MANUAL DEL DESARROLLADOR

POR

JAVIER SANTIAGO BORBÓN BORBÓN Y JOSÉ ANDRÉS SANABRIA ARIAS

Tabla de contenido

INTRODUCCIÓN

En este documento usted encontrará información que le ayudará a comprender mejor este proyecto realizado en Java, específicamente en el IDE de Eclipse. Con una versión de Java SE-13.

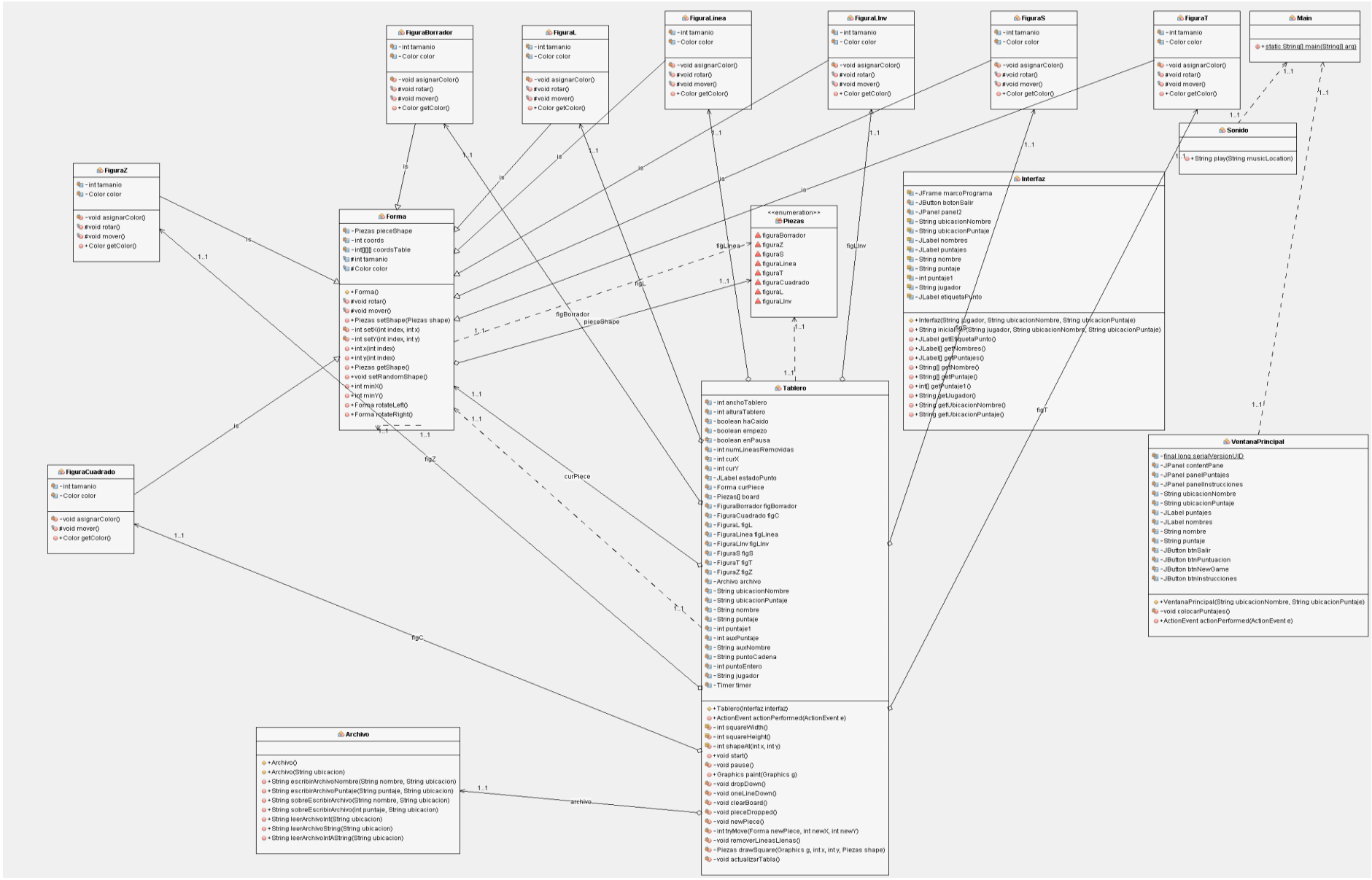
Así como se evidenció en la tabla de contenido, se le mostrará primero el diagrama UML, donde podrá evidenciar las clases utilizadas y la relación que tienen estas. Se le mostrará el índice de los archivos en el IDE y por último se le anexará cada una de las clases con su respectiva descripción.

Esperemos le sea de utilidad este manual. Y que pueda modificar el código de la mejor manera.

Adicional a esto, hemos documentado el programa, usted puede acceder a la información abriendo el archivo “Index” ubicado en la carpeta doc, en este documento se le especificará más.

Para mayor información, inquietudes y dudes, por favor escribanos al siguiente correo: mjoasanabriaa@correo.udistrital.edu.co

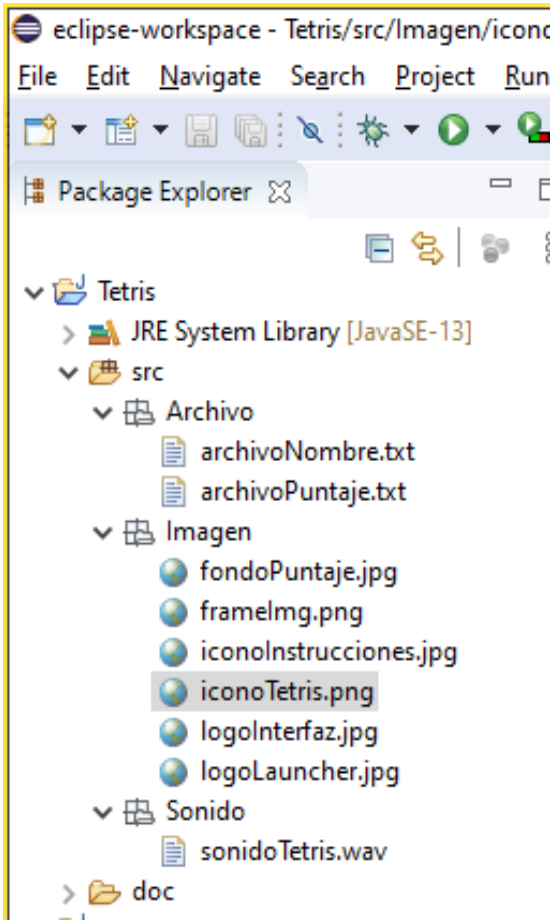
ARCHIVO UML



Para mejor visualización, este documento se encuentra en la misma carpeta de este documento.

