Acceso a Datos

Práctica Final Tema 1 – Arkanoid

Nathan

González Mercado

2ºDAM

Índice

1. Introducción

En esta práctica se nos ha encargado completar el editor de niveles del juego **Arkanoid**. El proyecto está casi terminado, teniendo la clase “Level” que representa un nivel, el cual cuenta con:

- Size: proveniente de la clase Size, representa el tamaño del juego, aunque esté está limitado.
- Blocks: Matriz de la clase Block, que corresponde a los ladrillos del juego.
- Time: Tiempo máximo para superar un nivel, dicho tiempo se almacena en segundos y es de tipo double
- Sound: Dirección a la canción que sonará en el nivel, este se almacena como cadena.
- Name: Corresponde al nombre que tiene el nivel y se almacena como una cadena
- BackgrounPosition: Indica el lugar de inicio del fondo y es de tipo Point2D.

En la clase App tenemos un atributo de tipo Level sobre el cual la interfaz gráfica interactúa pudiendo asignar música, crear ladrillos o asignar el tiempo. En un principio en dicha clase no modificaremos código si no que añadiremos.

Ficheros entregados:

- Este fichero PDF
- Fichero comprimido con el proyecto, así como una copia de este PDF.

2. Realización de la práctica.

2.1. Parte 1. Serializando y deserializando el nivel.

Añadir el código necesario para serializar y deserializar en ficheros el nivel de la aplicación en JSON y XML, asociando estas acciones a los botones del menu

- Save Level.
- Load Level.

Para ello comenzaremos con serializar, es decir, guardar el level (Save Level). Para ello en la clase Level deberemos crear el método estático Save, el cuál recibe un fichero y un Level.

Lo primero de todo sería saber qué extensión usaremos, para ello obtenemos el nombre del fichero y lo separamos por el punto, por ejemplo, si el fichero se llama nivel1.xml tendríamos 2 partes, nivel 1 y xml. Registramos la extensión como una String y en base a eso lo mandaremos al método correspondiente, saveXML o saveJSON, o en caso de ser una extensión no valida devolveremos un error.

El método save, incluyendo JSON y XML, se nos queda de la siguiente manera:

```
public static void save(File fichero, Level level) {

    String extension = "";
    if (fichero.getName().endsWith(".xml")) {
        extension = "xml";
    } else if (fichero.getName().endsWith(".json")) {
        extension = "json";
    } else {
        extension = "";
    }

    switch (extension) {
        case "xml":
            saveXML(fichero, level);
            break;
        case "json":
            saveJSON(fichero, level);
            break;
```

```
        default:
            System.out.println("Error");
    }
```

Primero conseguimos la extensión del fichero para saber qué tipo es, de manera provisional, se ha asignado que si no es una extensión valida. A su vez, los métodos saveXML y saveJSON se ven de la siguiente manera:

```
private static void saveXML(File fichero, Level level) {
    try {
        JAXBContext contexto =
JAXBContext.newInstance(Level.class);
        Marshaller marshaller = contexto.createMarshaller();

marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT,
Boolean.TRUE);
        marshaller.marshal(level, fichero);
    } catch (JAXBException e) {
        e.printStackTrace();
    }
}

private static void saveJSON (File fichero, Level level) {
    GsonBuilder gb = new GsonBuilder();

    gb.registerTypeAdapter(Point2D.class, new
AdapterPoint2DJSON());

    Gson gson = gb.create();
    String json = gson.toJson(level);

    try {
        BufferedWriter bw = new BufferedWriter(new
FileWriter(fichero));
        bw.write(json);
        bw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Por último, en la clase App.java deberemos agregar lo siguiente (lineas 203-212):

```
MenuItem saveLevelMenuItem = new MenuItem("Save Level");
saveLevelMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    File file = fileChooser.showSaveDialog(scene.getWindow());
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter("XML", "*.xml"),
            new FileChooser.ExtensionFilter("Json", "*.json"),
            new FileChooser.ExtensionFilter("Bin", "*.bin")
    );
    Level.save(file, this.level);
```

Donde nosotros deberemos agregar Level.save para llamar al método save que nos guardará el this.level en el fichero file.

A continuación, para los Load Level deberemos crear una clase load, la cuál se vera de la siguiente manera:

