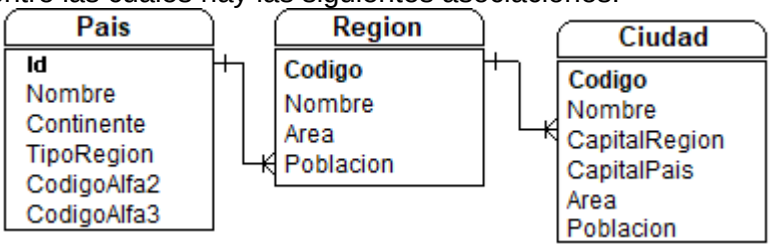
Taller 3 *Fecha: Octubre de 2025*

Competencia a evaluar: Aplicar los conceptos básicos de la lógica de programación y de la orientación a objetos en el desarrollo de una aplicación con interfaz gráfica de usuario que accede a archivos en formato JSON y archivos multimedia (sonido y video).

NOTAS:
- Este taller (punto 2) se debe hacer con carácter evaluativo. Representa en total una calificación de 15%.
- Se entregan ejercicios resueltos como ejemplo para el desarrollo de los demás.

Elaborar el diagrama de clases básico (sin las clases correspondientes a la interface de usuario según el lenguaje de implementación) y la respectiva aplicación en un leguaje orientado a objetos para los siguientes enunciados:

1. La información de la división política mundial puede almacenarse en 3 estructuras objetuales que recopilan los datos de los países, los datos de las regiones y los datos de las ciudades, entre las cuales hay las siguientes asociaciones:

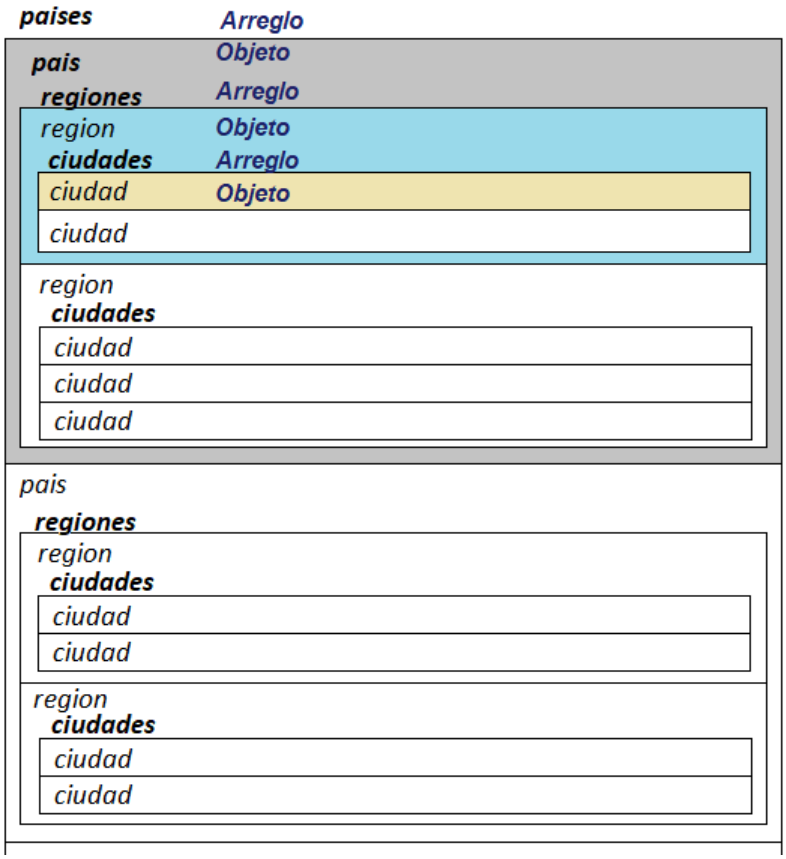


La asociación entre las estructuras es una relación **Uno a muchos**, lo cual significa que un objeto *País* contiene muchos objetos *Regiones* y a su vez, un objeto *Región* contiene muchos objetos *Ciudades*. Para esto se utiliza un archivo JSON como el siguiente

