOR 0
            } else if (jugador == 1) {
                graficos.setColor(Color.GREEN); // ASIGNA COLOR VERDE SI ES EL JUGADOR 1
            } else if (jugador == 2) {
                graficos.setColor(Color.YELLOW); // ASIGNA COLOR AMARILLO SI ES EL JUGADOR 2
            } else if (jugador == 3) {
                graficos.setColor(Color.BLUE); // ASIGNA COLOR AZUL SI ES EL JUGADOR 3
            }
            graficos.fillOval(temp1 + 5 + (posicionX * ancho), temp2 + 5 + (posicionY * altura), ancho - 10, altura - 10); // DIBUJA LA FICHA
            graficos.setStroke(new BasicStroke(2)); // ESTABLECE EL GROSOR DEL BORDE
            graficos.setColor(Color.BLACK); // ASIGNA EL COLOR NEGRO PARA EL BORDE
            graficos.drawOval(temp1 + 5 + (posicionX * ancho), temp2 + 5 + (posicionY * altura), ancho - 10, altura - 10); // DIBUJA EL BORDE DE LA FICHA
        }
    }
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 7

### - Clase Jugador:

```
import java.awt.Graphics2D;

public class Jugador {

    int altura, ancho, estado, fichas; // VARIABLES QUE DEFINEN EL JUGADOR
    Ficha[] fichasActivas = new Ficha[4]; // ARREGLO DE FICHAS DEL JUGADOR

    // CONSTRUCTOR DE LA CLASE JUGADOR
    public Jugador(int altura, int ancho) {
        estado = -1; // INICIALIZA EL ESTADO DEL JUGADOR A -1 (FUERA DEL JUEGO)
        fichas = 0; // INICIALIZA EL CONTEO DE FICHAS EN 0
        for (int i = 0; i < 4; i++) {
            fichasActivas[i] = new Ficha(altura, ancho); // INICIALIZA CADA FICHA CON LAS DIMENSIONES DADAS
        }
    }

    // MÉTODO PARA DIBUJAR EL JUGADOR
    public void dibujar(Graphics2D graficos) {
        // Código auxiliar de método generado automáticamente
    }
}
```

### - Clase MovimientosJuego:

```
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.awt.event.KeyEvent;
import java.awt.event.KeyListener;
import java.awt.event.MouseEvent;
import java.awt.event.MouseListener;

import javax.swing.Timer;
import javax.swing.JPanel;
import javax.swing.JButton;
import javax.swing.ImageIcon;

public class MovimientosJuego extends JPanel implements KeyListener, ActionListener, MouseListener {

    private JButton btnLanzarDado;

    private static final long serialVersionUID = 1L; // IDENTIFICADOR DE VERSIÓN PARA LA SERIALIZACIÓN
    Tablero tablero; // OBJETO QUE REPRESENTA EL TABLERO DEL JUEGO
    CrearJugadores jugador; // OBJETO QUE REPRESENTA LOS JUGADORES DEL JUEGO
    Timer temporizador; // OBJETO TIMER PARA EL CONTROL DEL TIEMPO
    int retraso = 10; // RETRASO DEL TEMPORIZADOR EN MILLISEGUNDOS
    int jugadorActual, dado; // VARIABLES PARA CONTROLAR EL JUGADOR ACTUAL Y EL VALOR DEL DADO
    int bandera = 0, tiro, eliminacion = 0; // VARIABLES DE ESTADO

    // CONSTRUCTOR DE LA CLASE MOVIMIENTOSJUEGO
    public MovimientosJuego() {
        setFocusable(true); // Asegura que el componente pueda recibir el foco
        setFocusTraversalKeysEnabled(false); // DESHABILITA LAS TECLAS DE TRAVESÍA DE FOCUS
        requestFocus(); // SOLICITA FOCUS PARA EL PANEL
        jugadorActual = 0; // INICIALIZA EL JUGADOR ACTUAL A 0
        tablero = new Tablero(80, 50); // CREA UN NUEVO TABLERO CON LAS DIMENSIONES DADAS
        jugador = new CrearJugadores(tablero.alto, tablero.ancho); // CREA LOS JUGADORES CON LAS DIMENSIONES DEL TABLERO
        dado = 0; // INICIALIZA EL VALOR DEL DADO A 0
        bandera = 0; // INICIALIZA LA BANDERA A 0
        tiro = 0; // INICIALIZA EL TIRO A 0
        eliminacion = 0; // INICIALIZA LA ELIMINACIÓN A 0
    }
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 8

```

        this.setBackground(new Color(255, 223, 110));

        btnLanzarDado = new JButton("Lanzar Dado");
        btnLanzarDado.setBounds(700, 200, 200, 50); // Ubicación y tamaño del botón
        btnLanzarDado.setBackground(Color.BLACK); // Color de fondo negro
        btnLanzarDado.setForeground(Color.WHITE); // Color del texto blanco
        btnLanzarDado.setFont(new Font("Arial", Font.BOLD, 18)); // Estilo del texto
        btnLanzarDado.addActionListener(this); // Agregar el listener de acción

        this.setLayout(null);
        this.add(btnLanzarDado); // AÑADE EL BOTON AL PANEL
    }

    // METODO PARA DIBUJAR LOS COMPONENTES DEL JUEGO
    @Override
    public void paint(Graphics g) {
        super.paint(g);
        tablero.dibujar((Graphics2D) g); // DIBUJA EL TABLERO
        jugador.dibujar((Graphics2D) g); // DIBUJA LOS JUGADORES

        if (jugador.jugadores[jugadorActual].fichas == 4) { // VERIFICA SI EL JUGADOR ACTUAL HA GANADO
            g.setColor(Color.WHITE);
            g.fillRect(590, 100, 380, 130); // DIBUJA UN RECTANGULO BLANCO PARA EL MENSAJE DE GANADOR
            if (jugadorActual == 0) {
                g.setColor(Color.RED);
            } else if (jugadorActual == 1) {
                g.setColor(Color.GREEN);
            } else if (jugadorActual == 2) {
                g.setColor(Color.YELLOW);
            } else if (jugadorActual == 3) {
                g.setColor(Color.BLUE);
            }
            g.setFont(new Font("serif", Font.BOLD, 40));
            g.drawString("Jugador " + (PantallaInicial.jugador[jugadorActual]) + " gana.", 600, 150); // MENSAJE DE GANADOR
            g.drawString("Felicidades.", 600, 200); // MENSAJE DE FELICITACIONES
            jugadorActual = 0; // REINICIA EL JUGADOR ACTUAL A 0
            tablero = new Tablero(80, 50); // CREA UN NUEVO TABLERO
            jugador = new CrearJugadores(tablero.alto, tablero.ancha); // CREA NUEVOS JUGADORES
            dado = 0; // REINICIA EL VALOR DEL DADO
            bandera = 0; // REINICIA LA BANDERA
            tiro = 0; // REINICIA EL TIRO
            eliminacion = 0; // REINICIA LA ELIMINACION
        } else if (dado != 0) { // SI SE HA LANZADO EL DADO
            if (jugadorActual == 0) {
                g.setColor(Color.RED); // Color para el jugador 1
            } else if (jugadorActual == 1) {
                g.setColor(Color.GREEN); // Color para el jugador 2
            } else if (jugadorActual == 2) {
                g.setColor(Color.YELLOW); // Color para el jugador 3
            } else if (jugadorActual == 3) {
                g.setColor(Color.BLUE); // Color para el jugador 4
            }
            g.fillRect(590, 100, 420, 70); // DIBUJA UN RECTANGULO PARA MOSTRAR EL VALOR DEL DADO
            if (jugadorActual == 0) {
                g.setColor(Color.RED);
            } else if (jugadorActual == 1) {
                g.setColor(Color.GREEN);
            } else if (jugadorActual == 2) {
                g.setColor(Color.YELLOW);
            } else if (jugadorActual == 3) {
                g.setColor(Color.BLUE);
            }
            g.setColor(Color.BLACK);
            g.setFont(new Font("serif", Font.BOLD, 40));
            g.drawString("Jugador " + (PantallaInicial.jugador[jugadorActual]), 600, 150); // MUESTRA EL JUGADOR ACTUAL