```
public static Level load(File fichero) {
    String extension = "";
    if (fichero.getName().endsWith(".xml")) {
        extension = "xml";
    } else if (fichero.getName().endsWith(".json")) {
        extension = "json";
    } else {
        extension = "";
    }

    switch (extension) {
        case "xml":
            return loadXML(fichero);
        case "json":
            return loadJSON(fichero);
        default:
            System.err.println("Error");
            return null;
    }
}
```

Donde primero obtendremos la extensión de nuestro fichero y según el resultado del mismo llamará a un método u otro. De forma provisional se indicó que por defecto nos diga error y devuelva un valor null.

Después, encontramos las clases loadXML y loadJSON que ambas reciben fichero se verán así:

```
private static Level loadXML(File fichero){
    Level l;
    try {
        JAXBContext contexto =
JAXBContext.newInstance(Level.class);
        Unmarshaller unmarshaller =
contexto.createUnmarshaller();

        l = (Level) unmarshaller.unmarshal(fichero);
        return l;
    } catch (JAXBException e) {
        e.printStackTrace();
        return null;
    }
}

private static Level loadJSON(File fichero) {
    Level l;
    try {
        FileReader fr = new FileReader(fichero);
        GsonBuilder gb = new GsonBuilder();

        gb.registerTypeAdapter(Point2D.class, new
AdapterPoint2DJSON());

        Gson gson = gb.create();

        l = gson.fromJson(fr,Level.class);
        return l;
    } catch (FileNotFoundException e) {
        throw new RuntimeException(e);
    }
}
```

Esto nos permitirá cargar ficheros de tipo XML haciendo uso de Unmarsheller mientras que en JSON haremos uso de gson.fromJson para leer el fichero.

Por último, en la clase App el apartado del menú de Load Level se verá de la siguiente manera (líneas 216 -230):

```
MenuItem loadLevelMenuItem = new MenuItem("Load Level");
loadLevelMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter("XML", "*.xml*"),
            new FileChooser.ExtensionFilter("Json", "*.json"),
            new FileChooser.ExtensionFilter("Bin", "*.bin")
    );
    File file = fileChooser.showOpenDialog(scene.getWindow());

    this.levelsPanel.reset();

    this.level = Level.load(file);
    this.editor.setLevel(this.level);

thi
s.levelsPanel.getListlevels().getItems().add(this.level.getNam
e());
});
```

Donde primero reseteamos el panel de niveles para que se quede vacío y no de problemas con el nuevo. Después indicamos que el nivel es el uso del método Load de la clase Level mandándole el file. Tras ello al editor le indicamos que el nivel a mostrar será el nivel que se ha cargado y por último los añadiremos al listview el nombre del nivel. Con esto las clases de Guardar y Cargar ya serán completamente funcionales.

2.2 Parte 2 - Añadiendo niveles

Hasta ahora contamos con un único nivel, lo que no parece una buena opción. Definir una nueva clase, Levels, que pueda gestionar un conjunto de niveles. En la clase LevelPane se encuentra limitada la opción de añadir niveles a 1, eliminar la condición para poder añadir más. De manera original se encuentra así:

```
if (result.isPresent()) {
    result.get();
    //solo se deja uno, modificar para la parte 2
    if(this.listlevels.getItems().size()==0) {
        if (result.get() != "") {
            //se añade y se deja seleccionado
            this.listlevels.getItems().add(result.get());

thi
s.listlevels.getSelectionModel().select(result.get());//.selec
t(this.listlevels.getSelectionModel());
            //se llama a la capa superior para informar
            if (this.onadd != null) {
                this.onadd.accept(result.get());
            }
        }
    }
}
```

Al quitar la limitación se nos queda de la siguiente manera:

```
if (result.isPresent()) {
    result.get();
    //solo se deja uno, modificar para la parte 2
        if (result.get() != "") {
            //se añade y se deja seleccionado
            this.listlevels.getItems().add(result.get());

thi
s.listlevels.getSelectionModel().select(result.get());//.selec
t(this.listlevels.getSelectionModel());
            //se llama a la capa superior para informar
            if (this.onadd != null) {
                this.onadd.accept(result.get());
```

```
        }
    }
}
```

La clase Levels se ve de la siguiente manera:

```
package ies.pedro.model;

import jakarta.xml.bind.annotation.XmlRootElement;

import java.util.ArrayList;

@XmlRootElement
public class Levels {
    private ArrayList<Level> listaLevels;