ELEMENTOS DEL PROYECTO

Este proyecto consta de los siguientes elementos:



En la imagen anterior se ven las carpetas de los recursos que utiliza el juego.

La carpeta Archivo contiene los archivos relacionados al nombre y al puntaje.

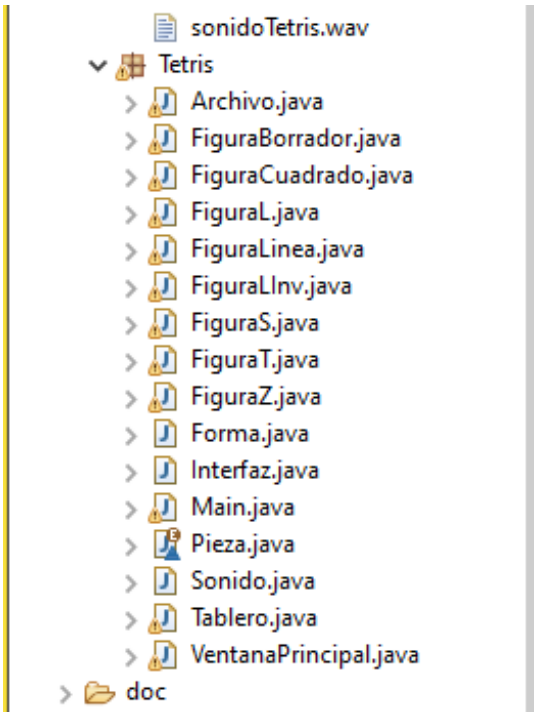
-archivoNombre.txt: Guarda los nombres en un documento de texto.
-archivoPuntaje.txt: Guarda los puntos (como String) en un documento de texto.

La carpeta Imagen contiene las diferentes imágenes que contiene el programa.

-fondoPuntaje.jpg: Corresponde a la imagen del fondo de la tabla de puntuación.
-frameImg.png: Corresponde a los bordes que tiene el tablero del juego.
-iconoInstrucciones.jpg: Corresponde a la imagen del panel de Instrucciones.
-iconoTetris.png: Corresponde al icono que aparece en la barra de tareas y al archivo .jar
-logoInterfaz.jpg: Corresponde a la imagen del panel derecho del juego.
-logoLauncher.jpg: Corresponde a la imagen que aparece en la ventana inicial.

La carpeta Sonido contiene el audio del juego: sonidoTetris.wav.

La carpeta: doc contiene todo lo relacionado a la documentación del proyecto



El paquete Tetris contiene toda la lógica del juego.

El proyecto cuenta con tres clases encargas de la parte gráfica, estas son: VentanaPrincipal.java, Tablero.java e Interfaz.java

Una clase encargada de la gestión del sonido: Sonido.java

Una clase encargada de la gestión de Archivos: Archivo.java

Una clase encargada de la ubicación de las dos clases anteriores y de instanciar a la ventana inicial: Main.java

La clase Pieza.java encargada de enlistar las ocho figuras (siete del Tetris y una que borra).

Y las demás clases, encargadas de la lógica del juego, a continuación, se detallarán cada una de las clases.

Archivo.java

```java
package Tetris;

 * NOMBRE DEL PROYECTO: TETRIS

import java.io.BufferedReader;

public class Archivo {

    public Archivo() {

    }

    // Método constructor, el cual crea el archivo, lo abre y cierra.
    public Archivo(String ubicacion) {

        DataOutputStream archivo = null;

        try {

            archivo = new DataOutputStream(new FileOutputStream(ubicacion, true));

            archivo.close();

        } catch (FileNotFoundException fnfe) {

        } catch (IOException ioe) {

        }

    }

    // Métodos que se utilizaron para crear el primer archivo
    public void escribirArchivoNombre(String nombre[], String ubicacion) {

        DataOutputStream archivo = null;

        try {

            archivo = new DataOutputStream(new FileOutputStream(ubicacion, true));

            for (int i = 0; i < 10; i++) {

                archivo.writeUTF(nombre[i]);

            }

            archivo.close();

        } catch (FileNotFoundException fnfe) {

        } catch (IOException ioe) {

        }

    }
```

```java
    // Métodos que se utilizaron para crear el primer archivo
    public void escribirArchivoPuntaje(String puntaje[], String ubicacion) {

        int[] puntaje1 = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };

        DataOutputStream archivo = null;

        try {

            archivo = new DataOutputStream(new FileOutputStream(ubicacion, true));

            for (int i = 0; i < 10; i++) {

                puntaje1[i] = Integer.parseInt(puntaje[i]);

            }

            for (int i = 0; i < 10; i++) {

                archivo.writeInt(puntaje1[i]);

            }

            archivo.close();

        } catch (FileNotFoundException fnfe) {

        } catch (IOException ioe) {

        }

    }

    // Método que hace la persistencia de los puntajes al final del juego
    public void sobreEscribirArchivo(String nombre[], String ubicacion) {

        // DataOutputStream archivo = null;

        try {

            // archivo = new DataOutputStream( new FileOutputStream(ubicacion,true));

            RandomAccessFile archivo = new RandomAccessFile(ubicacion, "rw");

            archivo.seek(0);

            for (int i = 0; i < 10; i++) {

                archivo.writeUTF(nombre[i]);

            }

            archivo.close();

        } catch (FileNotFoundException fnfe) {

        } catch (IOException ioe) {

        }

    }
```

```
141
142     // Método que hace la persistencia de los puntajes al final del juego
143     public void sobreEscribirArchivo(int puntaje[], String ubicacion) {
144
145         try {
146
147             RandomAccessFile archivo = new RandomAccessFile(ubicacion, "rw");
148
149             archivo.seek(0);
150
151             for (int i = 0; i < 10; i++) {
152
153                 archivo.writeInt(puntaje[i]);
154
155             }
156
157             archivo.close();
158
159         } catch (FileNotFoundException fnfe) {
160
161         } catch (IOException ioe) {
162
163         }
164
165     }
166
167     // Método que leen el archivo, y tranfieren sus datos a variables para su uso.
168     public int[] leerArchivoInt(String ubicacion) {
169
170         DataInputStream archivo = null;
171         int[] puntaje = { 0, 0, 0, 0, 0, 0, 0, 0, 0, 0 };
172
173         try {
174
175             archivo = new DataInputStream(new FileInputStream(ubicacion));
176
177             for (int i = 0; i < 10; i++) {
178
179                 puntaje[i] = archivo.readInt();
180
181             }
182
183         } catch (FileNotFoundException fnfe) {
184
185         } catch (IOException ioe) {
186
187         }
188
189         return puntaje;
190
191     }
192
193     // Método que leen el archivo, y tranfieren sus datos a variables para su uso.
194     public String[] leerArchivoString(String ubicacion) {
195
196         DataInputStream archivo = null;
197         String[] nombre = { null, null, null, null, null, null, null, null, null, null };
198
199         try {
200
201             archivo = new DataInputStream(new FileInputStream(ubicacion));
202
203             for (int i = 0; i < 10; i++) {
204
205                 nombre[i] = archivo.readUTF();
206
207             }
208
209         } catch (FileNotFoundException fnfe) {
210
211         } catch (IOException ioe) {
212
213         }
214
215         return nombre;
216
217     }
```

```
218
219     // Método que leen el archivo, y tranfieren sus datos a variables para su uso.
220     public String[] leerArchivoIntAString(String ubicacion) {
221
222         int puntajeEntero[] = leerArchivoInt(ubicacion);
223
224         String[] puntaje = { null, null, null, null, null, null, null, null, null, null };
225
226         for (int i = 0; i < 10; i++) {
227
228             puntaje[i] = Integer.toString(puntajeEntero[i]);
229
230         }
231
232         return puntaje;
233
234     }
235
236 }
```