DivisionPolitica.json


```
[
{ "id":4, "nombre":"Afganistán", "continente": "ASIA", "tipoRegion": "ESTADO", "codigoAlfa2": "AF", "codigoAlfa3": "AFG"},
{ "id":8, "nombre":"Albania", "continente": "EUROPA", "tipoRegion": "ESTADO", "codigoAlfa2": "AL", "codigoAlfa3": "ALB"},
{ "id":10, "nombre":"Antártida", "continente": "ANTARTIDA", "tipoRegion": "ESTADO", "codigoAlfa2": "AQ", "codigoAlfa3": "ATA"},
{ "id":12, "nombre":"Argelia", "continente": "AFRICA", "tipoRegion": "ESTADO", "codigoAlfa2": "DZ", "codigoAlfa3": "DZA"},
{ "id":16, "nombre":"Samoa Americana", "continente": "OCEANIA", "tipoRegion": "ESTADO", "codigoAlfa2": "AS", "codigoAlfa3": "ASM"},
{ "id":20, "nombre":"Andorra", "continente": "EUROPA", "tipoRegion": "ESTADO", "codigoAlfa2": "AD", "codigoAlfa3": "AND"},
{ "id":24, "nombre":"Angola", "continente": "AFRICA", "tipoRegion": "ESTADO", "codigoAlfa2": "AO", "codigoAlfa3": "AGO"},
{ "id":28, "nombre":"Antigua y Barbuda", "continente": "AMERICA", "tipoRegion": "ESTADO", "codigoAlfa2": "AG", "codigoAlfa3": "ATG"
{ "id":31, "nombre":"Azerbaiyán", "continente": "ASIA", "tipoRegion": "ESTADO", "codigoAlfa2": "AZ", "codigoAlfa3": "AZE"},
{ "id":32, "nombre":"Argentina", "continente": "AMERICA", "tipoRegion": "ESTADO", "codigoAlfa2": "AR", "codigoAlfa3": "ARG",
"regiones":[
{ "nombre":"Buenos Aires", "area": 307804, "poblacion": 15052177,
"ciudades": [ { "nombre": "La Plata", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Cordoba", "area": 168766, "poblacion": 3340041,
"ciudades": [ { "nombre": "Córdoba", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Santa Fe", "area": 133007, "poblacion": 3242551,
"ciudades": [ { "nombre": "Santa Fe de la Vera Cruz", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Mendoza", "area": 150839, "poblacion": 1729660,
"ciudades": [ { "nombre": "Mendoza", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Tucuman", "area": 22524, "poblacion": 1475384,
"ciudades": [ { "nombre": "San Miguel de TucumAn", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Entre Rios", "area": 78781, "poblacion": 1255787,
"ciudades": [ { "nombre": "Parana", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Salta", "area": 154775, "poblacion": 1224022,
"ciudades": [ { "nombre": "Salta", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Misiones", "area": 29801, "poblacion": 1077987,
"ciudades": [ { "nombre": "Posadas", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Chaco", "area": 99633, "poblacion": 1052185,
"ciudades": [ { "nombre": "Resistencia", "capitalRegion": true, "capitalPais": false}]},
{ "nombre":"Corrientes", "area": 88199, "poblacion": 1013443,
```


Este archivo distribuye la información en objetos *JSON* de acuerdo a la siguiente ilustración:



Donde se puede observar que las colecciones de objetos se agrupan en arreglos.

¿Qué es JSON?

JSON (**JavaScript Object Notation**) es un formato de intercambio de datos ligero y fácil de leer, basado en la sintaxis de objetos de JavaScript. Se utiliza ampliamente para el almacenamiento y transmisión de información entre aplicaciones, especialmente en el desarrollo web y APIs.

JSON usa dos estructuras principales:

- Objetos (colección de pares clave-valor)

```
{
    "nombre": "Juan",
    "edad": 20,
    "esEstudiante": true
}
```

- Arreglos (listas de valores u objetos)

```
[
    "Rojo",
    "Verde",
    "Azul"
]
```

Se pueden definir estructuras que combinen ambos:

```
{
    "pais": "Colombia",
    "capital": "Bogotá",
    "poblacion": 50800000,
    "idiomas": ["Español", "Inglés"],
```

```
    "moneda": {
        "nombre": "Peso Colombiano",
        "codigo": "COP"
    }
}
```

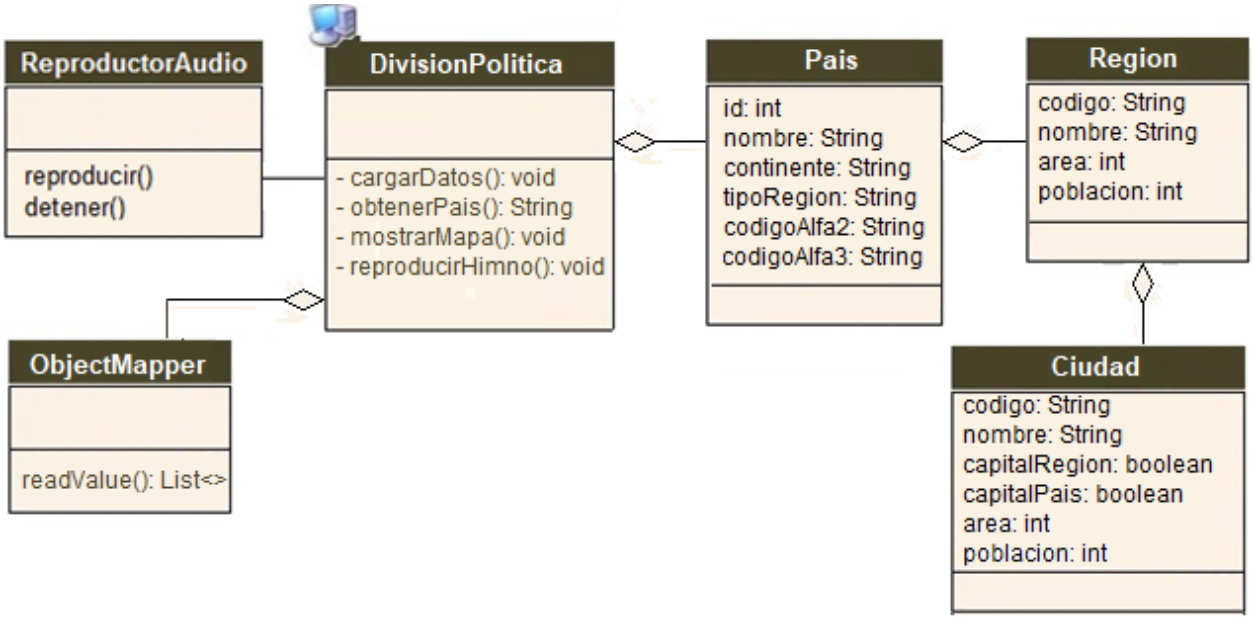
- *pais*, *capital* y *poblacion* son pares clave-valor
- *idiomas* es un arreglo
- *moneda* es un objeto anidado dentro del JSON