    //Constructores
    public Levels() {
        this.listaLevels = new ArrayList<Level>();
    }

    public Levels(ArrayList<Level> listaLevels) {
        this.listaLevels = listaLevels;
    }

    public void addLevel(Level level) {
        this.listaLevels.add(level);
    }

    public void deleteLevel(String nombreLevel) {
        for (int i = 0; i < this.listaLevels.size(); i++) {
            if
(this.listaLevels.get(i).getName().equals(nombreLevel)) {
                System.out.println("Nivel: " +
this.listaLevels.get(i).getName() + " eliminado");
                this.listaLevels.remove(i);
                break;
            }
        }
    }

    public Level changeLevel(String nombre) {
        for (int i = 0; i < this.listaLevels.size(); i++) {
```

```
if(this.listaLevels.get(i).getName().equals(nombre)) {
                return this.listaLevels.get(i);
            }
        }
        return null;
    }




    //Getters y Setters
    public ArrayList<Level> getListaLevels() {
        return listaLevels;
    }

    public void setListaLevels(ArrayList<Level> listaLevels) {
        this.listaLevels = listaLevels;
    }
}
```

Además, para arreglar su funcionamiento deberemos editar en Level Panel el funcionamiento de los botones Delete y Reset:

```
this.delete = new Button("Delete");
//similar a los eventos
this.delete.setOnMouseClicked(mouseEvent -> {
    if
(this.listlevels.getSelectionModel().getSelectedItem() !=
null) {
        if(this.ondelete!=null) {

thi
s.ondelete.accept(this.listlevels.getSelectionModel().getSelec
tedItem().toString());
        }

thi
s.listlevels.getItems().remove(this.listlevels.getSelectionMod
el().getSelectedItem());
```

```
    }
});
```

Donde por defecto, en this.ondelete.accept viene la palabra vacío indicaremos que aceptara borrar el nombre del objeto seleccionado del listview. Esto permitirá borrar de nuestra lista de niveles el nivel borrado.

Para reset, cambiaremos igualmente la palabra vacío por el nombre del objeto seleccionado en el listview.

Por último, en App deberemos editar varias cosas, empezando por crear un elemento Levels:

```
private Levels levels;
```

Además, deberemos inicializar este de la siguiente manera:

```
public App() {
    super();
    this.level = new Level();
    fileChooser = new FileChooser();
    this.levels = new Levels();
}
```

Después de eso, deberemos de en el método createLevelPanel editarlo para que haga uso del objeto Levels de la siguiente manera:

```
private LevelsPanel createLevelPanel() {
    LevelsPanel levelspanel = new LevelsPanel();
    levelspanel.init();
    levelspanel.setOnadd((s) -> {
        this.level = new Level(s);
        this.levels.addLevel(this.level);
        this.editor.setLevel(level);
        this.deleteMediaplayer();
    });
    levelspanel.setOndelete((s) -> {
        this.deleteMediaplayer();
        this.level=null;
        this.levels.deleteLevel(s);
```

```
        this.editor.setLevel(null);
    });
    levelspanel.setOnseleted((s) -> {

        this.deleteMediaplayer();
        this.level = this.levels.changeLevel(s);
        this.editor.setLevel(this.level);
        System.out.println("Se ha cambiado de nivel");
    });
```

Indicándole que para los distintos botones deberemos hacer uso del objeto levels. Para crear los añadiremos a la lista del objeto Levels, al borrarlo los eliminaremos de dicha lista y por último al seleccionar, se cambiará la vista del nivel de uno a otro, de manera que, si yo tengo level 1 con un fondo rojo y level 2 con un fondo azul, si cambio a level 1 deberemos ver su fondo rojo.

2.3. Parte 3 – Guardando niveles.

Añadir el código necesario para poder almacenar un conjunto de niveles en JSON y XML, de igual manera, se tendrá que poder cargar ficheros en esos formatos para poder seguir con la edición de los niveles.

Para conseguir esto, deberemos modificar la clase Levels creada anteriormente. A dicha clase, le deberemos agregar 2 métodos públicos estáticos, los cuales son:

- Save: Nos permite guardar la lista de niveles creados, separando la forma de guardado en distintos métodos privados que realizarán el guardado para un formato específico en base a la extensión del fichero, encontrándonos con guardados para XML y JSON. Por tanto, el método Save contará con otros 2 métodos, saveXML y saveJSON, donde el método save se ve de la siguiente manera:

```
public static void save(File fichero, Levels levels) { 1 usage
    String extension = "";
    if (fichero.getName().endsWith(".xml")) {
        extension = "xml";
    } else if (fichero.getName().endsWith(".json")) {
        extension = "json";
    } else {
        extension = "";
    }

    switch (extension) {
        case "xml":
            saveXML(fichero,levels);
            break;
        case "json":
            saveJSON(fichero,levels);
            break;
        default:
            System.out.println("Error");
    }
}
```

Donde primero obtendremos la extensión del fichero que obtenemos como parámetro y luego según dicha extensión realizamos distintos

métodos, donde ambos métodos reciben de parámetro la lista de niveles y el fichero.

- SaveXML: Permite guardar los niveles en XML, el código que realizado para esto se ve así:

```
private static void saveXML(File fichero, Levels levels) {  1 usage
    try {
        JAXBContext contexto = JAXBContext.newInstance(Levels.class);
        Marshaller marshaller = contexto.createMarshaller();
        marshaller.setProperty(Marshaller.JAXB_FORMATTED_OUTPUT, Boolean.TRUE);
        marshaller.marshal(levels, fichero);
    } catch (JAXBException e) {
        e.printStackTrace();
    }
}
```

Como se puede ver, sigue la misma lógica usada anteriormente para guardar un XML para un único nivel, siendo la única diferencia que la instancia ahora es de la clase Levels.

- SaveJSON: Permite guardar los niveles en JSON, el código que realiza esto se ve da la siguiente manera:

```
private static void saveJSON(File fichero, Levels levels) {  1 usage
    GsonBuilder gb = new GsonBuilder();
    gb.registerTypeAdapter(Point2D.class, new AdapterPoint2DJSON());

    Gson gson = gb.create();
    String json = gson.toJson(levels);
    try {
        BufferedWriter bw = new BufferedWriter(new FileWriter(fichero));
        bw.write(json);
        bw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Como se puede ver, sigue la misma lógica usada anteriormente para guardar un único JSON, haciendo uso tambien del adaptador creado previamente.

- Load: Será el método que nos permitirá cargar las distintas listas de niveles que hayamos creado anteriormente. Al igual que anteriormente, dentro de dicho método público se llamará a 2 métodos distintos según su extensión, loadXML y loadJSON. El método Load se verá de la siguiente manera, siguiendo la misma lógica que el save:

```
public static Levels load(File fichero) {  2 usages
    String extension = "";
    if (fichero.getName().endsWith(".xml")) {
        extension = "xml";
    } else if (fichero.getName().endsWith(".json")) {
        extension = "json";
    } else {
        extension = "";
    }

    switch (extension) {
        case "xml":
            return loadXML(fichero);
        case "json":
            return loadJSON(fichero);
        default:
            System.err.println("Error");
            return null;
    }
}
```

  - loadXML: Se encarga de cargar los ficheros XML, sigue la misma lógica usada para cargar un único nivel de XML, viéndose de la siguiente manera:

```
private static Levels loadXML(File fichero) {  1 usage
    Levels levels;
    try {
        JAXBContext contexto = JAXBContext.newInstance(Levels.class);
        Unmarshaller unmarshaller = contexto.createUnmarshaller();
        levels = (Levels) unmarshaller.unmarshal(fichero);
        return levels;
    } catch (JAXBException e) {
        return null;
    }
}
```

- loadJSON: Es el encargado de cargar los ficheros de formato JSON, los cuáles comparten la misma lógica que un único nivel, y se ve de la siguiente manera:

```
private static Levels loadJSON(File fichero) {  1 usage
    Levels l;
    try {
        FileReader fr = new FileReader(fichero);
        GsonBuilder gb = new GsonBuilder();

        gb.registerTypeAdapter(Point2D.class, new AdapterPoint2DJSON());
        Gson gson = gb.create();

        l = gson.fromJson(fr, Levels.class);
        return l;
    } catch (FileNotFoundException e) {
        return null;
    }
}
```

Por último, deberemos editar en la clase App los métodos Save y Load, los cuales son encargados de guardar y cargar la lista de niveles. Estos se verán de la siguiente manera:

```
MenuItem saveMenuItem = new MenuItem( s: "Save");
saveMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    File file = fileChooser.showSaveDialog(scene.getWindow());
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter( s: "XML",  ...strings: "*.xml"),
            new FileChooser.ExtensionFilter( s: "Json",  ...strings: "*.json"),
            new FileChooser.ExtensionFilter( s: "Bin",  ...strings: "*.bin")
    );
    if (file != null) {
        if (file.getName().endsWith(".xml") || file.getName().endsWith(".json")) {
            Levels.save(file, this.levels);
        } else {
            fileError();
        }
    } else {
        fileError();
    }
});
```

Primero comprobaremos que el fichero no sea nulo, en caso de no serlo, comprobaremos que cumpla las extensiones necesarias y en caso de no hacerlo devolverá error. El método fileError mostrará una alerta avisando del error correspondiente, en este caso que el fichero seleccionado no tiene las extensiones indicadas.

