

Asignatura:

Desarrollo e Integración de Software

Título del documento:

Práctica Final DIS

Preparado por:

Nombre

Sergio Luna, Daniel Sánchez,
Daniel Ojeda, Andrés Arcones

10/11/20

Nombre de fichero:

Práctica-1-Diseño de Interfaz.docx

Fecha:

10/11/20

Edición:

1

Página:

1/12



Registro de cambios

Ed.	Fecha	Cambio (Incluya el capítulo / subcapítulo y una corta descripción)
1	22/11/20	V.1

Índice

1	Introducción	4
1.1	Objetivos	4
1.2	Descripción del Proyecto	4

1 Introducción

1.1 Objetivos

- Aprender a manejar los comandos de Git
- Aprender a controlar un repositorio en equipo
- Aprender a leer un fichero con Java
- Aprender XML y DTD
- Aprender a trabajar con lectura de ficheros estructurados

1.2 Descripción del Proyecto

En esta práctica, debemos crear una aplicación Java, que lea un fichero JSON, creado a partir de un fichero XML y que nos muestre diversos contenidos.

Lo primero que hacemos es crear nuestro fichero XML, por ejemplo

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE videotecas SYSTEM "videotecas.dtd">
<videotecas>
  <videoteca>
    <nombre>Películas Star Wars</nombre>
    <ubicacion>Salon</ubicacion>
    <fecha_ult_actualizacion>14/11/2020</fecha_ult_actualizacion>
    <películas>
      <pelicula id="1" minutos="136" estreno="1999" >
        <titulo>Star Wars: Episodio I - La amenaza fantasma</titulo>
        <sinopsis>La trama describe la historia del maestro jedi Qui-Gon Jinn y de su aprendiz Obi-Wan Kenob
        </sinopsis>
        <enlace_IMDB>https://www.imdb.com/title/tt0120915/?ref_=nv_sr_srsrg_6</enlace_IMDB>
        <genero>Space Opera</genero>
        <reparto>
          <actor>
            <nombre>Liam Neeson</nombre>
            <enlace_wiki>https://es.wikipedia.org/wiki/Liam_Neeson</enlace_wiki>
          </actor>
          <actor>
            <nombre>Samuel Leroy Jackson</nombre>
            <enlace_wiki>https://es.wikipedia.org/wiki/Samuel_L._Jackson</enlace_wiki>
          </actor>
          <actor>

```

Una vez creado nuestro XML, vamos a crear nuestro proyecto usando Maven

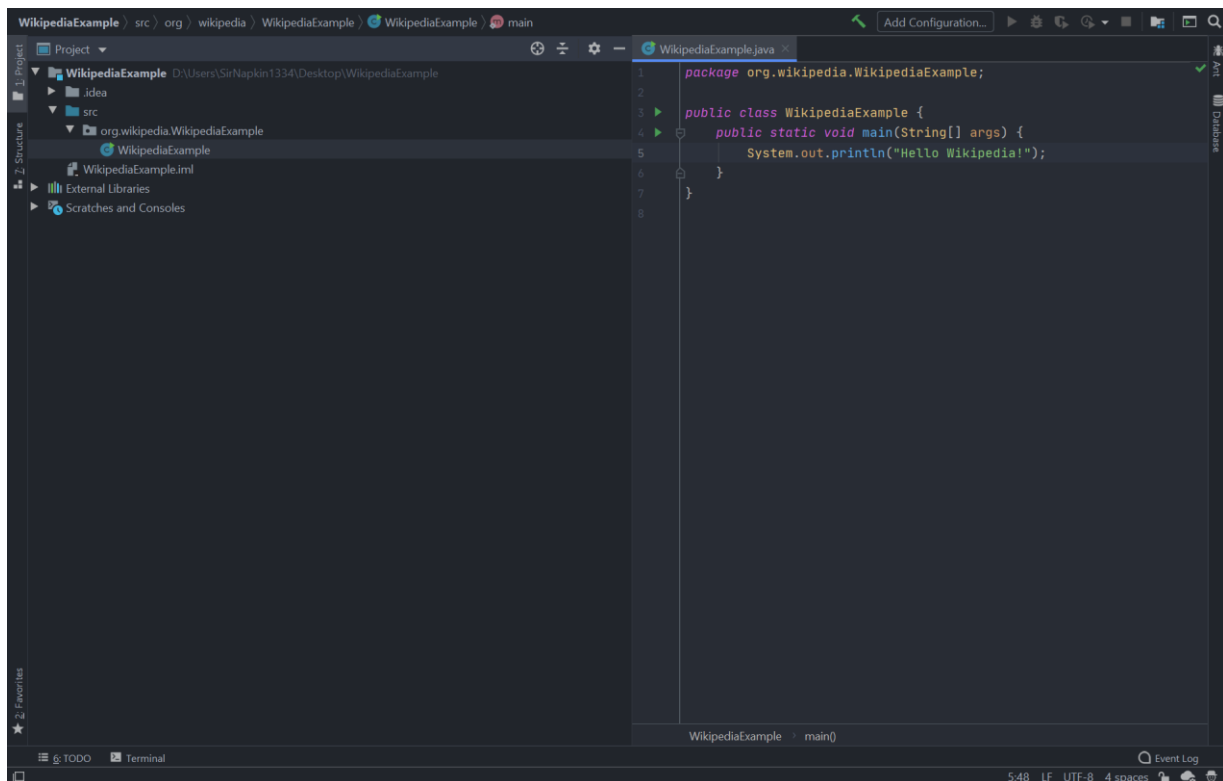
Ayudandonos de Maven, creamos nuestro fichero POM.XML, donde vamos a configurar las dependencias y plugins que tendrá nuestro proyecto.