Se requiere una aplicación que acceda el archivo JSON con la información antes planteada y permitirle al usuario:

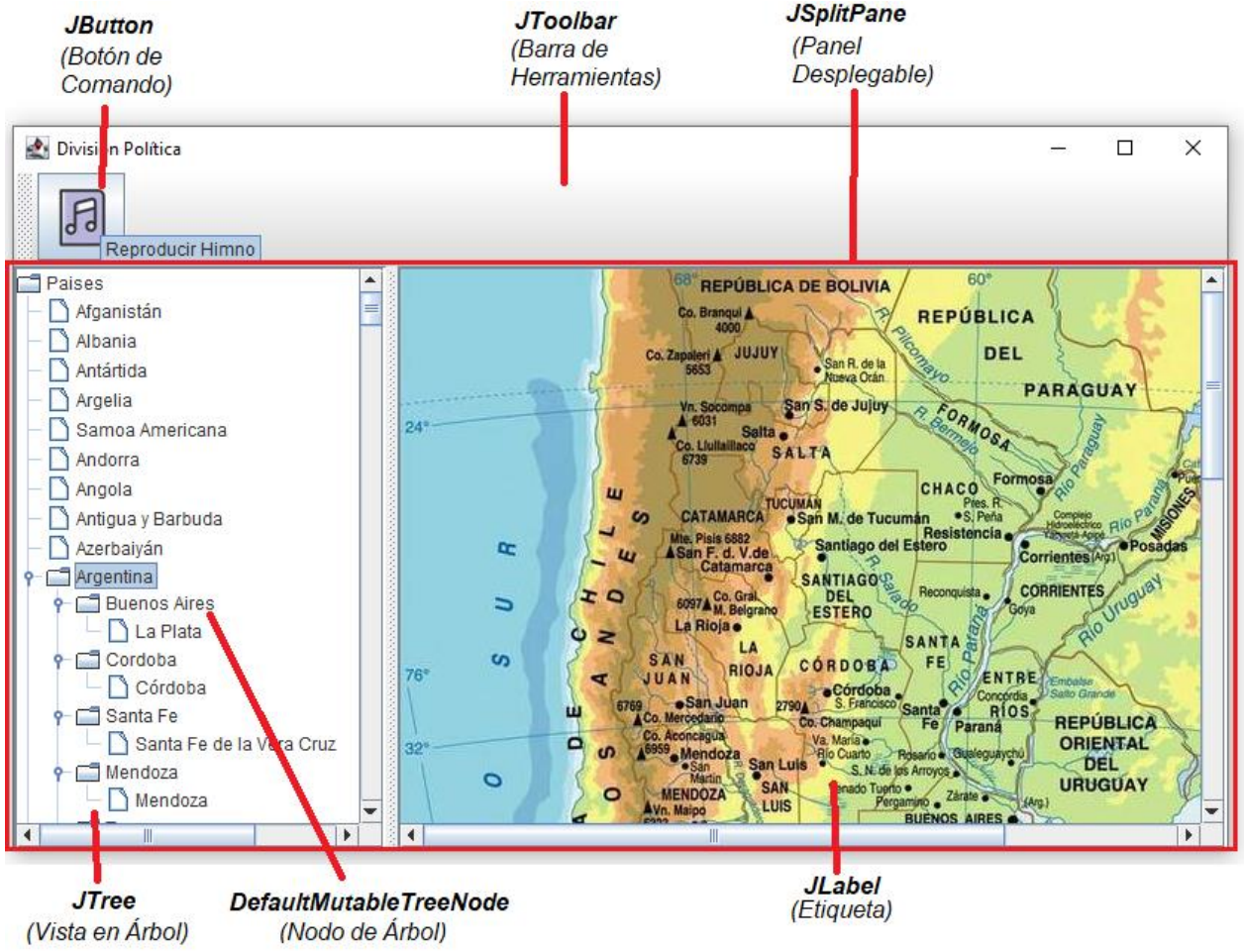
- Listar en una vista en árbol los países, regiones y ciudades, siguiendo la jerarquía planteada
- Seleccionar un país y mostrar su mapa
- Habiendo seleccionado un país reproducir su himno nacional