Método Load:

```
MenuItem loadMenuItem = new MenuItem( s: "Load");
loadMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter( s: "XML",  ...strings: "*.xml*"),
            new FileChooser.ExtensionFilter( s: "Json",  ...strings: "*.json"),
            new FileChooser.ExtensionFilter( s: "Bin",  ...strings: "*.bin")
    );
    File file = fileChooser.showOpenDialog(scene.getWindow());
    if (file != null) {
        if (Levels.load(file) != null) {
            this.levels = Levels.load(file);
            this.level = this.levels.getListaLevels().get(0);
            this.editor.setLevel(this.level);
            for (int i = 0; i < this.levels.getListaLevels().size(); i++) {
                this.levelsPanel.getListlevels().getItems().add(this.levels.getListaLevels().get(i).getName());
            }
        } else {
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("Fallo al cargar");
            alert.setContentText("El fichero no contiene datos de ningun nivel o estan dañados.");
            alert.showAndWait();
        }
    }else {
        fileError();
    }
```

Tras elegir fichero, comprobaremos que el fichero no es nulo y tras ello haremos la comprobación de que cargar los niveles de dicho fichero no devuelve nulo, es decir, que existen datos de niveles en el fichero, y si no es nulo cargaremos todos los niveles, si no mostrará una alerta avisando de que el fichero no tiene niveles o están dañados.

Como bien se ha mencionado anteriormente, el método fileError nos devolverá una alerta para cuando se encuentre algún error. Este se ve de la siguiente manera:

```
private void fileError() {
    Alert alert = new Alert(AlertType.ERROR);
    alert.setTitle("Error");
    alert.setHeaderText("Fallo al cargar");
    alert.setContentText("Se ha de seleccionar un fichero con
las extensiones JSON o XML.");
    alert.showAndWait();
}
```

Anteriormente se dejó un mensaje de “Error” para los métodos de guardar un único nivel y cargar un único nivel. Dicho mensaje ha sido sustituido por esta alerta, quedando de la siguiente manera:

SaveLevel:

```
MenuItem saveLevelMenuItem = new MenuItem("Save Level");
saveLevelMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    File file = fileChooser.showSaveDialog(scene.getWindow());
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter("XML", "*.xml"),
            new FileChooser.ExtensionFilter("Json", "*.json"),
            new FileChooser.ExtensionFilter("Bin", "*.bin")
    );
    if (file != null) {
        if (file.getName().endsWith(".xml") ||
file.getName().endsWith(".json")) {
            Level.save(file, this.level);
        } else {
            fileError();
        }
    } else {
        fileError();
    }
```

LoadLevel:

```
MenuItem loadLevelMenuItem = new MenuItem("Load Level");
loadLevelMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter("XML", "*.xml*"),
            new FileChooser.ExtensionFilter("Json", "*.json"),
            new FileChooser.ExtensionFilter("Bin", "*.bin")
    );
    File file = fileChooser.showOpenDialog(scene.getWindow());

    if (file != null) {
        if (Level.load(file) != null) {
            this.levelsPanel.reset();
            this.level = Level.load(file);
            this.editor.setLevel(this.level);

thi
s.levelsPanel.getListlevels().getItems().add(this.level.getNam
e());
        } else {
            Alert alert = new Alert(AlertType.ERROR);
            alert.setTitle("Error");
            alert.setHeaderText("Fallo al cargar");
            alert.setContentText("El fichero no contiene datos
de ningun nivel o estan dañados.");
            alert.showAndWait();
        }
    } else {
        fileError();
    }

});
```

Con este tendremos todos los posibles errores para los guardados controlados y podremos guardar y cargar un nivel y una lista de niveles.