            String dadoPath = "images/dado" + dado + ".png";
            ImageIcon dadoIcon = new ImageIcon(dadoPath);
            g.drawImage(dadoIcon.getImage(), 900, 100, this);
        }
        if (bandera == 0 && dado != 0 && dado != 6 && eliminacion == 0) {
            jugadorActual = (jugadorActual + 1) % PantallaInicial.JUGADORES; // CAMBIA AL SIGUIENTE JUGADOR
        }
        eliminacion = 0; // REINICIA LA ELIMINACION
    }

    // METODO PARA MANEJAR EVENTOS DE TECLADO
    @Override
    public void keyPressed(KeyEvent e) {
        if (e.getKeyCode() == KeyEvent.VK_ENTER && bandera == 0) { // SI SE PRESIONA ENTER Y LA BANDERA ESTE EN 0
            tiro = 0; // REINICIA EL TIRO
        }
    }

```



Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 9

```
        dado = 1 + (int) (Math.random() * 6); // GENERA UN NÚMERO ALEATORIO ENTRE 1 Y 6
        repaint(); // REPINTA EL COMPONENTE
        for (int i = 0; i < 4; i++) {
            if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual != -1
                && jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual != 56
                && (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual + dado) <= 56) {
                bandera = 1; // ACTUALIZA LA BANDERA SI HAY UN MOVIMIENTO VÁLIDO
                break;
            }
        }
        if (bandera == 0 && dado == 6) { // SI NO HAY MOVIMIENTO VÁLIDO PERO EL DADO ES 6
            for (int i = 0; i < 4; i++) {
                if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual == -1) {
                    bandera = 1; // ACTUALIZA LA BANDERA SI HAY UNA FICHA FUERA DEL TABLERO
                    break;
                }
            }
        }
    }
}

// MÉTODO QUE MANEJA LOS EVENTOS DE CLIC DEL RATÓN
public void mouseClicked(MouseEvent e) {
    if (bandera == 1) { // VERIFICA SI SE PUEDE REALIZAR UNA ACCIÓN
        int x = e.getX(); // OBTIENE LA COORDENADA X DEL CLIC
        int y = e.getY(); // OBTIENE LA COORDENADA Y DEL CLIC
        x = x - 80; // AJUSTA LA COORDENADA X
        y = y - 50; // AJUSTA LA COORDENADA Y
        x = x / 30; // ESCALA LA COORDENADA X A LA DIMENSIÓN DE LAS CASILLAS
        y = y / 30; // ESCALA LA COORDENADA Y A LA DIMENSIÓN DE LAS CASILLAS
        int valor = -1; // VARIABLE PARA GUARDAR LA FICHA SELECCIONADA

        if (dado == 6) { // SI EL DADO ES 6
            for (int i = 0; i < 4; i++) {
                if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionX == x
                    && jugador.jugadores[jugadorActual].fichasActivas[i].posicionY == y
                    && (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual + dado) <= 56) {
                    valor = i; // ASIGNA LA FICHA SELECCIONADA
                    bandera = 0; // REINICIA LA BANDERA
                    break;
                }
            }
            if (valor != -1) { // SI SE SELECCIONÓ UNA FICHA
                jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual += dado; // MUEVE LA FICHA
                if (jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual == 56) {
                    jugador.jugadores[jugadorActual].fichas++; // INCREMENTA LAS FICHAS DEL JUGADOR
                }
                int k = 0;
                int pos = jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual;
                if ((pos % 13) != 0 && (pos % 13) != 8 && pos < 51) { // VERIFICA SI LA FICHA CAE EN UNA POSICIÓN ESPECIAL
                    for (int i = 0; i < PantallaInicial.JUGADORES; i++) {
                        if (i != jugadorActual) {
                            for (int j = 0; j < 4; j++) {
                                int temp1 = Camino.posicionX[jugadorActual][jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual];
                                int temp2 = Camino.posicionY[jugadorActual][jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual];
                                if (jugador.jugadores[i].fichasActivas[j].posicionX == temp1
                                    && jugador.jugadores[i].fichasActivas[j].posicionY == temp2) {
                                    jugador.jugadores[i].fichasActivas[j].posicionActual = -1; // ELIMINA LA FICHA DEL CONTRINCANTE
                                    eliminacion = 1; // MARCA QUE SE REALIZÓ UNA ELIMINACIÓN
                                    k = 1;
                                    break;
                                }
                            }
                        }
                    }
                    if (k == 1) {
                        break;
                    }
                }
            }
        } else { // SI NO SE SELECCIONÓ UNA FICHA
            for (int i = 0; i < 4; i++) {
                if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual == -1) {
                    jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual = 0; // MUEVE LA FICHA DESDE LA BASE
                    bandera = 0; // REINICIA LA BANDERA
                    break;
                }
            }
        }
    } else { // SI EL DADO NO ES 6
        for (int i = 0; i < 4; i++) {
            if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionX == x
                && jugador.jugadores[jugadorActual].fichasActivas[i].posicionY == y
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 10

```
        && (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual + dado) <= 56) {
            valor = i; // ASIGNA LA FICHA SELECCIONADA
            bandera = 0; // REINICIA LA BANDERA
            break;
        }
    }
    if (valor != -1) { // SI SE SELECCIONA UNA FICHA
        jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual += dado; // MUEVE LA FICHA
        if (jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual == 56) {
            jugador.jugadores[jugadorActual].fichas++; // INCREMENTA LAS FICHAS DEL JUGADOR
        }
        int k = 0;
        int pos = jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual;
        if ((pos % 13) != 0 && (pos % 13) != 8 && pos < 51) { // VERIFICA SI LA FICHA CAE EN UNA POSICION ESPECIAL
            for (int i = 0; i < PantallaInicial.JUGADORES; i++) {
                if (i != jugadorActual) {
                    for (int j = 0; j < 4; j++) {
                        int temp1 = Camino.posicionX[jugadorActual][jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual];
                        int temp2 = Camino.posicionY[jugadorActual][jugador.jugadores[jugadorActual].fichasActivas[valor].posicionActual];
                        if (jugador.jugadores[i].fichasActivas[j].posicionX == temp1
                            && jugador.jugadores[i].fichasActivas[j].posicionY == temp2) {
                            jugador.jugadores[i].fichasActivas[j].posicionActual = -1; // ELIMINA LA FICHA DEL CONTRINCANTE
                            eliminacion = 1; // MARCA QUE SE REALIZA UNA ELIMINACION
                            k = 1;
                            break;
                        }
                    }
                }
            }
            if (k == 1) {
                break;
            }
        }
    }
    repaint(); // REPINTA EL COMPONENTE
}

// METODO QUE SE EJECUTA CON EL BOTON LANZAR DADO
@Override
public void actionPerformed(ActionEvent e) {
    if (bandera == 0) { // SI LA BANDERA ESTÁ EN 0
        tiro = 0; // REINICIA EL TIRO
        dado = 1 + (int) (Math.random() * 6); // GENERA UN DADO (NÚMERO ALEATORIO ENTRE 1 Y 6)
        repaint(); // REPINTA LA PANTALLA

        // VERIFICA SI ALGUNA FICHA ACTIVA DEL JUGADOR ACTUAL PUEDE MOVERSE
        for (int i = 0; i < 4; i++) {
            if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual != -1 // LA FICHA ESTÁ EN EL TABLERO
                && jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual != 56 // LA FICHA NO ESTÁ EN META
                && (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual + dado) <= 56) { // EL MOVIMIENTO ES VÁLIDO
                    bandera = 1; // ESTABLECE LA BANDERA SI HAY UN MOVIMIENTO VÁLIDO
                    break;
            }
        }

        // SI NO HAY MOVIMIENTO VÁLIDO PERO EL DADO ES 6, VERIFICA FICHAS FUERA DEL TABLERO
        if (bandera == 0 && dado == 6) {
            for (int i = 0; i < 4; i++) {
                if (jugador.jugadores[jugadorActual].fichasActivas[i].posicionActual == -1) { // HAY UNA FICHA FUERA DEL TABLERO
                    bandera = 1; // ESTABLECE LA BANDERA SI HAY UNA FICHA FUERA
                    break;
                }
            }
        }
    }
}

// METODO QUE SE EJECUTA CUANDO SE SUELTA UNA TECLA (NO IMPLEMENTADO)
@Override
public void keyReleased(KeyEvent arg0) {
    // TODO Auto-generated method stub
}

// METODO QUE SE EJECUTA CUANDO SE PRESIONA UNA TECLA (NO IMPLEMENTADO)
@Override
public void keyTyped(KeyEvent arg0) {
    // TODO Auto-generated method stub
}
```