R/

- El modelado bajo el paradigma Orientado a Objetos se ilustra en el siguiente diagrama de clases:



- **Programa**
  Para la implementación del aplicativo se debe comenzar con el diseño de un formulario como el siguiente y que corresponde a la clase *DivisionPolitica* del anterior diagrama de clases:

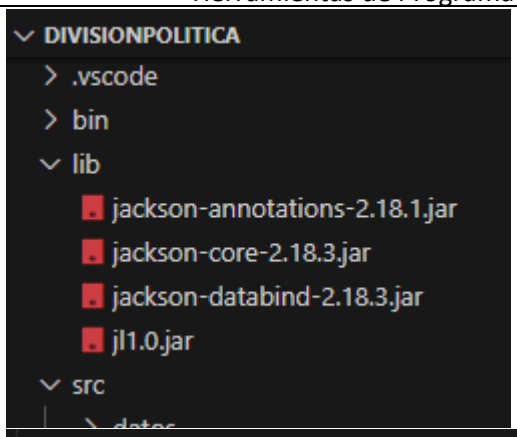


La siguiente tabla relaciona los objetos a añadir con las propiedades cuyos valores deben ser cambiados:

Tipo Control	Nombre	Otras Propiedades
JToolBar	*tbDivisionPolitica*	
JButton	*btnHimno*	Text = *“”* Icon = ”Himno.png” ToolTipText = “Reproducir Himno”
JSplitPane	*spDivisionPolitica*	DividerLocation = *250*
JLabel	*lblMapa*	
JTree	*arbol*	

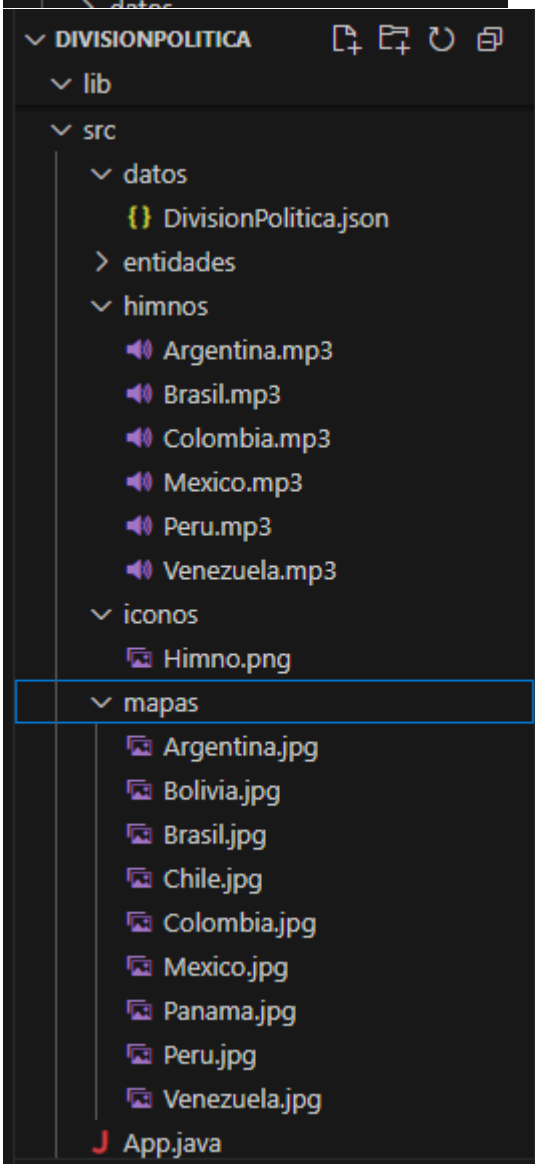
Se debe también agregar al proyecto las siguientes 4 librerías (archivos compilados de Java):

- Una que permite reproducir archivos de audio *MP3*. En este caso se utilizará *JLayer* de *JavaZoom* (http://www.javazoom.net/javalayer/sources.html)
- 3 que en conjunto permiten manipular objetos JSON: *jackson-annotations*, *jackson-core* y *jackson-databind*

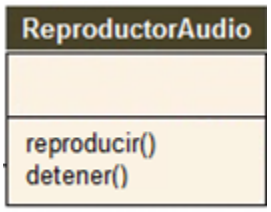


Se deben crear también en el proyecto las carpetas *datos*, *himnos*, *iconos* y *mapas* para incluir:

- En la primera, el archivo plano en formato *JSON* con los datos
- En la segunda, los audios en formato MP3 de los himnos nacionales
- En la tercera, las imágenes para los botones de comando
- En la cuarta, las imágenes de los mapas de los países



De acuerdo al modelo de clases planteado, se debe editar la clase *ReproductorAudio* la cual ofrece funcionalidad relacionada con la reproducción de audio en formato MP3. Se compone de:



- El método *reproducir()* que permite abrir un archivo de audio y reproducirlo por los parlantes
- El método *detener()* que permite terminar la reproducción de un archivo de audio iniciado