2.4 Parte 4 – Transformar a HTML

Crea el código necesario, para que a partir de un fichero XML de nivel, realice la transformación, por ejemplo, en tabla, para ver el nivel, incluyendo los datos del nivel como el tiempo, el fondo y la cadena con la canción. Tras ello hacer lo mismo, pero con niveles.

Para ello, dentro de la carpeta utils he creado la clase "XMLtoHTML" la cuál será la encargada de realizar la transformación de los ficheros XML a HTML.

Centrándonos de momento en un único nivel, nos encontramos que el código es el siguiente:

```
public class XMLtoHTML {

    public static void leveltoHTML(File levelXML) {
        String parent = levelXML.getParent();
        String name = levelXML.getName().replace(".xml", "");
        String dir = parent + "/htmls/" + name + ".html";
        File html = new File(dir);

        File xslF = new File(parent + "/transformer.xsl");
        StreamSource xls = new StreamSource(xslF);
        try {
        StreamSource xml = new StreamSource(levelXML);
        TransformerFactory tf =
TransformerFactory
.newInstance("org.apache.xalan.processor.TransformerFactoryImp
l", null);

        StreamResult result = new StreamResult(html);
        Transformer transformer = tf.newTransformer(xls);
        transformer.transform(xml, result);
        } catch (TransformerException e) {
            e.printStackTrace();
        }
    }
```

Arriba se puede ver que hemos creado el método estático **leveltoHTML** que recibe como parámetro un nivel en XML.

Lo primero que hacemos es obtener la ruta del padre del fichero XML y tras ello creamos el nombre, quitándole la extensión del fichero. Por último, creamos la String dir que será el directorio donde guardaremos nuestro HTML, la creamos

haciendo uso del padre, dentro de la carpeta htmls y de nombre tendrá el parámetro name creado anteriormente con la extensión .html. Por último, creamos un fichero que tendrá la ruta de HTML

Tras ello creamos el fichero xslF que tendrá como ruta donde se encuentre nuestro fichero XSL, en mi caso estoy usando la ruta parent y el fichero transformer.xsl

Después creamos los StreamSource de XML y XSLy tras ello creamos el TransformerFactory, obtenido en los apuntes del temario.

Seguido de esto creamos un StreamResult llamado resultado que usará el fichero HTML creado anteriormente.

Por último, creamos el transformador usando el transformerFactory anterior mandándole el XLS y después hacemos un transform de ese XML y le indicamos que se mande a result.

A continuación, veremos el cómo debe verse el fichero XSL, el cual se ve así:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="no"/>
  <xsl:template match="/level">
    <html lang="es">
        <head>
          <meta charset="UTF8"/>
          <title>Level</title>
        </head>
    <body>
      <h2>
        Listado de niveles
      </h2>
      <xsl:for-each select="//level">
      <h3>
        Nombre:
      </h3>
      <p>
        <xsl:value-of select="./name"/>
      </p>
      <h3>
        Tamano:
```

```
        </h3>
        <p>
            Altura: <xsl:value-of select="./size/height"/>
            Anchura: <xsl:value-of select="./size/width"/>
        </p>

        <h3>
            Sonido:
        </h3>
        <p>
             <xsl:value-of select="./sound"/>
        </p>
        <h3>
            Tiempo:
        </h3>
        <p>
             <xsl:value-of select="./time"/>
        </p>
        <h3>
            Posicion del Fondo:
        </h3>
        <p>
            <xsl:value-of select="./backgroundPosition"/>
        </p>

        <h3>
            Previsualización del nivel:
        </h3>
        <table border="1">
            <tbody>
                <xsl:for-each select=".//blocks">
                    <tr>
                        <xsl:for-each select=".//item">
                            <xsl:variable name="fondo">
                                <xsl:choose>
                                    <xsl:when test="not(value)">C0C0C0</xsl:when>
                                    <xsl:when test="value ='Blue'">0070FF</xsl:when>
                                    <xsl:when test="value ='Red'">DC143C</xsl:when>
                                    <xsl:when test="value ='White'">FFFFFF</xsl:when>
                                    <xsl:when test="value ='Cyan'">00FFFF</xsl:when>
                                    <xsl:when test="value ='Magenta'">FF00FF</xsl:when>
                                    <xsl:when test="value ='Yellow'">FFFF00</xsl:when>
                                    <xsl:when test="value ='Orange'">FFA500</xsl:when>
                                    <xsl:when test="value ='Green'">008000</xsl:when>
```