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

**Aprobación:** 2022/03/01

**Código:** GUIA-PRLE-001

**Página:** 11

```
// METODO QUE SE EJECUTA CUANDO EL RATON ENTRA EN EL COMPONENTE (NO IMPLEMENTADO)
@Override
public void mouseEntered(MouseEvent arg0) {
    // TODO Auto-generated method stub
}

// METODO QUE SE EJECUTA CUANDO EL RATON SALE DEL COMPONENTE (NO IMPLEMENTADO)
@Override
public void mouseExited(MouseEvent arg0) {
    // TODO Auto-generated method stub
}

// METODO QUE SE EJECUTA CUANDO SE PRESIONA UN BOTON DEL RATON (NO IMPLEMENTADO)
@Override
public void mousePressed(MouseEvent e) {
    // TODO Auto-generated method stub
}

// METODO QUE SE EJECUTA CUANDO SE SUELTA UN BOTON DEL RATON (NO IMPLEMENTADO)
@Override
public void mouseReleased(MouseEvent arg0) {
    // TODO Auto-generated method stub
}
}
```

### - Clase PantallaInicial:

```
import javax.swing.*;
import java.awt.*;
import java.awt.event.*;

public class PantallaInicial extends JFrame {
    private static final int ANCHO = 300;
    private static final int LARGO = 350;
    public static String[] jugador = new String[4];
    public static int JUGADORES;
    private JRadioButton[] cantidad;
    private JTextField[] jugadores;

    public PantallaInicial() {
        setTitle("Configuracion"); // TITULO DE LA VENTANA
        setSize(ANCHO, LARGO); // TAMAÑO DE LA VENTANA
        setResizable(false); // DESHABILITA CAMBIO DE TAMAÑO
        setLocationRelativeTo(null); // CENTRA LA VENTANA
        setAlwaysOnTop(true); // PONEMS ENCIMA DE OTRAS VENTANAS
        setLayout(new GridLayout(2, 0, 0, 15));
        getContentPane().setBackground(new Color(173, 216, 230)); // COLOR CELESTE DE LA VENTANA
        setDefaultCloseOperation(EXIT_ON_CLOSE); // TERMINA EL PROGRAMA AL CERRAR
        crearContenido(); // LLAMA A CREAR LOS ELEMENTOS DE LA VENTANA
        setVisible(true); // HACE LA VENTANA VISIBLE
        setEnabled(true); // HABILITA LA INTERACCION
    }

    public void crearContenido() {
        cantidadJugadores(); // CREA LAS OPCIONES PARA SELECCIONAR CANTIDAD DE JUGADORES
        ingresoDatos(); // CREA EL FORMULARIO PARA INGRESAR LOS NOMBRES DE LOS JUGADORES
    }

    public void cantidadJugadores() {
        // CANTIDAD DE JUGADORES
        ButtonGroup botones = new ButtonGroup();
        cantidad = new JRadioButton[3];

        cantidad[0] = new JRadioButton("2 Jugadores");
        cantidad[1] = new JRadioButton("3 Jugadores");
    }
}
```

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 12

```
cantidad[2] = new JRadioButton("4 Jugadores");

JPanel cantidadPanelTitulo = new JPanel(new FlowLayout(FlowLayout.CENTER, 50, 10));
cantidadPanelTitulo.add(new JLabel("OPCIONES DE PARTIDA"));

JPanel cantidadPanel = new JPanel(new GridLayout(4, 1));
cantidadPanel.add(new JLabel("Numero de Jugadores:")); // TITULO PARA SELECCION DE JUGADORES

for (JRadioButton jugadores : cantidad) {
    botones.add(jugadores); // AGREGA LOS BOTONES AL GRUPO
    cantidadPanel.add(jugadores);
}

agregarListener(); // AGREGA LOS LISTENERS PARA LOS BOTONES DE OPCION

JPanel cantidadPanel2 = new JPanel(new FlowLayout(FlowLayout.LEFT, 30, 10));
cantidadPanel2.add(cantidadPanelTitulo);
cantidadPanel2.add(cantidadPanel);
add(cantidadPanel2); // AGREGA EL PANEL A LA VENTANA
}

public void agregarListener() {
    cantidad[0].addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (jugadores != null) {
                jugadores[1].setEnabled(true);
                jugadores[2].setEnabled(false);
                jugadores[3].setEnabled(false);
            }
            JUGADORES = 2; // ESTABLECE 2 JUGADORES
        }
    });

    cantidad[1].addActionListener(new ActionListener() {
        public void actionPerformed(ActionEvent e) {
            if (jugadores != null) {
                jugadores[1].setEnabled(true);
                jugadores[2].setEnabled(true);
            }
        }
    });
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 13

```
jugadores[3].setEnabled(false);
}
JUGADORES = 3; // ESTABLECE 3 JUGADORES
});

cantidad[2].addActionListener(new ActionListener() {
    public void actionPerformed(ActionEvent e) {
        if (jugadores != null) {
            jugadores[1].setEnabled(true);
            jugadores[2].setEnabled(true);
            jugadores[3].setEnabled(true);
        }
        JUGADORES = 4; // ESTABLECE 4 JUGADORES
    }
});
}

public void ingresoDatos() {
    // INGRESO DE NOMBRES DE LOS JUGADORES
    JPanel panelIngresar = new JPanel(new FlowLayout());
    jugadores = new JTextField[4];

    for (int i = 0; i < jugadores.length; i++) {
        jugadores[i] = new JTextField(15); // CAJAS DE TEXTO PARA LOS NOMBRES
    }

    for (int i = 0; i < jugadores.length; i++) {
        panelIngresar.add(new JLabel("Jugador " + (i + 1) + " :")); // ETIQUETA PARA CADA JUGADOR
        panelIngresar.add(jugadores[i]); // AGREGA LA CAJA DE TEXTO
    }