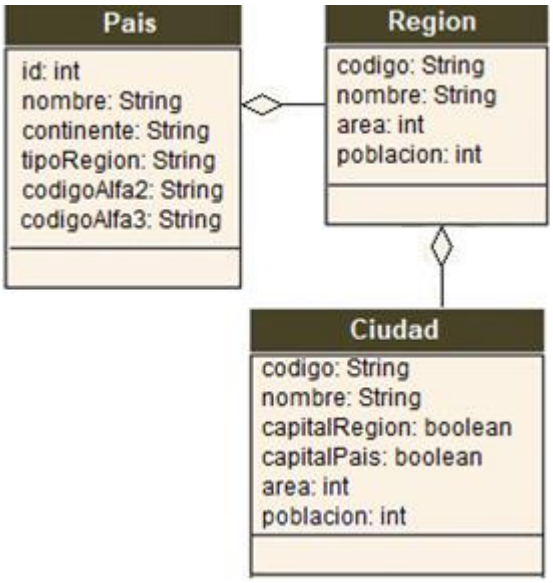
El código de la clase sería el siguiente:

```
import javazoom.jl.player.*;
import java.io.*;
import javax.swing.*;

public class ReproductorAudio {

    private static Player reproductor;

    //Detener la reproducción
    public static void detener() {
        if (reproductor != null) {
            reproductor.close();
            reproductor=null;
        }
    }

    //Reproducir el archivo MP3
    public static void reproducir(String nombreArchivo) {
        detener();
        try {
            FileInputStream fis = new FileInputStream(nombreArchivo);
            BufferedInputStream bis = new BufferedInputStream(fis);
            reproductor = new Player(bis);
        } catch (Exception e) {
            JOptionPane.showMessageDialog(new JFrame(), e);
        }

        // Ejecutar en un nuevo hilo la reproducción de la canción
        new Thread() {

            public void run() {
                try {
                    reproductor.play();
                } catch (Exception e) {
                    JOptionPane.showMessageDialog(new JFrame(), e);
                }
            }
        }.start();
    }//reproducir
}
```

En la implementación de esta clase es importante resaltar lo siguiente:

- El objeto *reproductor* (de la clase ***Player*** que está incluida en la librería ***jl1.0***) se encarga de la operatividad del audio que se cargue a partir del archivo MP3. El archivo es leído mediante un objeto de la clase ***FileInputStream*** el cual obtiene la secuencia de bytes en un objeto de la clase ***BufferedInputStream***
- La ejecución de la reproducción del audio se hace en un nuevo hilo, para que la ejecución del aplicativo siga su curso

Las clases *Pais*, *Region* y *Ciudad* corresponden a **Entidades** o sea estructuras que modelan un concepto, objeto o elemento de la realidad, generalmente incluyendo solo sus atributos. En el contexto de bases de datos y frameworks como *JPA* (*Java Persistence API*), una entidad también suele estar asociada a una tabla en una base de datos.



El siguiente sería el código de estas entidades:

```
package entidades;

public class Ciudad {
    private String codigo;
    private String nombre;
    private boolean capitalRegion;
    private boolean capitalPais;

    // Getters y Setters
    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public boolean isCapitalRegion() {
        return capitalRegion;
    }

    public void setCapitalRegion(boolean capitalRegion) {
        this.capitalRegion = capitalRegion;
    }

    public boolean isCapitalPais() {
        return capitalPais;
    }
```

```
    public void setCapitalPais(boolean capitalPais) {
        this.capitalPais = capitalPais;
    }
}
```

```
package entidades;

import java.util.List;

public class Region {
    private String codigo;
    private String nombre;
    private int area;
    private int poblacion;
    private List<Ciudad> ciudades;

    // Getters y Setters
    public String getCodigo() {
        return codigo;
    }

    public void setCodigo(String codigo) {
        this.codigo = codigo;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public int getArea() {
        return area;
    }

    public void setArea(int area) {
        this.area = area;
    }

    public int getPoblacion() {
        return poblacion;
    }

    public void setPoblacion(int poblacion) {
        this.poblacion = poblacion;
    }

    public List<Ciudad> getCiudades() {
        return ciudades;
    }

    public void setCiudades(List<Ciudad> ciudades) {
        this.ciudades = ciudades;
    }

}
```

```
package entidades;

import java.util.List;

public class Pais {
    private int id;
    private String nombre;
    private String continente;
    private String tipoRegion;
    private String codigoAlfa2;
    private String codigoAlfa3;
    private List<Region> regiones;

    // Getters y Setters
    public int getId() {
        return id;
    }

    public void setId(int id) {
        this.id = id;
    }

    public String getNombre() {
        return nombre;
    }

    public void setNombre(String nombre) {
        this.nombre = nombre;
    }

    public String getContinente() {
        return continente;
    }

    public void setContinente(String continente) {
        this.continente = continente;
    }

    public String getTipoRegion() {
        return tipoRegion;
    }

    public void setTipoRegion(String tipoRegion) {
        this.tipoRegion = tipoRegion;
    }

    public String getCodigoAlfa2() {
        return codigoAlfa2;
    }

    public void setCodigoAlfa2(String codigoAlfa2) {
        this.codigoAlfa2 = codigoAlfa2;
    }

    public String getCodigoAlfa3() {
        return codigoAlfa3;
    }
```

```
    public void setCodigoAlfa3(String codigoAlfa3) {
        this.codigoAlfa3 = codigoAlfa3;
    }

    public List<Region> getRegiones() {
        return regiones;
    }

    public void setRegiones(List<Region> regiones) {
        this.regiones = regiones;
    }
}
```