```
                    </xsl:choose>
                </xsl:variable>
                <td bgcolor="{$fondo}">
                    <xsl:choose>
                        <xsl:when test="not(value)">null</xsl:when>
                        <xsl:when test="value ='Blue'">Blue</xsl:when>
                        <xsl:when test="value ='Red'">Red</xsl:when>
                        <xsl:when test="value ='White'">White</xsl:when>
                        <xsl:when test="value ='Cyan'">Cyan</xsl:when>
                        <xsl:when test="value ='Magenta'">Magenta</xsl:when>
                        <xsl:when test="value ='Yellow'">Yellow</xsl:when>
                        <xsl:when test="value ='Orange'">Orange</xsl:when>
                        <xsl:when test="value ='Green'">Green</xsl:when>
                    </xsl:choose>
                </td>
            </xsl:for-each>
        </tr>
    </xsl:for-each>
    </tbody>
    </table>
    </xsl:for-each>
  </body>
  </html>
 </xsl:template>
</xsl:stylesheet>
```

Por último, en App buscaremos el botón Level XMLToHTML del menu y le agregaremos la llamada del método, quedando así:

```
MenuItem xmlToHMTLLevelMenuItem = new MenuItem("Level XMLToHTML");
xmlToHMTLLevelMenuItem.setOnAction(actionEvent -> {
    final FileChooser fileChooser = new FileChooser();
    fileChooser.getExtensionFilters().addAll(
            new FileChooser.ExtensionFilter("XML", "*.xml*"),
            new FileChooser.ExtensionFilter("Json", "*.json"),
            new FileChooser.ExtensionFilter("Bin", "*.bin")
    );
    File file = fileChooser.showOpenDialog(scene.getWindow());
    if (file != null) {
        XMLtoHTML.leveltoHTML(file);
    } else {
        fileError();
    }
});
```

Donde en base al fichero obtenido, que será el XML de nivel, llamaremos al método para realizarlo. Si todo es correcto nos quedará un fichero html de la siguiente :

Listado de niveles

Nombre:

Juanito

Tamano:

Altura: 30 Anchura: 14

Sonido:

/home/natgonmer/Documents/AD/ArkanoidNGM/sounds/toki.mp3

Tiempo:

175.34939759036146

Posicion del Fondo:

0.0,0.0

Previsualización del nivel:

Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Blue	Blue	Red	Red	White	White	Cyan	Cyan	Yellow	Yellow	Magenta	Magenta	Orange	null
Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	Green	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null

A continuación, se mostrará lo realizado para una lista de niveles:

```
public static void levelstoHTML(File levelsXML) {
    String parent = levelsXML.getParent();
    String name = levelsXML.getName().replace(".xml", "");
    String dir = parent + "/htmls/" + name + ".html";
    File html = new File(dir);

    File xslF = new File( parent + "/transformerLevels.xsl");
    StreamSource xls = new StreamSource(xslF);

    try {
        StreamSource xml = new StreamSource(levelsXML);
        TransformerFactory tf =
TransformerFactory.newInstance("org.apache.xalan.processor.TransformerFactoryImpl", null);

        StreamResult result = new StreamResult(html);
        Transformer transformer = tf.newTransformer(xls);
        transformer.transform(xml, result);
    } catch (TransformerException e) {
        e.printStackTrace();
    }