FiguraBorrador.java

```java
package Tetris;

import java.awt.Color;

public class FiguraBorrador extends Forma {

    private int tamanio;
    private Color color;

    private void asignarColor(){
    // Asignación de colores a las figuras
    Color negro = new Color(0, 0, 0); // negro

    this.color=negro;
    }

    public Color getColor() {
        asignarColor();
        return color;
    }


}

```

FiguraCuadrado.java

```java
package Tetris;

import java.awt.Color;

public class FiguraCuadrado extends Forma {

    private int tamanio;
    private Color color;

    private void asignarColor(){
    // Asignación de colores a las figuras
    Color azul = new Color(12, 67, 194); // Cuadrado
    this.color=azul;
    }

    protected void mover() {

    }
    public Color getColor() {
        asignarColor();
        return color;
    }


}
```

FiguraL.java

```java
package Tetris;

import java.awt.Color;

public class FiguraL extends Forma {

    private int tamanio;
    private Color color;
    private void asignarColor(){
    // Asignación de colores a las figuras

    Color magenta = new Color(205, 46, 150); // L
    this.color=magenta;
    }
    protected void rotar() {

    }

    protected void mover() {

    }
    public Color getColor() {
        asignarColor();
        return color;
    }


}
```

FiguraLinea.java

```java
package Tetris;

import java.awt.Color;

public class FiguraLinea extends Forma {

    private int tamanio;
    private Color color;

    private void asignarColor(){
    // Asignación de colores a las figuras
    Color rojo = new Color(216, 29, 43); // Linea
    this.color=rojo;
    }
    protected void rotar() {

    }

    protected void mover() {

    }
    public Color getColor() {
        asignarColor();
        return color;
    }


}
```

FiguraLInv.java

```java
package Tetris;

import java.awt.Color;

public class FiguraLInv extends Forma {

    private int tamanio;
    private Color color;
    private void asignarColor(){
    // Asignación de colores a las figuras

    Color blanco = new Color(255, 255, 255); // L invertida
    this.color=blanco;
    }
    protected void rotar() {

    }

    protected void mover() {

    }
    public Color getColor() {
        asignarColor();
        return color;
    }


}

```

FiguraS.java

```java
package Tetris;

import java.awt.Color;

public class FiguraS extends Forma {

    private int tamanio;
    private Color color;
    private void asignarColor(){
    // Asignación de colores a las figuras

    Color verde = new Color(5, 162, 11); // S
    this.color=verde;
    }
    protected void rotar() {

    }

    protected void mover() {

    }
    public Color getColor() {
        asignarColor();
        return color;
    }


}
```

FiguraT.java

```java
package Tetris;

import java.awt.Color;

public class FiguraT extends Forma {

    private int tamanio;
    private Color color;

    private void asignarColor() {
        // Asignación de colores a las figuras

        Color marron = new Color(158, 114, 15); // T
        this.color = marron;
    }

    protected void rotar() {

    }

    protected void mover() {

    }

    public Color getColor() {
        asignarColor();
        return color;
    }

}
```

FiguraZ.java

```java
package Tetris;

import java.awt.Color;

public class FiguraZ extends Forma {

    private int tamanio;
    private Color color;
    private void asignarColor(){
    // Asignación de colores a las figuras
    Color cyan = new Color(0, 132, 144); // Z

    this.color=cyan;
    }
    protected void rotar() {

    }

    protected void mover() {

    }
    public Color getColor() {
        asignarColor();
        return color;
    }


}
```

Forma.java

```java
package Tetris;

 * NOMBRE DEL PROYECTO: TETRIS

import java.util.Random;

public class Forma {
    // Varibles fundamentales para manipular las piezas en el tablero, mediante su
    // forma y coordenadas.

    private Pieza pieceShape;
    private int coords[][];
    private int[][][] coordsTable;

    protected int tamanio;
    protected Color color;

    protected void rotar() {
        //MÉTODO POR EL CUAL LAS FICHAS ROTAN
    }

    protected void mover() {
        //MÉTODO POR EL CUAL LAS FICHAS SE MUEVEN
    }


    // Método constructor que genera un contorno de pieza en el tablero. Ya en el
    // tablero toma una forma gracias al ramdon.
    public Forma() {

        coords = new int[4][2];
        setShape(Pieza.figuraBorrador);

    }

    // Método fundamental que crea una pieza con su forma, ubicacion.
    public void setShape(Pieza shape) {

        coordsTable = new int[][][] { { { 0, 0 }, { 0, 0 }, { 0, 0 }, { 0, 0 } },
                { { 0, -1 }, { 0, 0 }, { -1, 0 }, { -1, 1 } }, { { 0, -1 }, { 0, 0 }, { 1, 0 }, { 1, 1 } },
                { { 0, -1 }, { 0, 0 }, { 0, 1 }, { 0, 2 } }, { { -1, 0 }, { 0, 0 }, { 1, 0 }, { 0, 1 } },
                { { 0, 0 }, { 1, 0 }, { 0, 1 }, { 1, 1 } }, { { -1, -1 }, { 0, -1 }, { 0, 0 }, { 0, 1 } },
                { { 1, -1 }, { 0, -1 }, { 0, 0 }, { 0, 1 } } };

        for (int i = 0; i < 4; i++) {
            for (int j = 0; j < 2; ++j) {
                coords[i][j] = coordsTable[shape.ordinal()][i][j]; //CoordsTable guarda el #de la ficha segun la clase Piezas
            }
        }
        pieceShape = shape;

    }

    private void setX(int index, int x) {
        coords[index][0] = x;
    }

    private void setY(int index, int y) {
        coords[index][1] = y;
    }