Se continuará ahora con la codificación de la clase *DivisionPolitica* que además de ser una interfaz gráfica, incluye la funcionalidad que permite interactuar con la información:

- El método *cargarDatos()* permitirá leer el archivo con los datos en *JSON* y los cargará en una estructura a modo de colección basada en las clases correspondientes a las entidades.
- El método *obtenerPais()* permite a partir de cualquier nodo de la vista en árbol (de ciudad, región o país), obtener el nombre del país que le corresponde.
- El método *mostrarPapa()* permite cargar en memoria la imagen del mapa de un país a partir de un archivo cuyo nombre debe ser el del país y luego lo muestra en un *JLabel*.
- El método *reproducirHimno()* permite cargar en memoria el archivo de audio con el himno nacional de un país seleccionado y luego reproducirlo

El código de la clase sería el siguiente:

```
import com.fasterxml.jackson.databind.ObjectMapper;

import entidades.Ciudad;
import entidades.Pais;
import entidades.Region;

public class FrmDivisionPolitica extends JFrame {

    List<Pais> paises;


    private void cargarDatos() {
        ObjectMapper objectMapper = new ObjectMapper();
        try {
            String nombreArchivo = System.getProperty("user.dir")
                    + "/src/datos/DivisionPolitica.json";
            paises = objectMapper.readValue(new File(nombreArchivo),

objectMapper.getTypeFactory().constructCollectionType(List.class,
Pais.class));

            if (paises != null) {
```

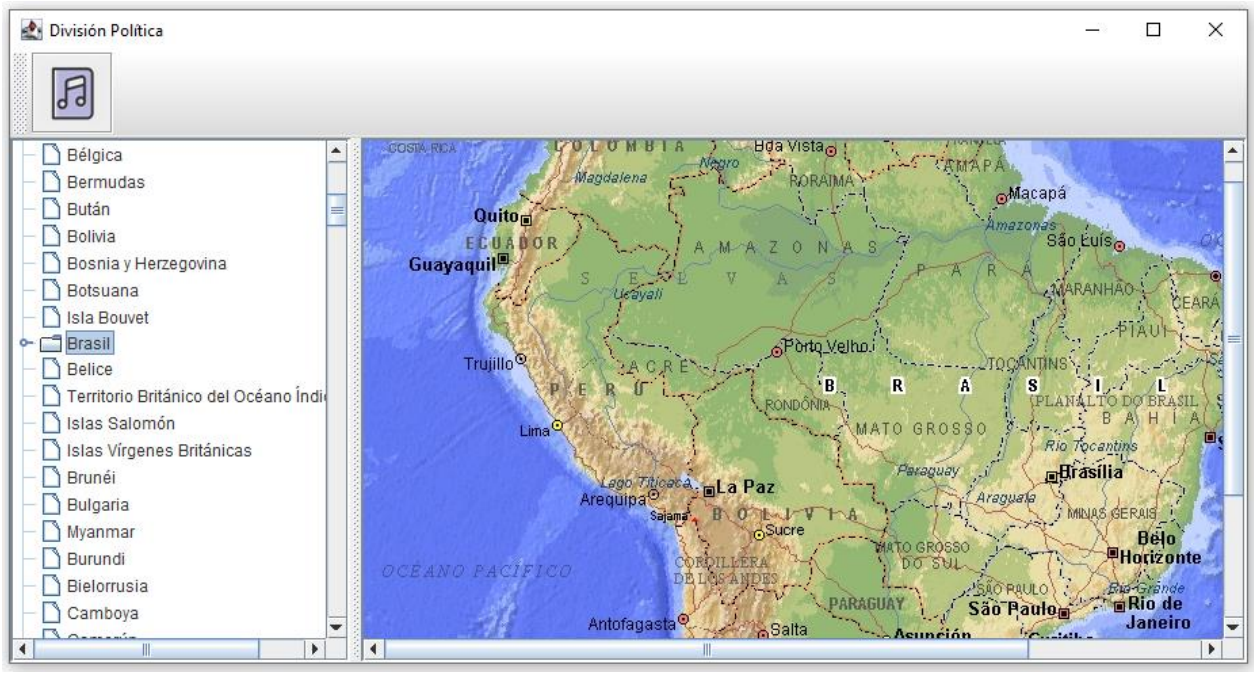
```
                for (Pais pais : paises) {
                    DefaultMutableTreeNode       nodoPais       =       new
DefaultMutableTreeNode(pais.getNombre());
                    if (pais.getRegiones() != null) {
                        for (Region region : pais.getRegiones()) {
                            DefaultMutableTreeNode    nodoRegion    =    new
DefaultMutableTreeNode(region.getNombre());
                            if (region.getCiudades() != null) {
                                for (Ciudad ciudad : region.getCiudades())
{
                                    nodoRegion.add(new
DefaultMutableTreeNode(ciudad.getNombre()));
                                }
                            }
                            nodoPais.add(nodoRegion);
                        }
                    }
                    nodoRaiz.add(nodoPais);
                }
            }
        } catch (IOException e) {
            JOptionPane.showMessageDialog(null, "No se pudieron cargar los
datos" + e);
        }
    }

    private String obtenerPais(DefaultMutableTreeNode nodo) {
        while (nodo != null) {
            if (nodo.getParent() == nodoRaiz) { // Si el padre inmediato es
la raíz "Paises"
                return nodo.toString().replace("á", "a")
                        .replace("é", "e")
                        .replace("í", "i")
                        .replace("ó", "o")
                        .replace("ú", "u"); // Es un país
            }
            nodo = (DefaultMutableTreeNode) nodo.getParent();
        }
        return "";
    }

    private void mostrarMapa(DefaultMutableTreeNode nodo) {
        if (nodo != null) {
            String nombrePais = obtenerPais(nodo);
            if (!nombrePais.equals("")) {
                String rutaMapa = "src/mapas/" + nombrePais + ".jpg"; //
Carpeta donde estarán los mapas
                File archivoMapa = new File(rutaMapa);

                if (archivoMapa.exists()) {
                    lblMapa.setIcon(new ImageIcon(rutaMapa));
                } else {
                    lblMapa.setIcon(null);
                    JOptionPane.showMessageDialog(this,    "No    hay    mapa
disponible para " + nombrePais);
                }
            }
        }
```

```
    }

    private void btnReproducirHimno() {
        DefaultMutableTreeNode nodoSeleccionado = (DefaultMutableTreeNode)
arbol.getLastSelectedPathComponent();
        if (nodoSeleccionado != null) {
            String nombrePais = obtenerPais(nodoSeleccionado);
            if (!nombrePais.equals("")) {
                String rutaHimno= "src/himnos/" + nombrePais + ".mp3"; //
Carpeta donde estarán los himnos
                ReproductorAudio.reproducir(rutaHimno);
            }
        }
    }
}
```