    // BOTON DE ENVIAR CONFIGURACIÓN
    JButton botonEnviar = new JButton("Configurar");
    botonEnviar.setBackground(Color.GREEN); // COLOR VERDE PARA EL BOTON
    botonEnviar.addActionListener(new Listener()); // AGREGA EL LISTENER AL BOTON
    panelIngresar.add(botonEnviar);
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 14

```
    add(panelIngresar); // AGREGA EL PANEL A LA VENTANA
}

private class Listener implements ActionListener {
    public void actionPerformed(ActionEvent e) {
        // METODO PARA ESTABLECER LOS NOMBRES DE LOS JUGADORES
        for (int i = 0; i < jugadores.length; i++) {
            // SOLO ASIGNA NOMBRES SI EL JUGADOR ESTÁ HABILITADO
            if (i < JUGADORES) {
                jugador[i] = jugadores[i].getText(); // ASIGNA EL NOMBRE
            } else {
                jugador[i] = ""; // SI NO ESTÁ HABILITADO, ASIGNA VALOR VACÍO
            }
        }

        dispose(); // CIERRA LA VENTANA ACTUAL
        new PantallaJuego(); // INICIA LA PANTALLA DEL JUEGO
    }
}

public static void main(String[] args) {
    new PantallaInicial(); // INICIA LA PANTALLA INICIAL
}
```

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 15

### - Clase PantallaJuego:

```
import javax.swing.JFrame;

public class PantallaJuego {

    public PantallaJuego() {

        // CREA EL OBJETO JFRAME QUE REPRESENTA LA VENTANA DEL JUEGO
        JFrame ventanaJuego = new JFrame();

        // ESTABLECE LAS DIMENSIONES Y LA UBICACIÓN DE LA VENTANA DEL JUEGO
        ventanaJuego.setBounds(10, 10, 1220, 600);

        // ESTABLECE EL TÍTULO DE LA VENTANA DEL JUEGO
        ventanaJuego.setTitle("LUDO");

        // IMPIDE QUE LA VENTANA DEL JUEGO SE PUEDA REDIMENSIONAR
        ventanaJuego.setResizable(false);

        // ESTABLECE LA OPERACIÓN POR DEFECTO AL CERRAR LA VENTANA
        ventanaJuego.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // CREA UN OBJETO DE LA CLASE MOVIMIENTOSJUEGO QUE CONTIENE LA LÓGICA DEL JUEGO
        MovimientosJuego movimientosJuego = new MovimientosJuego();

        // ESTABLECE EL OBJETO MOVIMIENTOSJUEGO COMO FOCUSABLE
        movimientosJuego.setFocusable(true);

        // AÑADE EL OBJETO MOVIMIENTOSJUEGO COMO ESCUCHADOR DE EVENTOS DEL TECLADO Y RATÓN
        movimientosJuego.addKeyListener(movimientosJuego);
        movimientosJuego.addMouseListener(movimientosJuego);

        // AÑADE EL OBJETO MOVIMIENTOSJUEGO A LA VENTANA DEL JUEGO
        ventanaJuego.add(movimientosJuego);

        // HACE VISIBLE LA VENTANA DEL JUEGO
        ventanaJuego.setVisible(true);
    }
}
```

Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 16

## - Clase Tablero:

```
import java.awt.BasicStroke;
import java.awt.Color;
import java.awt.Font;
import java.awt.Graphics2D;

public class Tablero {
    int x, y, ancho, alto; // COORDENADAS Y DIMENSIONES DEL TABLERO

    // CONSTRUCTOR DE LA CLASE TABLERO
    public Tablero(int posX, int posY) {
        x = posX; // ASIGNA LA POSICION X
        y = posY; // ASIGNA LA POSICION Y
        ancho = 30; // ASIGNA EL ANCHO DE CADA CASILLA
        alto = 30; // ASIGNA LA ALTURA DE CADA CASILLA
    }

    // METODO PARA DIBUJAR EL TABLERO
    public void dibujar(Graphics2D graficos) {
        graficos.setColor(Color.WHITE);
        graficos.fillRect(x, y, 15 * ancho, 15 * alto); // DIBUJA EL FONDO DEL TABLERO

        // DIBUJA LAS CASILLAS PARA CADA JUGADOR
        for (int i = 0; i < 6; i++) {
            graficos.setColor(Color.RED);
            graficos.fillRect(x + (i * ancho), y, ancho, alto);
            graficos.fillRect(x, y + (i * alto), ancho, alto);
            graficos.fillRect(x + (i * ancho), y + (5 * alto), ancho, alto);
            graficos.fillRect(x + (5 * ancho), y + (i * alto), ancho, alto);
            graficos.setColor(Color.GREEN);
            graficos.fillRect(x + ((i + 9) * ancho), y, ancho, alto);
            graficos.fillRect(x + (9 * ancho), y + (i * alto), ancho, alto);
            graficos.fillRect(x + ((i + 9) * ancho), y + (5 * alto), ancho, alto);
            graficos.fillRect(x + (14 * ancho), y + (i * alto), ancho, alto);
            graficos.setColor(Color.YELLOW);
            graficos.fillRect(x + ((i + 9) * ancho), y + (9 * alto), ancho, alto);
            graficos.fillRect(x + (9 * ancho), y + ((i + 9) * alto), ancho, alto);
            graficos.fillRect(x + ((i + 9) * ancho), y + (14 * alto), ancho, alto);
            graficos.fillRect(x + (14 * ancho), y + ((i + 9) * alto), ancho, alto);
            graficos.setColor(Color.BLUE);
            graficos.fillRect(x + (i * ancho), y + (9 * alto), ancho, alto);
            graficos.fillRect(x, y + ((i + 9) * alto), ancho, alto);
            graficos.fillRect(x + (i * ancho), y + (14 * alto), ancho, alto);
            graficos.fillRect(x + (5 * ancho), y + ((i + 9) * alto), ancho, alto);
        }
    }
}
```



**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

**Aprobación:** 2022/03/01

**Código:** GUIA-PRLE-001

**Página:** 17

```
// DIBUJA EL CAMINO DE CADA JUGADOR
for (int i = 1; i < 6; i++) {
    graficos.setColor(Color.RED);
    graficos.fillRect(x + (i * ancho), y + (7 * alto), ancho, alto);
    graficos.setColor(Color.YELLOW);
    graficos.fillRect(x + ((8 + i) * ancho), y + (7 * alto), ancho, alto);
    graficos.setColor(Color.GREEN);
    graficos.fillRect(x + (7 * ancho), y + (i * alto), ancho, alto);
    graficos.setColor(Color.BLUE);
    graficos.fillRect(x + (7 * ancho), y + ((8 + i) * alto), ancho, alto);
}

// DIBUJA LAS CASILLAS ESPECIALES PARA CADA JUGADOR
graficos.setColor(Color.RED);
graficos.fillRect(x + (1 * ancho), y + (6 * alto), ancho, alto);
graficos.setColor(Color.YELLOW);
graficos.fillRect(x + (13 * ancho), y + (8 * alto), ancho, alto);
graficos.setColor(Color.GREEN);
graficos.fillRect(x + (8 * ancho), y + (1 * alto), ancho, alto);
graficos.setColor(Color.BLUE);
graficos.fillRect(x + (6 * ancho), y + (13 * alto), ancho, alto);

// DIBUJA LOS INICIOS DE CADA JUGADOR
int temp1 = x + 45, temp2 = y + 45;
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        graficos.setColor(Color.RED);
        graficos.fillRect(temp1 + (2 * i * ancho), temp2 + (2 * j * alto), ancho, alto);
        graficos.setColor(Color.YELLOW);
        graficos.fillRect(temp1 + (2 * i * ancho) + 9 * ancho, temp2 + (2 * j * alto) + 9 * alto, ancho, alto);
        graficos.setColor(Color.GREEN);
        graficos.fillRect(temp1 + (2 * i * ancho) + 9 * ancho, temp2 + (2 * j * alto), ancho, alto);
        graficos.setColor(Color.BLUE);
        graficos.fillRect(temp1 + (2 * i * ancho), temp2 + (2 * j * alto) + 9 * alto, ancho, alto);
    }
}

// DIBUJA LOS TRIANGULOS CENTRALES PARA CADA JUGADOR
graficos.setColor(Color.RED);
int puntosX0[] = { x + (6 * ancho), x + (6 * ancho), x + 15 + (7 * ancho) };
int puntosY0[] = { y + (6 * alto), y + (9 * alto), y + 15 + (7 * alto) };
int numPuntos0 = 3;
graficos.fillPolygon(puntosX0, puntosY0, numPuntos0);
```