```
<dependencies>
  <dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.11</version>
    <scope>test</scope>
  </dependency>
  <dependency>
    <groupId>com.google.code.gson</groupId>
    <artifactId>gson</artifactId>
    <version>2.8.5</version>
  </dependency>
  <dependency>
    <groupId>javax.xml.stream</groupId>
    <artifactId>stax-api</artifactId>
    <version>1.0-2</version>
  </dependency>
  <dependency>
    <groupId>org.jetbrains</groupId>
    <artifactId>annotations-java5</artifactId>
    <version>RELEASE</version>
    <scope>compile</scope>
  </dependency>
</dependencies>
```

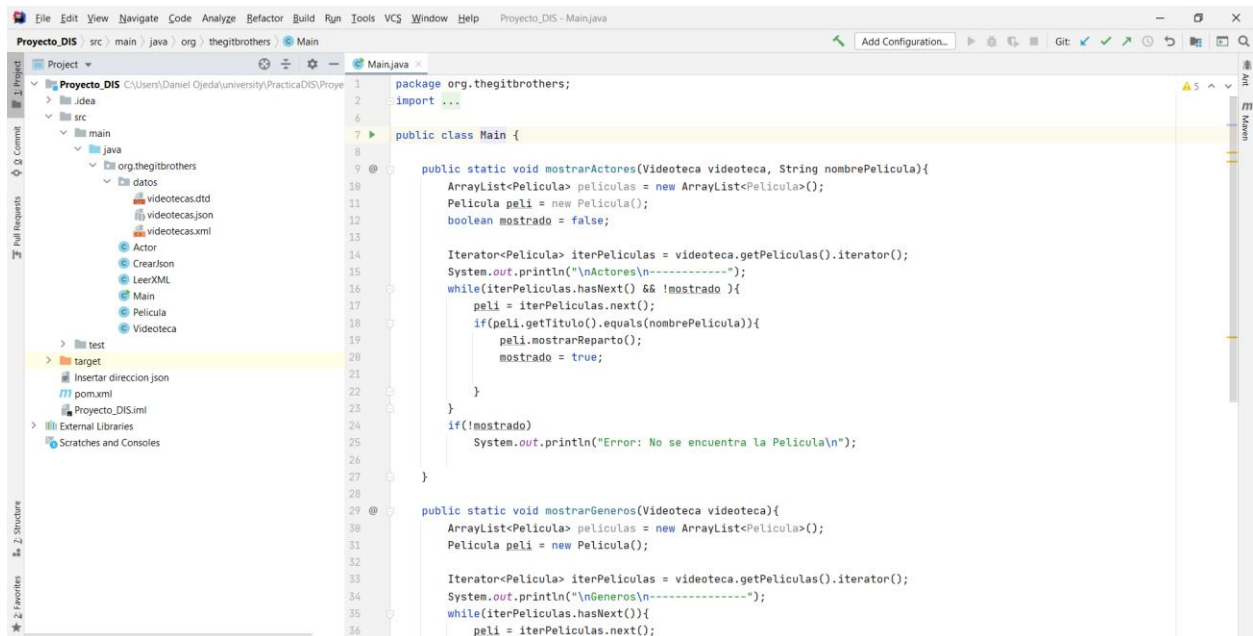
Señalamos en azul, las dos dependencias que hemos añadido, en este caso, la que nos permite trabajar con GSON y la que nos permite leer XML con Java.

Una vez hecho, nuestro POM.XML, podemos abrir el proyecto, a partir, de este fichero

En nuestro caso, vamos a usar IntelliJ, como entorno de desarrollo.

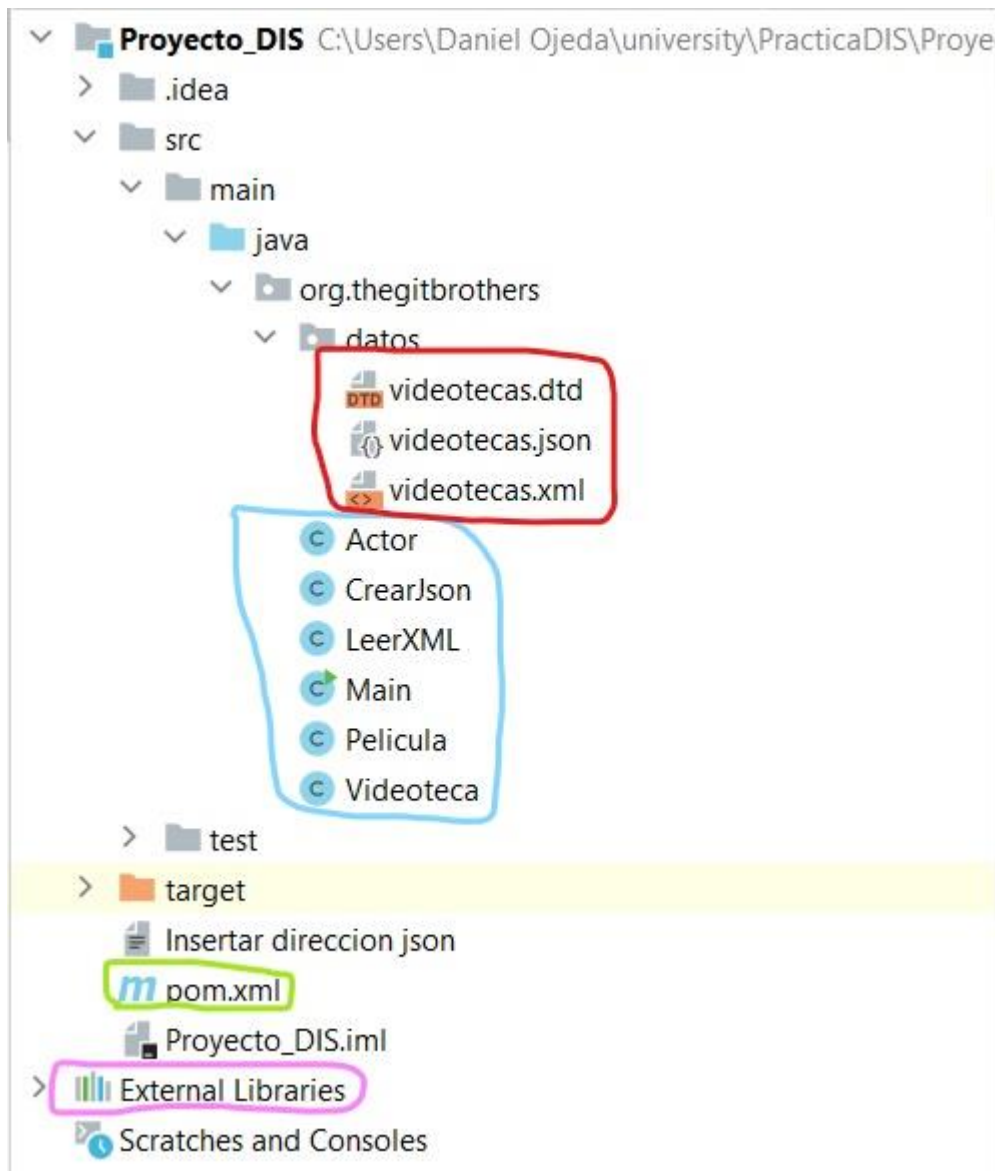


Ahora con nuestro entorno abierto, importamos nuestro proyecto, a partir del POM.XML



Ahora, vamos a describir, cómo está formado nuestra aplicación

Siguiendo las indicaciones del documento de la práctica, creamos varias clases, para estructurar mejor la aplicación y después en el Main, invocamos a la clase correspondiente.



En **rojo** tenemos los distintos ficheros fuente de nuestra aplicación, en este caso, tenemos nuestro fichero XML(videotecas.xml) que ya conocemos, tenemos nuestro fichero JSON(videotecas.json) que se genera en un proceso de nuestra aplicación, que después comentaremos y por último el fichero DTD(videotecas.dtd) que genera nuestro entorno de desarrollo.

En **azul** tenemos todas nuestras clases Java, con sus métodos, que comentaremos.

En **verde** tenemos nuestro fichero pom.xml.

Por último, tenemos en **morado** tenemos la sección de librerías externas, donde se importan aquellas librerías que necesitamos para llevar acabo las tareas que queremos hacer en esta aplicación.