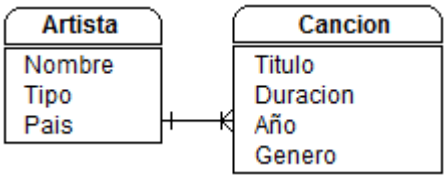
En la implementación de esta clase es importante resaltar lo siguiente:

- El objeto *objectMapper (*de la clase ***ObjectMapper*** que está incluida en la librería ***jackson-databind***) se encarga de convertir *JSON* en objetos Java y viceversa
- ***readValue(...)*** es un método de *ObjectMapper* que lee datos *JSON* y los convierte en una estructura de datos Java
- La instancia de ***File*** a partir del String *nombreArchivo* (que contiene la ruta del archivo JSON) permite el respectivo acceso en disco del archivo, el cual debe existir y contener un JSON válido
- ***getTypeFactory()*** obtiene un *TypeFactory*, que se usa para definir tipos genéricos en la deserialización (La **deserialización** es el proceso de convertir datos almacenados en un formato estructurado (como JSON, XML o binario) en un objeto de un lenguaje de programación, en este caso, un objeto en Java)
- La instrucción ***constructCollectionType(List.class, Pais.class)*** indica que el *JSON* representa una lista (objeto ***List***) de objetos *Pais*
- Los objetos ***DefaultMutableTreeNode*** son los componentes de la vista en árbol. Generalmente esta vista inicia con un nodo raíz del que se derivan los demás, en este caso en el primer nivel estarán los correspondientes a los países, luego en el segundo nivel las regiones y en el último nivel las ciudades. Los nodos de manera recursiva pueden contener a otros nodos para poder permitir estos niveles (se agregan con la instrucción ***add()***)
- La instrucción ***getParent()*** obtiene el nodo padre de un nodo. En este caso se usa para saber cual es el país de un nodo ciudad o región

La ejecución del programa luciría así:


División Política
Bélgica
Bermudas
Bután
Bolivia
Bosnia y Herzegovina
Botsuana
Isla Bouvet
Brasil
Belice
Territorio Británico del Océano Índi
Islas Salomón
Islas Vírgenes Británicas
Brunéi
Bulgaria
Myanmar
Burundi
Bielorrusia
Camboya
COLOMBIA
Boa Vista
AMAPÁ
RORAIMA
Macapá
Quito
ECUADOR
Guayaquil
AMAZONAS
SELVAS
PARÁ
São Luís
MARANHÃO
CEARÁ
PIAUÍ
Porto Velho
TOCANTINS
Trujillo
ACRE
PERÚ
RONDÔNIA
BRASIL
PLANALTO DO BRASIL
BAHIA
Lima
MATO GROSSO
Rio Tocantins
Paraguay
Brasília
Lago Titicaca
La Paz
Arequipa
Araguaia
MINAS GERAIS
BOLIVIA
Sucre
Belo Horizonte
MATO GROSSO DO SUL
OCÉANO PACÍFICO
CORDILLERA DE LOS ANDES
SÃO PAULO
Rio Grande
PARAGUAY
São Paulo
Rio de Janeiro
Antofagasta
Salta
Asunción