```

```
    public int x(int index) {
        return coords[index][0];
    }

    public int y(int index) {
        return coords[index][1];
    }

    public Pieza getShape() {
        return pieceShape;
    }

    // Método por el cual crea una pieza aleatoria de 7 posibles.
    public void setRandomShape() {
        Random r = new Random();
        int x = Math.abs(r.nextInt()) % 7 + 1;
        Pieza[] values = Pieza.values();
        setShape(values[x]);
    }

    // Método que devuelve el mínimo de la ficha en X.
    public int minX() {
        int m = coords[0][0];
        for (int i = 0; i < 4; i++) {
            m = Math.min(m, coords[i][0]);
        }
        return m;
    }

    // Método que devuelve el mínimo de la ficha en Y.
    public int minY() {
        int m = coords[0][1];
        for (int i = 0; i < 4; i++) {
            m = Math.min(m, coords[i][1]);
        }
        return m;
    }

     // Método que rota la pieza
     public Forma rotateLeft() {
         if (pieceShape == Pieza.figuraCuadrado)
             return this;

         Forma result = new Forma();
         result.pieceShape = pieceShape;

         for (int i = 0; i < 4; ++i) {
             result.setX(i, y(i));
             result.setY(i, -x(i));
         }
         return result;
     }

     // Método que rota la pieza
     public Forma rotateRight() {
         if (pieceShape == Pieza.figuraCuadrado)
             return this;

         Forma result = new Forma();
         result.pieceShape = pieceShape;

         for (int i = 0; i < 4; ++i) {
             result.setX(i, -y(i));
             result.setY(i, x(i));
         }
         return result;
     }
}
```

Interfaz.java

```java
package Tetris;

 * NOMBRE DEL PROYECTO: TETRIS

import java.awt.Color;

public class Interfaz {

    // Objetos para la interfaz, ventana,paneles y botón.
    JFrame marcoPrograma;
    private JButton botonSalir;
    private JPanel panel2;

    // Objetos para la persistencia mediante Archivos.
    String ubicacionNombre = null;
    String ubicacionPuntaje = null;
    JLabel nombres[] = new JLabel[10];
    JLabel puntajes[] = new JLabel[10];
    String nombre[];
    String puntaje[];
    int puntaje1[];
    String jugador = null;
    JLabel etiquetaPunto;

    // El constructor de la clase Interfaz la cual contiene el método Inicializador
    public Interfaz(String jugador, String ubicacionNombre, String ubicacionPuntaje) {
        inicializar(jugador, ubicacionNombre, ubicacionPuntaje);

    }

    // Método inicializador, acá pasa toda la magia.
    public void inicializar(String jugador, String ubicacionNombre, String ubicacionPuntaje) {

        // Asignación del nombre del jugador a la variable de la clase Interfaz.

        this.jugador = jugador;

        // Creación de la ventana, con sus dimensiones, icono y estableciendo el manejo
        // de los paneles.
        marcoPrograma = new JFrame();
        marcoPrograma.setUndecorated(true);
        marcoPrograma.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        marcoPrograma.getContentPane().setBackground(Color.BLACK);
        marcoPrograma.setBounds(0, 0, 600, 400);
        marcoPrograma.setLocationRelativeTo(null);
        marcoPrograma
                .setIconImage(Toolkit.getDefaultToolkit().getImage(Main.class.getResource("/Imagen/iconoTetris.png")));
        marcoPrograma.getContentPane().setLayout(null);

        // Asignación de las ubicaciones de los archivos a las variables de la clase
        // Interfaz.
        this.ubicacionNombre = ubicacionNombre;
        this.ubicacionPuntaje = ubicacionPuntaje;

        // Creación de los archivos.
        Archivo archivoNombre = new Archivo(ubicacionNombre);
        Archivo archivoPuntaje = new Archivo(ubicacionPuntaje);

```

```
82          // Asignación de los datos de los archivos para trabajar durante todo el
83          // programa
84          nombre = archivoNombre.leerArchivoString(ubicacionNombre);
85          puntaje = archivoPuntaje.leerArchivoIntAString(ubicacionPuntaje);
86          puntaje1 = archivoPuntaje.leerArchivoInt(ubicacionPuntaje);
87
88          // Creación del panel 2 con sus respectivas etiquetas, texto y botón. El panel 2
89          // se encarga de llevar el nombre del jugador, el puntaje del jugador y el botón
90          // para salir del juego.
91          panel2 = new JPanel();
92          panel2.setBounds(237, 0, 200, 420);
93          panel2.setBackground(Color.DARK_GRAY);
94          marcoPrograma.getContentPane().add(panel2);
95          panel2.setLayout(null);
96
97          // Logo que aparece en el panel derecho
98          JLabel lblLogo = new JLabel("");
99          lblLogo.setVerticalAlignment(SwingConstants.TOP);
100         lblLogo.setIcon(new ImageIcon(Interfaz.class.getResource("/Imagen/logoInterfaz.jpg")));
101         lblLogo.setBounds(10, 11, 180, 101);
102         panel2.add(lblLogo);
103
104         botonSalir = new JButton("SALIR");
105         botonSalir.setBackground(Color.BLACK);
106         botonSalir.setBounds(43, 346, 80, 36);
107         botonSalir.setForeground(Color.RED);
108         botonSalir.addActionListener(new ActionListener() {
109             // Cuando se le da click al botón de salir se cierra el juego.
110             public void actionPerformed(ActionEvent e) {
111
112                 System.exit(0);
113             }
114         });
115         panel2.add(botonSalir);
116
117         etiquetaPunto = new JLabel("0");
118         etiquetaPunto.setForeground(Color.CYAN);
119         etiquetaPunto.setFont(new Font("Segoe UI", Font.PLAIN, 25));
120         etiquetaPunto.setBackground(Color.BLACK);
121         etiquetaPunto.setBounds(73, 266, 117, 36);
122         panel2.add(etiquetaPunto);
123
124         JLabel lblHighScoreTitle = new JLabel("PUNTAJE M\u00C1S ALTO:");
125         lblHighScoreTitle.setForeground(Color.PINK);
126         lblHighScoreTitle.setBounds(10, 123, 164, 14);
127         panel2.add(lblHighScoreTitle);
128
129         JLabel lblScore = new JLabel("999999 puntos");
130         lblScore.setForeground(Color.CYAN);
131         lblScore.setFont(new Font("Dialog", Font.PLAIN, 20));
132         lblScore.setBounds(20, 148, 170, 36);
133         lblScore.setText(puntaje[0] + " puntos");
134         panel2.add(lblScore);
135
136         JLabel etiquetaPlayer = new JLabel("JUGADOR:");
137         etiquetaPlayer.setForeground(Color.PINK);
138         etiquetaPlayer.setBackground(Color.BLACK);
139         etiquetaPlayer.setBounds(55, 196, 68, 23);
```

```
135
136         JLabel etiquetaPlayer = new JLabel("JUGADOR:");
137         etiquetaPlayer.setForeground(Color.PINK);
138         etiquetaPlayer.setBackground(Color.BLACK);
139         etiquetaPlayer.setBounds(55, 196, 68, 23);
140         panel2.add(etiquetaPlayer);
141
142         JLabel etiquetaJugador = new JLabel("<dynamic>");
143         etiquetaJugador.setHorizontalAlignment(SwingConstants.CENTER);
144         etiquetaJugador.setForeground(Color.CYAN);
145         etiquetaJugador.setFont(new Font("Tahoma", Font.PLAIN, 25));
146         etiquetaJugador.setBackground(Color.BLACK);
147         etiquetaJugador.setBounds(20, 217, 151, 23);
148         etiquetaJugador.setText(jugador);
149         panel2.add(etiquetaJugador);
150
151         JLabel etiquetaPuntaje2 = new JLabel("PUNTAJE");
152         etiquetaPuntaje2.setForeground(Color.PINK);
153         etiquetaPuntaje2.setBackground(Color.BLACK);
154         etiquetaPuntaje2.setBounds(55, 251, 68, 23);
155         panel2.add(etiquetaPuntaje2);
156
157         JLabel logo = new JLabel("");
158         logo.setVerticalAlignment(SwingConstants.TOP);
159         logo.setIcon(new ImageIcon(Interfaz.class.getResource("/Imagen/frameImg.png")));
160         logo.setBounds(0, 0, 237, 420);
161         marcoPrograma.getContentPane().add(logo);
162
163         // Creación del panel central, panel 3 o tablero. Este tablero se encarga del
164         // juego.
165         Tablero tablero = new Tablero(this);
166         tablero.setBounds(18, 0, 200, 400);
167         marcoPrograma.getContentPane().add(tablero);
168         panel2.add(lblScore);
169
170         // Inicia el juego.
171         tablero.start();
172
173         marcoPrograma.setSize(438, 420);
174
175     }
176
177     // Métodos getters para que Tablero realice sus respectivos procesos.
178
179     public JLabel getEtiquetaPunto() {
180         return etiquetaPunto;
181     }
182
183     public JLabel[] getNombres() {
184
185         return nombres;
186
187     }
188
189     public JLabel[] getPuntajes() {
190
191         return puntajes;
192
193     }
```

```
194
195     public String[] getNombre() {
196
197         return nombre;
198
199     }
200
201     public String[] getPuntaje() {
202
203         return puntaje;
204
205     }
206
207     public int[] getPuntaje1() {
208
209         return puntaje1;
210
211     }
212
213     public String getJugador() {
214
215         return jugador;
216
217     }
218
219     public String getUbicacionNombre() {
220
221         return ubicacionNombre;
222
223     }
224
225     public String getUbicacionPuntaje() {
226
227         return ubicacionPuntaje;
228
229     }
230
231 }
232
```