```
public class CrearJson {

    private static final String dir_json = "/Volumes/DATA MAC OSX/andres/Documents/aaUni/Tercer Curso/Primer Cuatrimestre/DIS/d

    public static void crearFicheroJson(ArrayList<Videoteca> videotecas){
        try{
            FileWriter mywriter = new FileWriter(dir_json);
            mywriter.write(new Gson().toJson(videotecas));
            mywriter.close();
        }catch (Exception ex){
            System.out.println(ex);
        }
    }

    public static ArrayList<Videoteca> leerFicheroJson(){
        ArrayList<Videoteca> videotecas = new ArrayList<>();
        Gson gson = new Gson();
        try {
            videotecas = gson.fromJson(new FileReader(dir_json), new TypeToken<ArrayList<Videoteca>>(){}.getType());
        } catch (FileNotFoundException e) {
            e.printStackTrace();
        }

        return videotecas;
    }
}
```

Nuestra primera clase, se llama CrearJson

En ella, como su nombre indica, generamos un fichero JSON, a partir del fichero XML que le pasamos, en nuestro caso, videotecas.xml

Además, también tenemos el método leerFicheroJson() que trae la información del fichero .json a las clases.

Continuar explicación...

La siguiente clase, que podemos apreciar es la clase Actor, que básicamente contiene dos strings (nombre y enlace_wiki), inicializamos los dos strings a NULL.

Después, creamos dos métodos(void) setNombre y setEnlace_wiki que los usamos para asignar a esas variables, los valores que encontremos en nuestro fichero mientras lo vamos leyendo.

Por último, formamos la expresión de cómo queremos que el programa saque la información de los actores, que encontremos en el fichero.