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

**Aprobación:** 2022/03/01

**Código:** GUIA-PRLE-001

**Página:** 18

```
graficos.setColor(Color.YELLOW);
int puntosX1[] = { x + (9 * ancho), x + (9 * ancho), x + 15 + (7 * ancho) };
int puntosY1[] = { y + (6 * alto), y + (9 * alto), y + 15 + (7 * ancho) };
int numPuntos1 = 3;
graficos.fillPolygon(puntosX1, puntosY1, numPuntos1);

graficos.setColor(Color.GREEN);
int puntosX2[] = { x + (6 * ancho), x + (9 * ancho), x + 15 + (7 * ancho) };
int puntosY2[] = { y + (6 * alto), y + (6 * alto), y + 15 + (7 * ancho) };
int numPuntos2 = 3;
graficos.fillPolygon(puntosX2, puntosY2, numPuntos2);

graficos.setColor(Color.BLUE);
int puntosX3[] = { x + (6 * ancho), x + (9 * ancho), x + 15 + (7 * ancho) };
int puntosY3[] = { y + (9 * alto), y + (9 * alto), y + 15 + (7 * ancho) };
int numPuntos3 = 3;
graficos.fillPolygon(puntosX3, puntosY3, numPuntos3);

// DIBUJA LOS BORDES Y LAS LINEAS DEL TABLERO
graficos.setStroke(new BasicStroke(2));
graficos.setColor(Color.BLACK);

// DIBUJA LAS CASILLAS DE CADA JUGADOR EN EL TABLERO CENTRAL
for (int i = 0; i < 3; i++) {
    for (int j = 0; j < 6; j++) {
        graficos.drawRect(x + ((i + 6) * ancho), y + (j * alto), ancho, alto); // CASILLAS DE LA PARTE SUPERIOR DEL TABLERO CENTRAL
        graficos.drawRect(x + (j * ancho), y + ((i + 6) * alto), ancho, alto); // CASILLAS DE LA PARTE IZQUIERDA DEL TABLERO CENTRAL
        graficos.drawRect(x + ((i + 6) * ancho), y + ((j + 9) * alto), ancho, alto); // CASILLAS DE LA PARTE INFERIOR DEL TABLERO CENTRAL
        graficos.drawRect(x + ((j + 9) * ancho), y + ((i + 6) * alto), ancho, alto); // CASILLAS DE LA PARTE DERECHA DEL TABLERO CENTRAL
    }
}

// DIBUJA LOS RECTANGULOS GRANDES EN LAS ESQUINAS (INICIO DE CADA JUGADOR)
graficos.drawRect(x + (1 * ancho), y + (1 * alto), 4 * ancho, 4 * alto);
graficos.drawRect(x + (10 * ancho), y + (1 * alto), 4 * ancho, 4 * alto);
graficos.drawRect(x + (1 * ancho), y + (10 * alto), 4 * ancho, 4 * alto);
graficos.drawRect(x + (10 * ancho), y + (10 * alto), 4 * ancho, 4 * alto);
graficos.drawRect(x, y, 15 * ancho, 15 * alto); // DIBUJA EL BORDE EXTERIOR DEL TABLERO

// DIBUJA LOS RECTANGULOS PEQUEÑOS DE COLORES PARA CADA JUGADOR
for (int i = 0; i < 2; i++) {
    for (int j = 0; j < 2; j++) {
        graficos.drawRect(temp1 + (2 * i * ancho), temp2 + (2 * j * alto), ancho, alto);
    }
}
```

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

**Aprobación:** 2022/03/01

**Código:** GUIA-PRLE-001

**Página:** 19

```
graficos.drawRect(temp1 + (2 * i * ancho) + 9 * ancho, temp2 + (2 * j * alto) + 9 * alto, ancho, alto);
graficos.drawRect(temp1 + (2 * i * ancho) + 9 * ancho, temp2 + (2 * j * alto), ancho, alto);
graficos.drawRect(temp1 + (2 * i * ancho), temp2 + (2 * j * alto) + 9 * alto, ancho, alto);
}
}

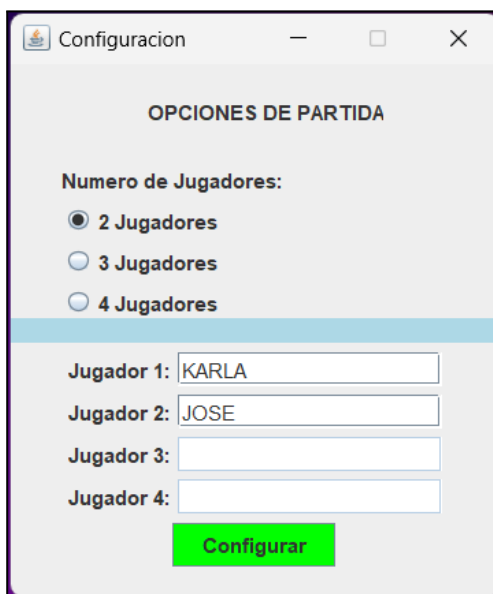
// DIBUJA LOS TRIANGULOS CENTRALES DE COLORES PARA CADA JUGADOR
graficos.drawPolygon(puntosX0, puntosY0, numPuntos0);
graficos.drawPolygon(puntosX1, puntosY1, numPuntos1);
graficos.drawPolygon(puntosX2, puntosY2, numPuntos2);
graficos.drawPolygon(puntosX3, puntosY3, numPuntos3);

// DIBUJA LAS OVBICAS DE LOS COLORES PARA CADA JUGADOR
graficos.drawOval(x + 5 + (6 * ancho), y + 5 + (2 * alto), ancho - 10, alto - 10);
graficos.drawOval(x + 5 + (12 * ancho), y + 5 + (6 * alto), ancho - 10, alto - 10);
graficos.drawOval(x + 5 + (8 * ancho), y + 5 + (12 * alto), ancho - 10, alto - 10);
graficos.drawOval(x + 5 + (2 * ancho), y + 5 + (8 * alto), ancho - 10, alto - 10);

// CONFIGURA LA FUENTE Y DIBUJA LOS TEXTOS DE INSTRUCCIONES
graficos.setFont(new Font("serif", Font.BOLD, 30));
graficos.drawString(PantallaInicial.jugador[0] + "", 90, 35);
graficos.drawString(PantallaInicial.jugador[1] + "", 370, 35);
graficos.drawString(PantallaInicial.jugador[3] + "", 90, 540);
graficos.drawString(PantallaInicial.jugador[2] + "", 370, 540);
graficos.drawString("Instrucciones:", 550, 300);
graficos.drawString("1. Lanza el dado y mueve una ficha. ", 550, 350);
graficos.drawString("2. Si obtienes un 6, puedes mover otra ficha", 550, 400);
graficos.drawString("3. Si no puedes mover, pasa el turno", 550, 450);
}
}
```