Main.java

```java
package Tetris;

 * NOMBRE DEL PROYECTO: TETRIS

import java.awt.EventQueue;

public class Main {

    public static void main(String[] arg) throws Exception {


        EventQueue.invokeLater(new Runnable() {
            public void run() {
                try {

                    // Se crean los archivos
                    String ubicacionNombre = "src/Archivo/archivoNombre.txt";
                    String ubicacionPuntaje = "src/Archivo/archivoPuntaje.txt";
                    VentanaPrincipal window = new VentanaPrincipal(ubicacionNombre, ubicacionPuntaje);
                    window.setVisible(true);

                    // Creacion del objeto que reproduce la música en el juego.
                    String filepath = "src/Sonido/sonidoTetris.wav";
                    Sonido music = new Sonido();
                    music.play(filepath);
                } catch (Exception e) {
                    e.printStackTrace();
                }
            }
        });
    }

}
```

Pieza.java

```java
package Tetris;

/*
 * NOMBRE DEL PROYECTO: TETRIS
 *
 * AUTORES: Javier Santiago Borbón Borbón y José Andrés Sanabria Arias
 * Códigos: 20182020085 y 2018202095
 *
 *
 * FECHA: 28/03/2020 (Marzo 28 de 2020)
 *
 *
 * Versión 1.0;
 *
 */

enum Pieza {

    figuraBorrador, figuraZ, figuraS, figuraLinea, figuraT, figuraCuadrado, figuraL, figuraLInv

}

```

Sonido.java

```java
package Tetris;

 * NOMBRE DEL PROYECTO: TETRIS

import java.io.File;

public class Sonido {
    public void play(String musicLocation) {
        try {
            File musicPath = new File(musicLocation);
            if (musicPath.exists()) {
                AudioInputStream audioInput = AudioSystem.getAudioInputStream(musicPath);
                Clip clip = AudioSystem.getClip();
                clip.open(audioInput);
                clip.start();
                clip.loop(Clip.LOOP_CONTINUOUSLY);

            } else {
                System.out.println("No se encuentra la música");
            }

        } catch (Exception ex) {
            ex.printStackTrace();
        }
    }
}