```
public class Actor {  
  
    private String nombre;  
    private String enlace_wiki;  
  
    public Actor(){  
        this.nombre = null;  
        this.enlace_wiki = null;  
    }  
  
    public String getNombre() { return nombre; }  
  
    public String getEnlace_wiki() { return enlace_wiki; }  
  
    public void setNombre(String nombre) { this.nombre = nombre; }  
  
    public void setEnlace_wiki(String enlace_wiki) { this.enlace_wiki = enlace_wiki; }  
  
    public String toString() { return "Nombre Actor: " + this.nombre + "  
Enlace Wikipedia" + this.enlace_wiki + "  
"; }  
}
```

A continuación, vemos otras dos clases muy parecidas, la clase Película y la clase Videoteca

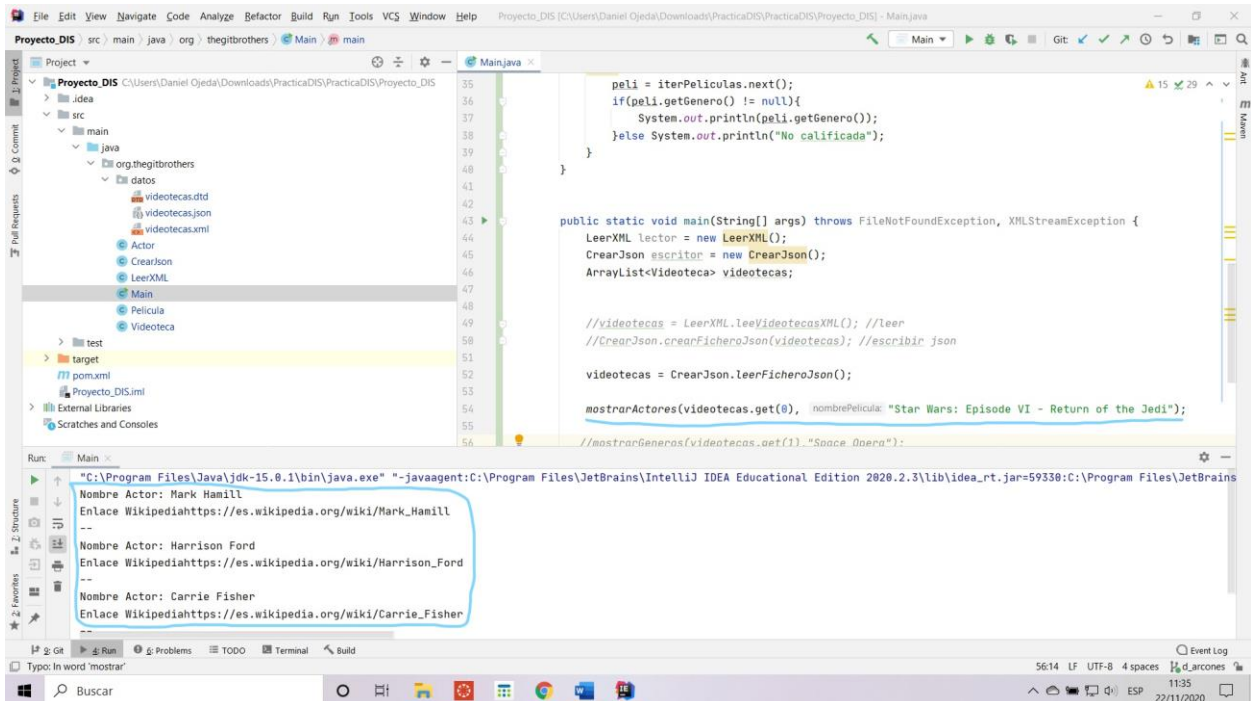
Que se construyen de la misma forma que la de actor, pero adaptándola a los campos que posee Película y Videoteca.