## 5. CAPTURA DE PANTALLAS

### - 2 Jugadores:



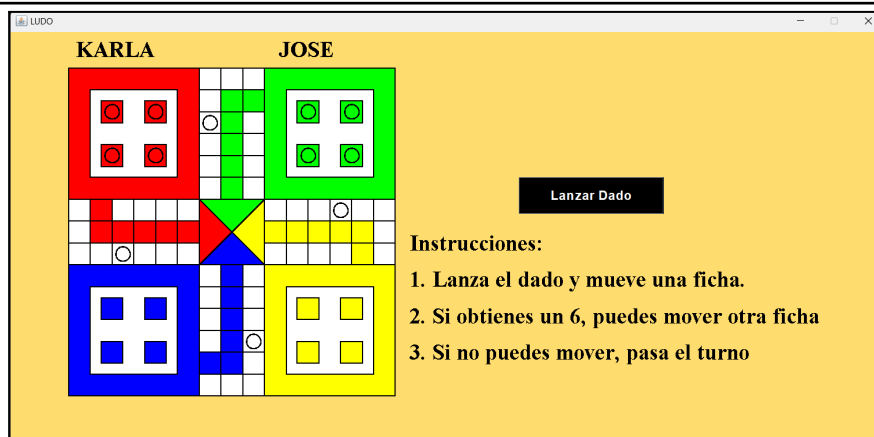
The screenshot shows a window titled "Configuracion" with a close button. Inside, the section "OPCIONES DE PARTIDA" is visible. Under "Numero de Jugadores:", there are three radio buttons: "2 Jugadores" (selected), "3 Jugadores", and "4 Jugadores". Below this, there are four text input fields for player names: "Jugador 1:" (containing "KARLA"), "Jugador 2:" (containing "JOSE"), "Jugador 3:" (empty), and "Jugador 4:" (empty). At the bottom, there is a green button labeled "Configurar".

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 20



- **3 Jugadores:**

Configuración

OPCIONES DE PARTIDA

Numero de Jugadores:  
☐ 2 Jugadores  
☒ 3 Jugadores  
☐ 4 Jugadores

Jugador 1:   
 Jugador 2:   
 Jugador 3:   
 Jugador 4:

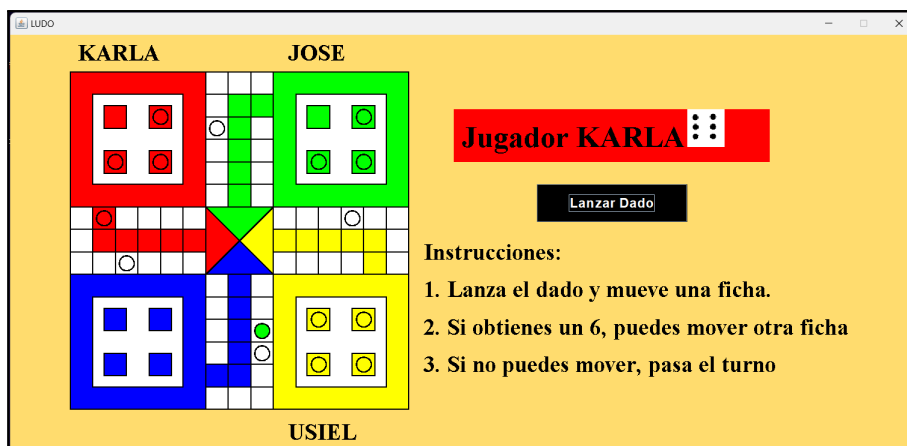
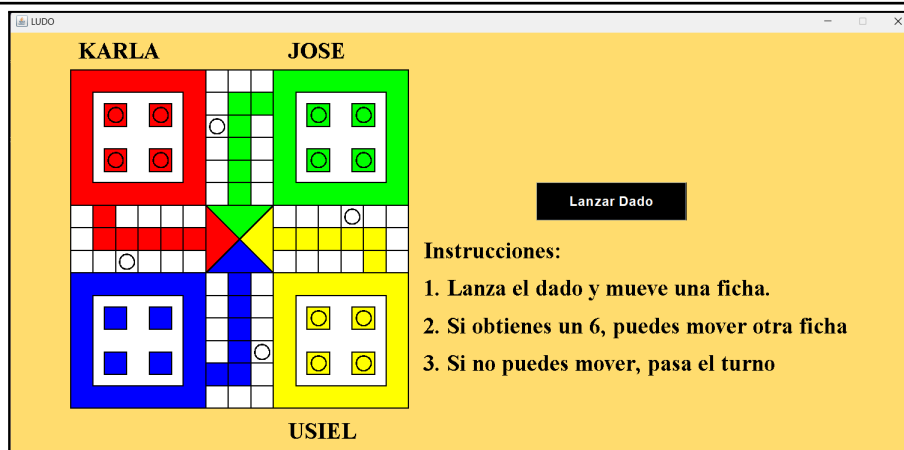
Configurar

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

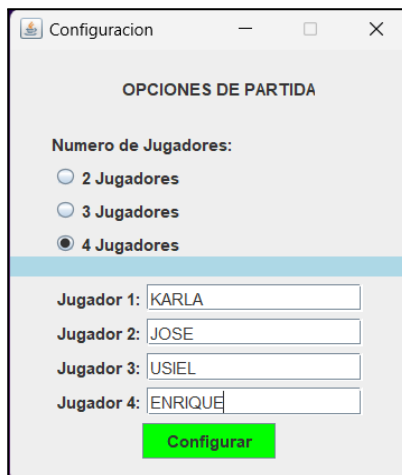
Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 21



- **4 Jugadores:**


**Configuración**

**OPCIONES DE PARTIDA**

**Numero de Jugadores:**

☐ 2 Jugadores
   
☐ 3 Jugadores
   
☒ 4 Jugadores

**Jugador 1:** 
  
**Jugador 2:** 
  
**Jugador 3:** 
  
**Jugador 4:**

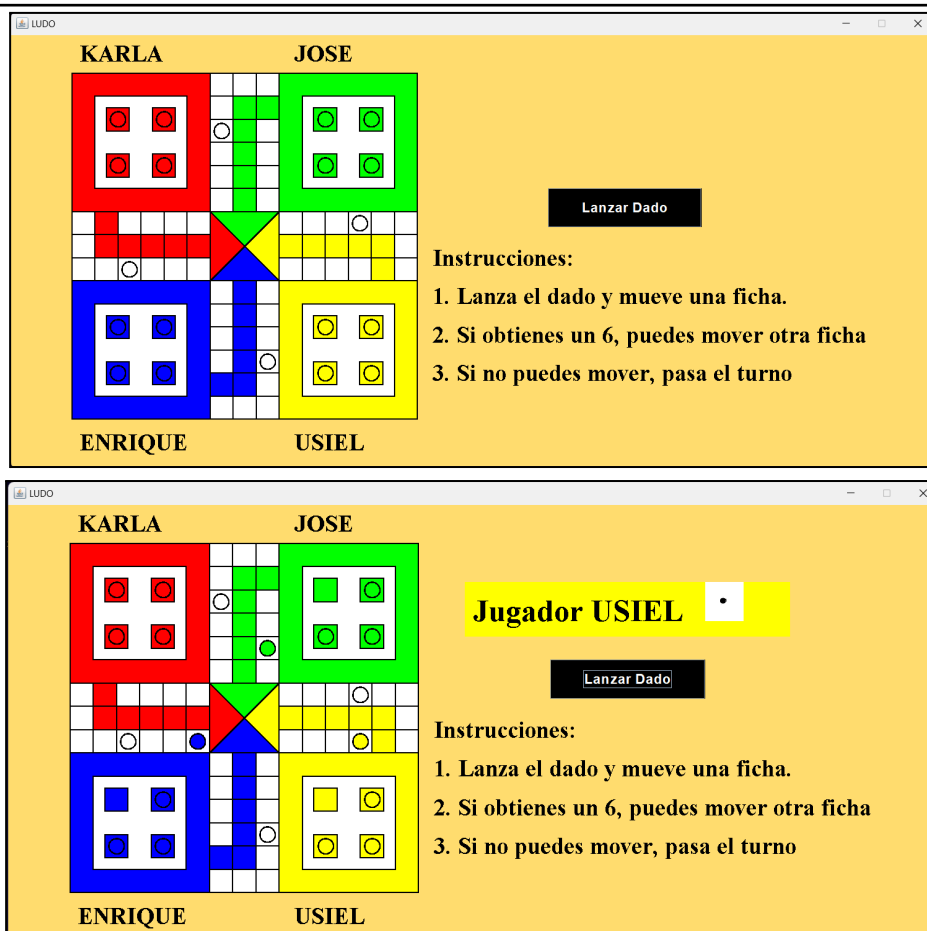
**Configurar**

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 22



## 6. MANUAL DE USUARIO

### Descripción general

**Ludo** es un juego de mesa clásico en el que los jugadores compiten para llevar sus fichas desde su casilla de inicio hasta la meta, utilizando un dado para determinar el movimiento de cada ficha. El juego se juega entre 2, 3 o 4 jugadores, y cada jugador tiene un color distinto para sus fichas.