2. La información de una colección de música puede almacenarse en dos estructuras que recopilen los datos de los artistas y los datos de las canciones, entre las cuales debe haber una asociación:



La asociación entre las estructuras es una relación *Uno a muchos. Para e*sto puede utilizarse un archivo JSON como el siguiente:

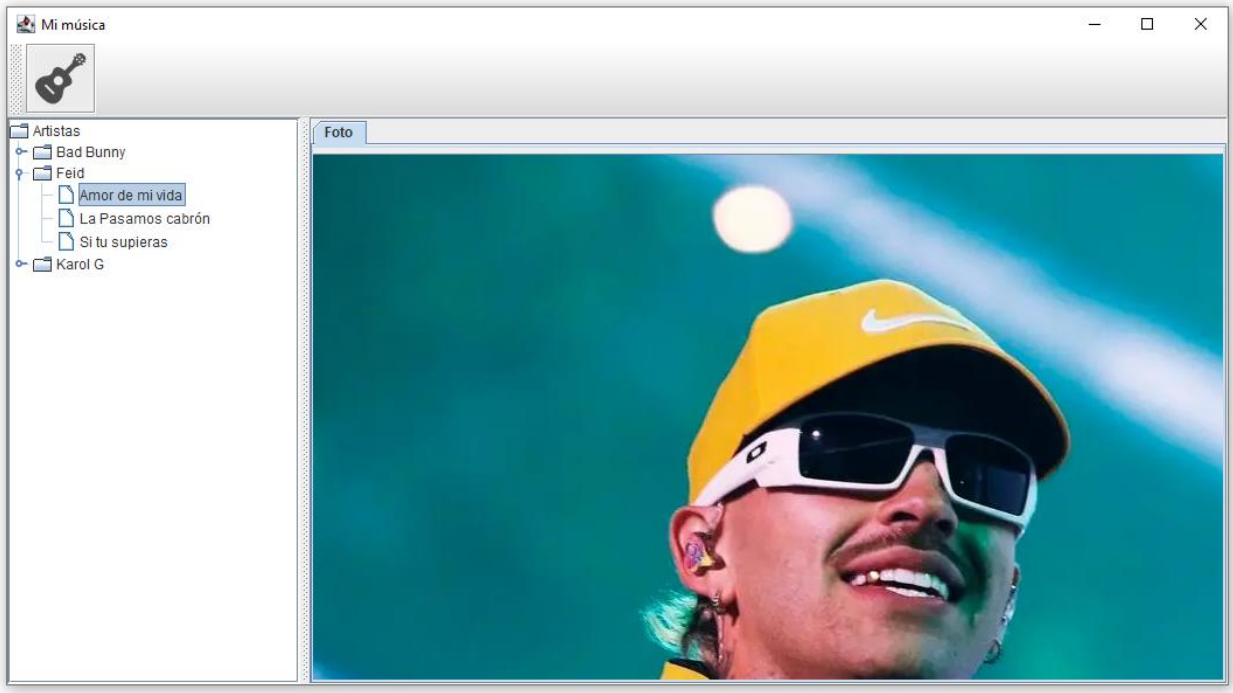
Musiteca.json

```
{
    "artistas": [
        {
            "nombre": "Bad Bunny",
            "tipo": "Solista",
            "pais": "Puerto Rico",
            "canciones": [
                {
                    "titulo": "Callaita",
                    "duracion": "4:12",
                    "año": "2019",
                    "genero": "Reggaeton"
                },
                {
                    "titulo": "Titi me preguntó",
                    "duracion": "4:04",
                    "año": "2022",
                    "genero": "Reggaeton"
                }
            ]
        },
        {
            "nombre": "Feid",
            "tipo": "Solista",
            "pais": "Colombia",
            "canciones": [
                {
                    "titulo": "Amor de mi vida",
                    "duracion": "3:11",
```

En este archivo se puede observar hay un arreglo **"*artistas*"** que incluye objetos "***artista"*** cada uno de los cuales incluyen no sólo los datos del artista, sino también un arreglo "**canciones**" de objetos ***"canción"***, los cuales incluyen los datos de las canciones. Este anidamiento cumple con el esquema mostrado anteriormente de que a *un Artista le corresponden muchas Canciones*.

Se debe realizar una aplicación que acceda el archivo JSON con la información antes planteada y permitirle al usuario:

- Listar los artistas y sus canciones en una vista en árbol
- Seleccionar un artista y mostrar su imagen
- Seleccionar una canción y poderla reproducir


Mi música
Artistas
Bad Bunny
Feid
Amor de mi vida
La Pasamos cabrón
Si tu supieras
Karol G
Foto