```
public class Videoteca {  
  
    private String nombre;  
    private String ubicacion;  
    private String fecha_ultima_act;  
    private ArrayList<Película> peliculas;  
  
    public Videoteca(){  
        this.nombre=null;  
        this.ubicacion=null;  
        this.fecha_ultima_act=null;  
        this.peliculas= new ArrayList<Película>();  
    }  
  
    public String getNombre() { return nombre; }  
  
    public String getUbicacion() { return ubicacion; }  
  
    public ArrayList<Película> getPeliculas() { return peliculas; }  
  
    public String getFecha_ultima_act() { return fecha_ultima_act; }  
  
    public void setNombre(String nombre) { this.nombre = nombre; }  
  
    public void setUbicacion(String ubicacion) { this.ubicacion = ubicacion; }  
  
    public void setPeliculas(ArrayList<Película> peliculas) { this.peliculas = peliculas; }  
  
    public void setFecha_ultima_act(String fecha_ultima_act) { this.fecha_ultima_act = fecha_ultima_act; }  
}
```

```
public class Pelicula {  
  
    private String titulo;  
    private String sinopsis;  
    private String genero;  
    private String enlace_IMDB;  
    private String id;  
    private String minutos;  
    private String anio_estreno;  
    private ArrayList<Actor> reparto;  
  
    public Pelicula(){  
        this.titulo = null;  
        this.sinopsis = null;  
        this.genero = null;  
        this.enlace_IMDB = null;  
        this.id = null;  
        this.minutos = null;  
        this.anio_estreno = null;  
        this.reparto = new ArrayList<Actor>();  
    }  
  
    public String getId() { return id; }  
  
    public String getMinutos() { return minutos; }  
  
    public String getAnio_estreno() { return anio_estreno; }  
  
    public String getTitulo() { return titulo; }
```

Falta comentar las clases LeerXML y Main.

Al final, cuando hacemos una búsqueda en el Main, obtenemos los resultados que buscamos, por ejemplo...



```
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

    peli = iterPeliculas.next();
    if(peli.getGenero() != null){
        System.out.println(peli.getGenero());
    }else System.out.println("No calificada");
}

public static void main(String[] args) throws FileNotFoundException, XMLStreamException {
    LeerXML lector = new LeerXML();
    CrearJson escritor = new CrearJson();
    ArrayList<Videoteca> videotecas;

    //videotecas = LeerXML.leerVideotecasXML(); //leer
    //CrearJson.crearFicheroJson(videotecas); //escribir json

    videotecas = CrearJson.leerFicheroJson();

    mostrarActores(videotecas.get(0), nombrePelicula: "Star Wars: Episode VI - Return of the Jedi");
    //mostrarGeneros(videotecas.get(1). "Space Opera");
}
```

Run: Main

"C:\Program Files\Java\jdk-15.0.1\bin\java.exe" "-javaagent:C:\Program Files\JetBrains\IntelliJ IDEA Educational Edition 2020.2.3\lib\idea_rt.jar=59330:C:\Program Files\JetBrains\IntelliJ IDEA Educational Edition 2020.2.3\bin" -Dfile.encoding=UTF-8

Nombre Actor: Mark Hamill
Enlace Wikipediahttps://es.wikipedia.org/wiki/Mark_Hamill
--
Nombre Actor: Harrison Ford
Enlace Wikipediahttps://es.wikipedia.org/wiki/Harrison_Ford
--
Nombre Actor: Carrie Fisher
Enlace Wikipediahttps://es.wikipedia.org/wiki/Carrie_Fisher

Aquí vemos, que queremos obtener los actores de la película Star Wars El Retorno del Jedi

Vemos que la aplicación, efectivamente, nos da dichos actores por pantalla.

LeerXML

Hacemos uso de la librería StAx, usando un xmlstreamreader para leer el xml al completo, hacemos uso de un switch y varios if que buscan el localname del xml (etiqueta) y si esta añade dicho elemento a la videoteca, se ha hecho así para que no importe el orden, además se ha usado arraylist para que el sistema sea dinámico y se puedan actualizar infinitas videotecas. Todas las variables se inicializan a null de tal manera que, si no hay género, ese se te quede a null y sea fácil y rápido encontrar aquellas películas sin género.

CrearJson:

Crear json hace uso de la biblioteca de Google, gson. Hacemos uso del método toJson junto con un writer de java para escribir objetos instanciados al fichero json. Para leer, este hacemos uso del método fromJson que también proporciona la biblioteca para recoger todas las videotecas en un array de videotecas para su uso.

En los comentarios de las clases java aparece comentado todo en mucho mas detalle.