### Requisitos

- **Cantidad de jugadores :** El juego soporta entre 2 y 4 jugadores.

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 23

## Inicio del juego

### 1. Pantalla de inicio

Cuando inicias el juego, se mostrará una **pantalla inicial** que te permitirá configurar tu partida:

1. **Selección de la cantidad de jugadores :**
  - Elige la cantidad de jugadores (2, 3 o 4) desde un menú desplegable.
2. **Asignación de nombres a los jugadores :**
  - A continuación, tendrás que ingresar los nombres de los jugadores. Cada jugador debe proporcionar su nombre, que será utilizado durante la partida.
3. **Ejemplo :**
  - Jugador 1: Juan
  - Jugador 2: María
  - Jugador 3: Carlos
  - Jugador 4: Ana
4. **Iniciar la partida :**
  - Una vez que hayas asignado los nombres y elegido la cantidad de jugadores, haz clic en el botón **“CONFIGURAR”** para comenzar el juego.

### 2. Pantalla de juego

Una vez que hayas iniciado la partida, se cargará la **pantalla del juego** . Aquí es donde ocurrirá la acción del juego, y cada jugador podrá realizar sus movimientos.

## Jugando al Ludo

### 1. Interfaz de pantalla de juego

La pantalla de juego está dividida en varias secciones:

- **Tablero :** Es el área central del juego, donde se muestra el recorrido de las fichas de los jugadores.
- **Fichas de los jugadores :** Cada jugador tiene un conjunto de fichas de un color determinado. Los colores pueden ser rojo, verde, azul y amarillo (dependiendo de la cantidad de jugadores).
- **Botón de lanzar dado :** Este botón se encuentra en la parte inferior o lateral de la pantalla. Al hacer clic en este botón o presionar la tecla **ENTER** , el jugador lanza el dado y se muestra el valor obtenido en el lanzamiento.

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 24

- **Área de información** : Muestra detalles como el turno del jugador, el valor del dado y las instrucciones del juego.

## 2. Jugando el turno

- En cada turno, el jugador debe **lanzar el dado** para determinar cuántos espacios puede mover sus fichas.
- Para lanzar el dado, puedes hacer clic en el botón **“Lanzar Dado”** o presionar la tecla **ENTER** .

## 3. Resultado del lanzamiento

- Después de lanzar el dado, el valor obtenido se mostrará en la pantalla, indicando cuántos espacios puede avanzar el jugador en el tablero.

## 4. Mover las fichas

- Los jugadores deben mover sus fichas según el número que hayan obtenido en el dado.
- Los jugadores pueden avanzar una ficha de su color o mover una ficha ya en el tablero, según las reglas del juego de Ludo.

## 5. Gana el jugador que llegue primero a la meta

- El objetivo del juego es llevar todas tus fichas a la meta antes que los demás jugadores. El primer jugador en conseguirlo será el **ganador** .
- Una vez que un jugador haya movido todas sus fichas a la meta, se mostrará un mensaje que indica que ese jugador ha ganado.

## 7. TRABAJO FUTURO

En un futuro, se podrían hacer varias mejoras al juego para que sea más completo y atractivo. Por ejemplo, se puede trabajar en una interfaz gráfica más moderna y dinámica, con un diseño más estilizado que haga la experiencia del usuario más agradable. También se podrían incluir animaciones para los movimientos de las fichas, los lanzamientos de dado y los eventos importantes del juego, lo que le daría un toque más visual y emocionante.

Además, se podría implementar un sistema de sonidos para acompañar las acciones del juego, como un efecto al lanzar el dado o al ganar la partida. También sería interesante añadir modos de juego alternativos o variantes de



	<p style="text-align: center;"><b>UNIVERSIDAD NACIONAL DE SAN AGUSTIN</b>  <b>FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS</b>  <b>ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</b></p>	
<p style="text-align: center;"><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p><b>Aprobación:</b> 2022/03/01</p>	<p><b>Código:</b> GUIA-PRLE-001</p>	<p><b>Página:</b> 25</p>

reglas para que los jugadores tengan más opciones. Incluso, en un futuro más ambicioso, se podría desarrollar una versión en línea para que los jugadores puedan competir a distancia con amigos o desconocidos.

**CONCLUSIONES**

En este trabajo final aprendimos mucho sobre cómo usar las interfaces gráficas (GUI) para crear algo más interactivo y llamativo, en este caso, un videojuego. Hacer el juego de Ludo fue todo un reto porque no solo aplicamos conceptos básicos de programación, como los ciclos y la generación de números aleatorios para el dado, sino que también trabajamos con cosas más avanzadas, como coordinar los turnos de los jugadores y mostrar todo de forma visual.

Nos dimos cuenta de lo importante que es planificar bien cada parte del proyecto, ya que todo debía encajar: desde el diseño del tablero hasta las funciones que controlan el movimiento de las fichas. Este proyecto nos dejó una gran experiencia al combinar programación con diseño, y también nos hizo ver cómo se pueden llevar ideas simples a algo más completo.

Al final, esta experiencia nos motiva a seguir aprendiendo y pensar en proyectos más ambiciosos, donde podamos usar todo lo que aprendimos aquí, mezclando técnica y creatividad para hacer aplicaciones que sean útiles, entretenidas o ambas.

**METODOLOGÍA DE TRABAJO**

**Plantamiento del Juego :**

El objetivo fue crear un juego de Ludo que pudiera ser jugado por dos o más personas en una misma computadora. Para programar el juego, se utilizó Java junto con el entorno de desarrollo NetBeans, lo que permitió crear una interfaz gráfica. El juego permitía a los jugadores lanzar los dados, mover sus fichas por el tablero y avanzar hacia la meta, replicando la dinámica tradicional del Ludo.

**Diseño del Tablero y la Interfaz :**

Se diseñó un tablero visual usando un panel con casillas numeradas. Cada jugador tenía su propio color y una base de inicio en el tablero, lo que facilitaba la identificación de las fichas de cada jugador. La interfaz incluía botones para lanzar el dado y mover las fichas, lo que hacía que la interacción con el juego fuera intuitiva y sencilla para los jugadores.

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 26

### **Estructura del Código :**

El código se actuará en varias clases para mejorar su modularidad. Se creó una clase principal para el juego, encargada de manejar la lógica central. Además, se desarrollaron clases para los jugadores, donde se guardaba información como el nombre, las fichas y el turno. La clase del tablero controlaba las casillas y los caminos de las fichas. También se implementaron clases específicas para las fichas, que representaban las piezas de los jugadores, y para las posiciones de las casillas, lo que permitiría determinar en qué casilla se encontraba cada ficha.