```

Tablero.java

```
1  package Tetris;
2
4  * NOMBRE DEL PROYECTO: TETRIS
16
17 import java.awt.Color;
29
30 public class Tablero extends JPanel implements ActionListener {
31
32     // Tamaño del tablero del tetris
33     private final int anchoTablero = 10;
34     private final int alturaTablero = 20;
35
36     // Varianbles necesarias para procesar el juego en el tablero.
37
38     private boolean haCaido = false;
39     private boolean empezo = false;
40     private boolean enPausa = false;
41     private int numLineasRemovidas = 0;
42     private int curX = 0;
43     private int curY = 0;
44     private JLabel estadoPunto;
45     private Forma curPiece;
46     private Pieza[] board;
47
48     // Llamamos a las figuras
49
50     private FiguraBorrador figBorrador = new FiguraBorrador();
51     private FiguraCuadrado figC = new FiguraCuadrado();
52     private FiguraL figL = new FiguraL();
53     private FiguraLinea figLinea = new FiguraLinea();
54     private FiguraLInv figLInv = new FiguraLInv();
55     private FiguraS figS = new FiguraS();
56     private FiguraT figT = new FiguraT();
57     private FiguraZ figZ = new FiguraZ();
58
59
60     // Variables necesarias para hacer la persistencia mediante Archivos.
61     private Archivo archivo = new Archivo();
62     private String ubicacionNombre;
63     private String ubicacionPuntaje;
64     private String nombre[];
65     private String puntaje[];
66     private int puntaje1[];
67     private int auxPuntaje = 0;
68     private String auxNombre = null;
69     private String puntoCadena = null;
70     private int puntoEntero = 0;
71     private String jugador = null;
72     private Timer timer;
73
```

```
74      // Contructor de la clase Tablero, en la cual se encarga del Panel3, central o
75      // tablero. Se requiere de parametro la Interfaz para hacer un traspaso de
76      // Variables.
77      public Tablero(Interfaz interfaz) {
78
79          setFocusable(true);
80          setBackground(Color.BLACK);
81          curPiece = new Forma();
82          timer = new Timer(400, this);
83          timer.start();
84          ubicacionNombre = interfaz.getUbicacionNombre();
85          ubicacionPuntaje = interfaz.getUbicacionPuntaje();
86          jugador = interfaz.getJugador();
87          estadoPunto = interfaz.getEtiquetaPunto();
88          nombre = interfaz.getNombre();
89          puntaje = interfaz.getPuntaje();
90          puntaje1 = interfaz.getPuntaje1();
91
92          board = new Pieza[anchoTablero * alturaTablero];
93          addKeyListener(new TAdapter());
94          clearBoard();
95      }
96
97      // Método que hace que la pieza caiga.
98      public void actionPerformed(ActionEvent e) {
99          if (haCaido) {
100             haCaido = false;
101             newPiece();
102         } else {
103             oneLineDown();
104         }
105     }
106
107     // Establece los límites del juego en el tablero.
108     int squareWidth() {
109         return (int) getSize().getWidth() / anchoTablero;
110     }
111
112     int squareHeight() {
113         return (int) getSize().getHeight() / alturaTablero;
114     }
115
116     Pieza shapeAt(int x, int y) {
117         return board[(y * anchoTablero) + x];
118     }
119
120     // Método principal del tablero, da inicio al timer, creación de piezas.
121     public void start() {
122         if (enPausa)
123             return;
124
125         empezo = true;
126         haCaido = false;
127         numLineasRemovidas = 0;
128         clearBoard();
129
130         newPiece();
131         timer.start();
132     }
133
```

```
133
134     // Método por el cual se logra detener el juego.
135     private void pause() {
136         if (!empezo)
137             return;
138
139         enPausa = !enPausa;
140         if (enPausa) {
141             timer.stop();
142             estadoPunto.setText("Pausado");
143         } else {
144             timer.start();
145             estadoPunto.setText(String.valueOf(numLineasRemovidas));
146         }
147         repaint();
148     }
149
150     // Método por el cual se pintan las pizas gracias a la clase Forma.
151     public void paint(Graphics g) {
152         super.paint(g);
153
154         Dimension size = getSize();
155         int boardTop = (int) size.getHeight() - alturaTablero * squareHeight(); // Pintar Tablero
156
157         for (int i = 0; i < alturaTablero; ++i) {
158             for (int j = 0; j < anchoTablero; ++j) {
159                 Pieza shape = shapeAt(j, alturaTablero - i - 1);
160                 if (shape != Pieza.figuraBorrador)
161                     drawSquare(g, 0 + j * squareWidth(), boardTop + i * squareHeight(), shape);
162             }
163         }
164
165         if (curPiece.getShape() != Pieza.figuraBorrador) {
166             for (int i = 0; i < 4; ++i) {
167                 int x = curX + curPiece.x(i);
168                 int y = curY - curPiece.y(i);
169                 drawSquare(g, 0 + x * squareWidth(), boardTop + (alturaTablero - y - 1) * squareHeight(),
170                         curPiece.getShape());
171             }
172         }
173     }
174
175     // Método para hacer caer la ficha (con BarraEspaciadora en el juego).
176     private void dropDown() {
177         int newY = curY;
178         while (newY > 0) {
179             if (!tryMove(curPiece, curX, newY - 1))
180                 break;
181             --newY;
182         }
183         pieceDropped();
184     }
185
186     // Método por el cual la ficha baja si no hay un obstaculo.
187     private void oneLineDown() {
188         if (!tryMove(curPiece, curX, curY - 1))
189             pieceDropped();
190     }
191
```

```
192     // Método para limpiar el rastro de las fichas al caer.
193     private void clearBoard() {
194         for (int i = 0; i < alturaTablero * anchoTablero; ++i)
195             board[i] = Pieza.figuraBorrador;
196     }
197 
198     // Método por el cual