}
```

Donde la mayor diferencia es el fichero XSL al que llamamos, el cuál es transformerLevels.xsl.

Dicho fichero XSL se verá de la siguiente manera:

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:output method="html" indent="no"/>
   <xsl:template match="/levels">
      <html lang="es">
            <head>
               <meta charset="UTF8"/>
               <title>Level</title>
            </head>
      <body>
         <h2>
            Listado de niveles
         </h2>
         <xsl:for-each select="//listaLevels">
         <h3>
            Nombre:
         </h3>
         <p>
            <xsl:value-of select="./name"/>
         </p>
         <h3>
            Tamano:
         </h3>
         <p>
            Altura: <xsl:value-of select="./size/height"/>
            Anchura: <xsl:value-of select="./size/width"/>
         </p>

         <h3>
            Sonido:
         </h3>
         <p>
             <xsl:value-of select="./sound"/>
         </p>
         <h3>
            Tiempo:
         </h3>
         <p>
             <xsl:value-of select="./time"/>
         </p>
         <h3>
            Posicion del Fondo:
```

```
        </h3>
        <p>
          <xsl:value-of select="./backgroundPosition"/>
        </p>

        <h3>
          Previsualización del nivel:
        </h3>
        <table border="1">
          <tbody>
            <xsl:for-each select=".//blocks">
              <tr>
                <xsl:for-each select=".//item">
                  <xsl:variable name="fondo">
                    <xsl:choose>
                      <xsl:when test="not(value)">C0C0C0</xsl:when>
                      <xsl:when test="value ='Blue'">0070FF</xsl:when>
                      <xsl:when test="value ='Red'">DC143C</xsl:when>
                      <xsl:when test="value ='White'">FFFFFF</xsl:when>
                      <xsl:when test="value ='Cyan'">00FFFF</xsl:when>
                      <xsl:when test="value ='Magenta'">FF00FF</xsl:when>
                      <xsl:when test="value ='Yellow'">FFFF00</xsl:when>
                      <xsl:when test="value ='Orange'">FFA500</xsl:when>
                      <xsl:when test="value ='Green'">008000</xsl:when>
                    </xsl:choose>
                  </xsl:variable>
                  <td bgcolor="{$fondo}">
                    <xsl:choose>
                      <xsl:when test="not(value)">null</xsl:when>
                      <xsl:when test="value ='Blue'">Blue</xsl:when>
                      <xsl:when test="value ='Red'">Red</xsl:when>
                      <xsl:when test="value ='White'">White</xsl:when>
                      <xsl:when test="value ='Cyan'">Cyan</xsl:when>
                      <xsl:when test="value ='Magenta'">Magenta</xsl:when>
                      <xsl:when test="value ='Yellow'">Yellow</xsl:when>
                      <xsl:when test="value ='Orange'">Orange</xsl:when>
                      <xsl:when test="value ='Green'">Green</xsl:when>
                    </xsl:choose>
                  </td>
                </xsl:for-each>
              </tr>
            </xsl:for-each>
          </tbody>
        </table>
```

```
        </xsl:for-each>
      </body>
      </html>
   </xsl:template>
</xsl:stylesheet>
```

Por último, en App igual que antes, llamamos al método que haga dicha transformación:

```
MenuItem xmlToHMTLLevelsMenuItem = new MenuItem("Levels XMLToHTML");
xmlToHMTLLevelsMenuItem.setOnAction(actionEvent -> {
   final FileChooser fileChooser = new FileChooser();
   fileChooser.getExtensionFilters().addAll(
         new FileChooser.ExtensionFilter("XML", "*.xml*"),
         new FileChooser.ExtensionFilter("Json", "*.json"),
         new FileChooser.ExtensionFilter("Bin", "*.bin")
   );
   File file = fileChooser.showOpenDialog(scene.getWindow());
   if (file != null) {
      XMLtoHTML.levelstoHTML(file);
   } else {
      fileError();
   }
});
```

Donde recibimos el fichero que contiene los levels XML y lo transformará en HTML, recibiendo un resultado similar a este:

Listado de niveles

Nombre:

nivel 1

Tamano:

Altura: 30 Anchura: 14

Sonido:

Tiempo:

0.0

Posicion del Fondo:

0.0,0.0

Previsualización del nivel:

Blue	Blue	Magenta	Magenta	null	null	null	null	null	null	null	null	null	null
Blue	Blue	Magenta	Magenta	null	null	null	null	null	null	null	null	null	null
Blue	Blue	Magenta	Magenta	null	null	null	null	null	null	null	null	null	null
Blue	Blue	Magenta	Magenta	null	null	null	null	null	null	null	null	null	null
Cyan	Cyan	Green	Green	null	null	null	null	null	null	null	null	null	null
Cyan	Cyan	Green	Green	null	null	null	null	null	null	null	null	null	null
Cyan	Cyan	Green	Green	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null

Nombre:

Nombre:

nivel 2

Tamano:

Altura: 30 Anchura: 14

Sonido:

Tiempo:

0.0

Posicion del Fondo:

232.0,232.0

Previsualización del nivel:

White	White	null	null	null	null	null	null	null	null	null	null	null	null
White	White	null	null	null	null	null	null	null	null	null	null	null	null
White	White	null	null	null	null	null	null	null	null	null	null	null	null
White	White	null	null	null	null	null	null	null	null	null	null	null	null
White	White	null	null	null	null	null	null	null	null	null	null	null	null
White	White	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null
null	null	null	null	null	null	null	null	null	null	null	null	null	null