### **Desarrollo y Pruebas :**

El desarrollo comenzó con la creación de la interfaz gráfica, que incluía el tablero y los botones necesarios para interactuar con el juego. Posteriormente, se integró la lógica para los dados y el movimiento de las fichas. Finalmente, se realizaron pruebas exhaustivas para garantizar que todo funcionará correctamente, asegurándose de que el juego fuera divertido y libre de errores.

Este enfoque permitió no solo desarrollar el juego de manera efectiva, sino también asegurar una experiencia de usuario fluida y entretenida.

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p>Formato: Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 27</p>

## Commits:

```

Usuario248@DESKTOP-6V821ET MINGW64 ~/Desktop/repositorio/Proyecto-Final---Fundam
entos (master)
$ git log
commit 96b010a2ad411cae7c2d9f80e8f56403df96ed99 (HEAD -> master, origin/master,
origin/HEAD)
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Dec 13 16:55:57 2024 -0500

    ACTUALIZACION FINAL

commit 14c1ac0543e0dcee64decde1e4e11eb1ccf5febc
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Dec 13 16:54:55 2024 -0500

    ACTUALIZACION FINAL

commit 025dd82bdc7702009d4e701693c7d9cae3e57261
Author: usiel33 <uquispep@unsa.edu.pe>
Date: Fri Dec 13 16:03:22 2024 -0500

    Se sube el informe final

commit 5b41c7d9c8cc46ab23192d9938d97de29ffa96ca
Author: KarlaBedregal <kbedregal@unsa.edu.pe>
Date: Thu Dec 12 20:18:26 2024 -0500

...skipping...
commit 96b010a2ad411cae7c2d9f80e8f56403df96ed99 (HEAD -> master, origin/master, origin/HEAD)
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Dec 13 16:55:57 2024 -0500

    ACTUALIZACION FINAL

commit 14c1ac0543e0dcee64decde1e4e11eb1ccf5febc
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Fri Dec 13 16:54:55 2024 -0500

    ACTUALIZACION FINAL

commit 025dd82bdc7702009d4e701693c7d9cae3e57261
Author: usiel33 <uquispep@unsa.edu.pe>
Date: Fri Dec 13 16:03:22 2024 -0500

    Se sube el informe final

commit 5b41c7d9c8cc46ab23192d9938d97de29ffa96ca
Author: KarlaBedregal <kbedregal@unsa.edu.pe>
Date: Thu Dec 12 20:18:26 2024 -0500

    añadiendo datos, botones, cambiando diseño y añadiendo color

commit bdae46f0b772fefb84ff1796ab53f61ea54a87c1
Author: leonhatches <jhatches@unsa.edu.pe>
Date: Thu Dec 12 01:05:37 2024 -0500

    Version final y funcional de la configuracion Inicial y seleccion de personajes en cantidad

commit 52aa40330a48948460a276448e60e5a4605b47ef
Author: leonhatches <jhatches@unsa.edu.pe>
Date: Thu Dec 12 00:13:06 2024 -0500

    Pantalla Inicial Funcional con 4 jugadores

```

	<p>UNIVERSIDAD NACIONAL DE SAN AGUSTIN FACULTAD DE INGENIERÍA DE PRODUCCIÓN Y SERVICIOS ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMA</p>	
<p><b>Formato:</b> Guía de Práctica de Laboratorio / Talleres / Centros de Simulación</p>		
<p>Aprobación: 2022/03/01</p>	<p>Código: GUIA-PRLE-001</p>	<p>Página: 28</p>

```
commit ee9487c982f8d1d4f70d4e47bb3a3e7c09aa9406
Author: leonhatches <jhatches@unsa.edu.pe>
Date: Wed Dec 11 21:05:52 2024 -0500

    Version Mejorada de la Estructura de la Pantalla Inicial

commit ca7ddc068b12994ab9bbb5531a3f3e62dc1bb029
Author: leonhatches <jhatches@unsa.edu.pe>
Date: Wed Dec 11 20:19:57 2024 -0500

    Pantalla Inicial - Estructural

commit 757df627e06139758d42bae0e2cbd9078b81e526
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Wed Dec 11 01:57:29 2024 -0500

    Se agrega La pantalla del juego que sera el main

commit dc63c9d9384aeb6eaa359a00b05501464347609f
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Wed Dec 11 01:26:48 2024 -0500

    Se agrega la logica de los movimientos realizables en el juego

commit c2feb9022d8cb752cac370d5b9e299af188cc5a2
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Wed Dec 11 01:08:09 2024 -0500

    Se agrega el tablero del LUDO terminado y mejorado

commit fd5f3d3a4fc90d3560b9e3525e68a5e2518ae7a4
Author: JoseMorocco <jmoroccosa@unsa.edu.pe>
Date: Tue Dec 10 23:38:56 2024 -0500

    Se agrega la clase CrearJugadores con psiciones inicales para cada jugador y sus fichas
```

LINK AL REPOSITORIO:

<https://github.com/KarlaBedregal/Proyecto-Final---Fundamentos>

**Formato:** Guía de Práctica de Laboratorio / Talleres / Centros de Simulación

Aprobación: 2022/03/01

Código: GUIA-PRLE-001

Página: 29

## REFERENCIAS Y BIBLIOGRAFÍA

### INVESTIGACIÓN Graphics2D :

- [1] Demostraciones 2. (sin fecha). *Tutorial de Java Graphics2D con ejemplos* . Recuperado el 13 de diciembre de 2024 de <https://www.demo2s.com/java/java-graphics2d-tutorial-with-examples.html>
- [2] Reingeniería Tecnológica. (Dakota del Norte). *Gráficos 2D en Java: Cómo dibujar formas, texto e imágenes* . Recuperado el 13 de diciembre de 2024 de <https://reintech.io>
- [3] Tutoriales Programación Ya. (sin fecha). *Gráficos en Java* . Recuperado el 13 de diciembre de 2024 de [https://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?codigo=130&punto=&inicio=40#google\\_vignette](https://www.tutorialesprogramacionya.com/javaya/detalleconcepto.php?codigo=130&punto=&inicio=40#google_vignette)
- [4] Universidad Politécnica de Madrid. (sin fecha). *Java 2D: Gráficos en Java* . Recuperado el 13 de diciembre de 2024 de [https://laurel.datsi.fi.upm.es/\\_media/docencia/cursos/java/java2d.pdf](https://laurel.datsi.fi.upm.es/_media/docencia/cursos/java/java2d.pdf)

## RUBRICA DE EVALUACIÓN:

Contenido y demostración		Puntos	Checklis t	Estudiant e	Profeso r
1. GitHub	Hay enlace URL activo del directorio para el laboratorio hacia su repositorio GitHub con código fuente terminado y fácil de revisar.	2	X	2	
2. Commits	Hay capturas de pantalla de los commits más importantes con sus explicaciones detalladas. (El profesor puede preguntar para refrendar calificación).	4	X	2	
3. Código fuente	Hay porciones de código fuente importantes con numeración y explicaciones detalladas de sus funciones.	2	X	1	
4. Ejecución	Se incluyen ejecuciones/pruebas del código fuente explicadas gradualmente.	2	X	2	
5. Pregunta	Se responde con completitud a la pregunta formulada en la tarea. (El profesor puede preguntar para refrendar calificación).	2	X	1	
6. Fechas	Las fechas de modificación del código fuente están dentro de los plazos de fecha de entrega establecidos.	2	X	2	
7. Ortografía	El documento no muestra errores ortográficos.	2	X	2	
8. Madurez	El Informe muestra de manera general una evolución de la madurez del código fuente, explicaciones puntuales pero precisas y un acabado impecable. (El profesor puede preguntar para refrendar calificación).	4	X	4	
TOTAL		20		16	