199     private void pieceDropped() {
200         for (int i = 0; i < 4; ++i) {
201             int x = curX + curPiece.x(i);
202             int y = curY - curPiece.y(i);
203             board[(y * anchoTablero) + x] = curPiece.getShape();
204         }
205 
206         removerLineasLlenas();
207 
208         if (!haCaido)
209             newPiece();
210     }
211 
212     // Método fundamental para crear piezas aleatorias, además de ser el método que
213     // crea la persistencia.
214     private void newPiece() {
215         curPiece.setRandomShape();
216         curX = anchoTablero / 2 + 1;
217         curY = alturaTablero - 1 + curPiece.minY();
218 
219         if (!tryMove(curPiece, curX, curY)) {
220             curPiece.setShape(Pieza.figuraBorrador);
221             timer.stop();
222             empezo = false;
223             puntoCadena = estadoPunto.getText();
224             puntoEntero = Integer.parseInt(puntoCadena);
225             JOptionPane.showMessageDialog(null, "JUEGO TERMINADO!!\n\nSu puntaje es: " + puntoEntero, "GAME OVER",
226                     JOptionPane.INFORMATION_MESSAGE, null);
227 
228             actualizarTabla();
229 
230         }
231     }
232 
233     // Método que ayuda a no sobre poner las fichas al rotar.
234     private boolean tryMove(Forma newPiece, int newX, int newY) {
235         for (int i = 0; i < 4; ++i) {
236             int x = newX + newPiece.x(i);
237             int y = newY - newPiece.y(i);
238             if (x < 0 || x >= anchoTablero || y < 0 || y >= alturaTablero)
239                 return false;
240             if (shapeAt(x, y) != Pieza.figuraBorrador)
241                 return false;
242         }
243 
244         curPiece = newPiece;
245         curX = newX;
246         curY = newY;
247         repaint();
248         return true;
249     }
250 
251     // Método por el cual identifica cuando hay una linea, aumenta el puntaje del
```

```
    // Método por el cual identifica cuando hay una linea, aumenta el puntaje del
    // jugador.
    private void removerLineasLlenas() {
        int numFullLines = 0;

        for (int i = alturaTablero - 1; i >= 0; --i) {
            boolean lineIsFull = true;

            for (int j = 0; j < anchoTablero; ++j) {
                if (shapeAt(j, i) == Pieza.figuraBorrador) {
                    lineIsFull = false;
                    break;
                }
            }

            if (lineIsFull) {
                ++numFullLines;
                for (int k = i; k < alturaTablero - 1; ++k) {
                    for (int j = 0; j < anchoTablero; ++j)
                        board[(k * anchoTablero) + j] = shapeAt(j, k + 1);
                }
            }
        }

        if (numFullLines > 0) {
            numLineasRemovidas += numFullLines;
            estadoPunto.setText(String.valueOf(numLineasRemovidas));
            haCaido = true;
            curPiece.setShape(Pieza.figuraBorrador);
            repaint();
        }
    }

    private void drawSquare(Graphics g, int x, int y, Pieza shape) {

        Color colors[] = { figBorrador.getColor(), figZ.getColor(), figS.getColor(), figLinea.getColor(), figT.getColor(), figC.getColor(), figL.getColor(), figLInv.getColor() };

        Color color = colors[shape.ordinal()]; //Se crea un arreglo de 7 figuras

        g.setColor(color);
        g.fillRect(x + 1, y + 1, squareWidth() - 2, squareHeight() - 2);

        g.setColor(color.brighter());
        g.drawLine(x, y + squareHeight() - 1, x, y);
        g.drawLine(x, y, x + squareWidth() - 1, y);

        g.setColor(color.darker());
        g.drawLine(x + 1, y + squareHeight() - 1, x + squareWidth() - 1, y + squareHeight() - 1);
        g.drawLine(x + squareWidth() - 1, y + squareHeight() - 1, x + squareWidth() - 1, y + 1);
    }
```

```
301
302     // Clase que capta los eventos por teclado posteriormente mueve la pieza o
303     // detiene el juego.
304     class TAdapter extends KeyAdapter {
305         public void keyPressed(KeyEvent e) {
306             if (!empezo || curPiece.getShape() == Pieza.figuraBorrador) {
307                 return;
308             }
309
310             int keycode = e.getKeyCode();
311
312             if (keycode == 'p' || keycode == 'P') {
313                 pause();
314                 return;
315             }
316
317             if (enPausa)
318                 return;
319
320             switch (keycode) {
321             case KeyEvent.VK_LEFT:
322                 tryMove(curPiece, curX - 1, curY);
323                 break;
324             case KeyEvent.VK_RIGHT:
325                 tryMove(curPiece, curX + 1, curY);
326                 break;
327             case KeyEvent.VK_DOWN:
328                 oneLineDown();
329                 break;
330             case KeyEvent.VK_UP:
331                 tryMove(curPiece.rotateLeft(), curX, curY);
332                 break;
333             case KeyEvent.VK_SPACE:
334                 dropDown();
335                 break;
336             }
337
338         }
339     }
340
341     private void actualizarTabla() {
342
343         nombre[10] = jugador;
344         puntaje1[10] = puntoEntero;
345
346         for (int i = 0; i < 10; i++) {
347
348             for (int j = 0; j < 10; j++) {
349
350                 if (puntaje1[j] < puntaje1[j + 1]) {
351
352                     auxPuntaje = puntaje1[j];
353                     puntaje1[j] = puntaje1[j + 1];
354                     puntaje1[j + 1] = auxPuntaje;
355
356                     auxNombre = nombre[j];
357                     nombre[j] = nombre[j + 1];
358                     nombre[j + 1] = auxNombre;
359
360                 }
```

```
                }

            }

        }

        for (int i = 0; i < 10; i++) {

            puntaje[i] = Integer.toString(puntaje1[i]);

        }

        archivo.sobreEscribirArchivo(nombre, ubicacionNombre);
        archivo.sobreEscribirArchivo(puntaje1, ubicacionPuntaje);

    }

}
```

VentanaPrincipal.java

```java
package Tetris;

/**
 * @tittle      Ventana Principal
 */
import java.awt.Color;

public class VentanaPrincipal extends JFrame implements ActionListener {
// Objetos para la persistencia mediante Archivos.

    private static final long serialVersionUID = 1L;
    private JPanel contentPane;
    private JPanel panelPuntajes;
    private JPanel panelInstrucciones;
    private String ubicacionNombre = null;
    private String ubicacionPuntaje = null;
    private JLabel puntajes[] = new JLabel[10];
    private JLabel nombres[] = new JLabel[10];
    private String nombre[];
    private String puntaje[];
    // BOTONES
    private JButton btnSalir;
    private JButton btnPuntuacion;
    private JButton btnNewGame;
    private JButton btnInstrucciones;

    public VentanaPrincipal(String ubicacionNombre, String ubicacionPuntaje) {
        this.ubicacionNombre = ubicacionNombre;
        this.ubicacionPuntaje = ubicacionPuntaje;
        setBounds(100, 100, 600, 300);
        setUndecorated(true);
        setLocationRelativeTo(null);
        contentPane = new JPanel();
        setContentPane(contentPane);
        contentPane.setLayout(null);

        // Asignación de las ubicaciones de los archivos a las variables de la clase
        // Interfaz.
        this.ubicacionNombre = ubicacionNombre;
        this.ubicacionPuntaje = ubicacionPuntaje;

        // Creación de los archivos.
        Archivo archivoNombre = new Archivo(ubicacionNombre);
        Archivo archivoPuntaje = new Archivo(ubicacionPuntaje);
        nombre = archivoNombre.leerArchivoString(ubicacionNombre);
        puntaje = archivoPuntaje.leerArchivoIntAString(ubicacionPuntaje);

       // Creacion del Panel en donde se muestran las instrucciones
        panelInstrucciones = new JPanel();
        panelInstrucciones.setBackground(Color.BLACK);
        panelInstrucciones.setBounds(170, 0, 259, 262);
        contentPane.add(panelInstrucciones);
        panelInstrucciones.setLayout(null);
        JTextPane textoInstruccion = new JTextPane();
        textoInstruccion.setEditable(false);
        textoInstruccion.setForeground(new Color(255, 255, 255));
        textoInstruccion.setBackground(Color.BLACK);
        textoInstruccion.setText(
                " P : Pausar el juego.\r\n \u2191 : Rotar la figura.\r\n \u2192 : Moverse a la derecha.\r\n \u2190 : Moverse a la izquierda.\r\n \u2193 : Moverse hacia abajo.\r\n Barra Espaciadora: Bajar la figura.");
        textoInstruccion.setBounds(42, 161, 217, 101);
        panelInstrucciones.add(textoInstruccion);

        JLabel lblIcono = new JLabel("");
        lblIcono.setIcon(new ImageIcon(VentanaPrincipal.class.getResource("/Imagen/iconoInstrucciones.jpg")));
        lblIcono.setBounds(0, 0, 259, 160);
        panelInstrucciones.add(lblIcono);
        panelInstrucciones.setVisible(false);
        // Creacion del Panel en donde se muestran los puntajes
        panelPuntajes = new JPanel();
        panelPuntajes.setBounds(112, 0, 375, 262);
        contentPane.add(panelPuntajes);
        JLabel lblTituloPuntajes = new JLabel("NUESTROS CAMPEONES");
        lblTituloPuntajes.setForeground(Color.BLUE);
        lblTituloPuntajes.setFont(new Font("Tw Cen MT Condensed Extra Bold", Font.PLAIN, 27));
        lblTituloPuntajes.setBounds(72, 5, 240, 30);
        colocarPuntajes();
        panelPuntajes.add(lblTituloPuntajes);
        JLabel lblName = new JLabel("NOMBRE");
        lblName.setFont(new Font("Times New Roman", Font.PLAIN, 16));
        lblName.setBounds(30, 40, 244, 14);
        panelPuntajes.add(lblName);

        JLabel lblPuntos = new JLabel("PUNTOS");
        lblPuntos.setFont(new Font("Times New Roman", Font.PLAIN, 16));
        lblPuntos.setBounds(266, 40, 99, 14);
        panelPuntajes.add(lblPuntos);

        JLabel lblImagenPFondo = new JLabel();
        lblImagenPFondo.setBounds(0, 0, 375, 262);
        lblImagenPFondo.setIcon(new ImageIcon(VentanaPrincipal.class.getResource("/Imagen/fondo.jpg")));
        panelPuntajes.add(lblImagenPFondo);
        panelPuntajes.setVisible(false);
        btnInstrucciones = new JButton("Instrucciones");
        btnInstrucciones.setBounds(170, 266, 118, 23);
        btnInstrucciones.addActionListener(this);
        contentPane.add(btnInstrucciones);

        btnNewGame = new JButton("Nuevo Juego");
        btnNewGame.setBounds(26, 266, 124, 23);
        btnNewGame.addActionListener(this);
        contentPane.add(btnNewGame);

        btnPuntuacion = new JButton("Puntuaci\u00F3n");
        btnPuntuacion.setBounds(312, 266, 100, 23);
        btnPuntuacion.addActionListener(this);
        contentPane.add(btnPuntuacion);
```

```
135         btnPuntuacion = new JButton("Puntuaci\u00F3n");
136         btnPuntuacion.setBounds(312, 266, 100, 23);
137         btnPuntuacion.addActionListener(this);
138         contentPane.add(btnPuntuacion);
139 
140         btnSalir = new JButton("Salir");
141         btnSalir.addActionListener(this);
142         btnSalir.setBounds(445, 266, 106, 23);
143         contentPane.add(btnSalir);
144 
145         JLabel lblImgVentana = new JLabel("");
146         lblImgVentana.setBounds(0, 0, 600, 300);
147         lblImgVentana.setIcon(new ImageIcon(VentanaPrincipal.class.getResource("/Imagen/Main Logo.jpg")));
148         contentPane.add(lblImgVentana);
149 
150     }
151 
152     private void colocarPuntajes() {
153         panelPuntajes.setLayout(null);
154 
155         for (int i = 0; i < puntajes.length; i++) {
156             nombres[i] = new JLabel(nombre[i]);
157             nombres[i].setBounds(10, 60 + 20 * i, 264, 14);
158             panelPuntajes.add(nombres[i]);
159             if (Integer.parseInt(puntaje[i]) < 10) {
160                 puntajes[i] = new JLabel("  " + puntaje[i]);
161             } else if (Integer.parseInt(puntaje[i]) > 100) {
162                 puntajes[i] = new JLabel(puntaje[i]);
163             } else {
164                 puntajes[i] = new JLabel(" " + puntaje[i]);
165             }
166             puntajes[i].setBounds(285, 60 + 20 * i, 80, 14);
167             panelPuntajes.add(puntajes[i]);
168         }
169 
170     }
171 
172     // Aquí es donde se le da funcionamiento a los botones
173     @Override
174     public void actionPerformed(ActionEvent e) {
175         if (e.getSource().equals(btnNewGame)) {
176             panelPuntajes.setVisible(false);
177             panelInstrucciones.setVisible(false);
178             String name="";
179             name=JOptionPane.showInputDialog("Ingrese el nombre del jugador:");
180             if (name==null||name.isEmpty()||name=="") { //Si el jugador no ingresa nombre alguno
181                 name = "Player 1";
182             }
183             Interfaz juegoTetris = new Interfaz (name, ubicacionNombre, ubicacionPuntaje);
184             dispose();
185             juegoTetris.marcoPrograma.setVisible(true);
186 
187         }
188 
189         if (e.getSource().equals(btnPuntuacion)) {
190             panelInstrucciones.setVisible(false);
191             if (panelPuntajes.isVisible()) {
192                 panelPuntajes.setVisible(false);
193             } else {
```

```
154
155         for (int i = 0; i < puntajes.length; i++) {
156             nombres[i] = new JLabel(nombre[i]);
157             nombres[i].setBounds(10, 60 + 20 * i, 264, 14);
158             panelPuntajes.add(nombres[i]);
159             if (Integer.parseInt(puntaje[i]) < 10) {
160                 puntajes[i] = new JLabel("  " + puntaje[i]);
161             } else if (Integer.parseInt(puntaje[i]) > 100) {
162                 puntajes[i] = new JLabel(puntaje[i]);
163             } else {
164                 puntajes[i] = new JLabel(" " + puntaje[i]);
165             }
166             puntajes[i].setBounds(285, 60 + 20 * i, 80, 14);
167             panelPuntajes.add(puntajes[i]);
168         }
169
170     }
171
172     // Aquí es donde se le da funcionamiento a los botones
173     @Override
174     public void actionPerformed(ActionEvent e) {
175         if (e.getSource().equals(btnNewGame)) {
176             panelPuntajes.setVisible(false);
177             panelInstrucciones.setVisible(false);
178             String name="";
179             name=JOptionPane.showInputDialog("Ingrese el nombre del jugador:");
180             if (name==null||name.isEmpty()||name=="") { //Si el jugador no ingresa nombre alguno
181                 name = "Player 1";
182             }
183             Interfaz juegoTetris = new Interfaz (name, ubicacionNombre, ubicacionPuntaje);
184             dispose();
185             juegoTetris.marcoPrograma.setVisible(true);
186
187         }
188
189         if (e.getSource().equals(btnPuntuacion)) {
190             panelInstrucciones.setVisible(false);
191             if (panelPuntajes.isVisible()) {
192                 panelPuntajes.setVisible(false);
193             } else {
194                 panelPuntajes.setVisible(true);
195             }
196
197         }
198         if (e.getSource().equals(btnSalir)) {
199             System.exit(0);
200         }
201         if (e.getSource().equals(btnInstrucciones)) {
202             panelPuntajes.setVisible(false);
203
204             if (panelInstrucciones.isVisible()) {
205                 panelInstrucciones.setVisible(false);
206             } else {
207                 panelInstrucciones.setVisible(true);
208             }
209
210         }
211     }
212 }
213
```

Index.html

En este documento se puede ver cada una de las clases, este índice además de mostrar los diferentes archivos, también le muestra a la persona los métodos, las variables y otros elementos empleados en este proyecto. Si necesita ayuda con este sumario, tiene la opción de ver la ayuda, en la pestaña que dice “HELP”.