PROYECTO DE PROGRAMACIÓN GRÁFICA Y FICHEROS – APLICACIÓN DE LISTAS MUSICALES

Descripción del proyecto

En este proyecto tendrás que desarrollar una **aplicación Java en modo gráfico con Java Swing que permita gestionar listas de reproducción musicales en diferentes formatos**. La descripción completa y detallada de la implementación de este proyecto se encuentra en las siguientes secciones.

Debes aplicar todas las buenas prácticas aprendidas, entre otras:

- Implementar los métodos de las clases realizando control de errores.
- Implementar los constructores y métodos de la clase reutilizando todo el código posible.
- Implementar las clases sin modificar la interfaz pública de la misma.
- Validar todos los datos de entrada, tanto los introducidos por teclado por el usuario como los parámetros de las funciones.
- Evitar código duplicado y reutilizar código mediante funciones.
- Nombrar las variables, clases y métodos de acuerdo a las convenciones del lenguaje, y usar identificadores descriptivos y representativos de lo que almacena o realiza la variable/función.
- Documentar las clases con comentarios Javadoc.
- Seguir las especificaciones descritas en este guion sin cambiarlas, salvo justificación y previa consulta con el profesor.

Importante: No se corregirá ningún proyecto en el que haya errores de compilación.

Se trata de desarrollar una aplicación Java en modo gráfico con ***Java Swing*** que permita gestionar listas de reproducción musicales en tres formatos distintos: M3U, PLS y XSPF.

La aplicación se diseñará de tal forma que se puedan cargar datos de cualquiera de estos tres tipos de ficheros y guardar la información en cualquiera de estos tres formatos de ficheros.

Se implementará la clase ***Canción*** con tres atributos privados (*titulo: String, ruta: String, duracion: int*). Se crearán los métodos y constructores necesarios para manejar esta clase.

Cuando el usuario abra un fichero ***m3u***, ***pls*** o ***xspf***, se deberán cargar los datos en un ***ArrayList<Cancion>*** y mostrarlos en un componente ***JTable***.

La aplicación debe ser capaz de:

- Crear nuevas listas de reproducción.
- Agregar, modificar o eliminar canciones de la lista de reproducción.
- Cargar/Guardar la lista de reproducción en cualquiera de los tres formatos indicados.

El diseño de la interfaz gráfica de la aplicación es libre.

La tarea se califica entre 0 y 10 puntos desglosados de la siguiente manera:

- **Interfaz gráfico (3 puntos)**: Para tener en cuenta este criterio, el programa debe tener la funcionalidad completa de lectura y escritura en al menos un formato de lista de reproducción. Este apartado engloba los siguientes elementos:
    - Diseño, robustez y reusabilidad de componentes (**1 punto**): menús, barra de herramientas, aceleradores, mensajes, validación de datos, etc.
    - Funcionalidad (**2 puntos**): manejo de la lista de canciones en el componente ***JTable*** que permite crear nuevas listas y añadir, eliminar, modificar datos de una lista.
- **Lectura y escritura de listas M3U (2 puntos)**.
- **Lectura y escritura de listas PLS (2 puntos)**.
- **Lectura y escritura de listas XSPF (2 puntos)**.

Descripción del formato de los archivos

Formato M3U

El siguiente cuadro muestra un ejemplo de fichero M3U:

```
#EXTM3U
#EXTINF:315,David Bowie - Space Oddity
Music/David Bowie/Singles 1/01-Space Oddity.ogg
#EXTINF:-1,David Bowie - Changes
Music/David Bowie/Singles 1/02-Changes.ogg
...
#EXTINF:251,David Bowie - Day In Day Out
Music/David Bowie/Singles 2/18-Day In Day Out.ogg
```

El fichero comienza con el literal #EXTM3U. Cada canción se representa por dos líneas, la primera línea comienza con el literal #EXTINF: y es seguida por la duración de la canción en segundos, una coma y el nombre de la canción. Una duración de -1 significa que la longitud es desconocida (en ambos formatos). La segunda línea es la ruta al fichero donde se encuentra la canción.

Formato PLS

A continuación se muestra el fichero equivalente en formato PLS:

```
[playlist]
File1=Music/David Bowie/Singles 1/01-Space Oddity.ogg
Title1=David Bowie - Space Oddity
Length1=315
File2=Music/David Bowie/Singles 1/02-Changes.ogg
Title2=David Bowie - Changes
Length2=-1
...
File33=Music/David Bowie/Singles 2/18-Day In Day Out.ogg
Title33=David Bowie - Day In Day Out
Length33=251
NumberOfEntries=33
Version=2
```

El fichero comienza con el literal [playlist] . Cada canción se representa por tres entradas clave-valor: nombre del fichero, titulo y duración. El fichero finaliza con dos líneas de metadatos.

Formato XSPF (XML Shareable Playlist Format)

Se debe dar soporte a las características principales de la versión 1 de **XSPF**.

Más información sobre estos formatos:
http://es.wikipedia.org/wiki/M3U
http://es.wikipedia.org/wiki/PLS
http://es.wikipedia.org/wiki/XML_Shareable_Playlist_Format
http://www.xspf.org/

Entrega de la tarea

El proyecto se desarrollará con el IDE Netbeans. El nombre del proyecto y de las clases deben llamarse de la forma indicada en la descripción de la tarea. Se comprimirá la carpeta del proyecto Netbeans en un único fichero en formato .ZIP o 7z que se subirá al buzón de la tarea en la plataforma Moodle.

El archivo se nombrará siguiendo las siguientes pautas:

Apellido1_Apellido2_Nombre_PROG_ProyectoListasMusicales.zip

Resultados de aprendizaje y criterios de evaluación relacionados

En esta actividad se evalúan los siguientes resultados de aprendizaje con los criterios de evaluación que se relacionan para cada uno de ellos:

- RA3. Escribe y depura código, analizando y utilizando las estructuras de control del lenguaje.
  - a) Se ha escrito y probado código que haga uso de estructuras de selección.
  - b) Se han utilizado estructuras de repetición.
  - c) Se han reconocido las posibilidades de las sentencias de salto.
  - d) Se ha escrito código utilizando control de excepciones.
  - e) Se han creado programas ejecutables utilizando diferentes estructuras de control.
  - f) Se han probado y depurado los programas.
  - g) Se ha comentado y documentado el código.
- RA 4. Desarrolla programas organizados en clases analizando y aplicando los principios de la programación orientada a objetos.
  - a) Se ha reconocido la sintaxis, estructura y componentes típicos de una clase.
  - b) Se han definido clases.
  - c) Se han definido propiedades y métodos.

d) Se han creado constructores.

e) Se han desarrollado programas que instancien y utilicen objetos de las clases creadas anteriormente.

f) Se han utilizado mecanismos para controlar la visibilidad de las clases y de sus miembros.

h) Se han creado y utilizado métodos estáticos.

- RA 5. Realiza operaciones de entrada y salida de información, utilizando procedimientos específicos del lenguaje y librerías de clases.

  b) Se han aplicado formatos en la visualización de la información.

  c) Se han reconocido las posibilidades de entrada / salida del lenguaje y las librerías asociadas.

  d) Se han utilizado ficheros para almacenar y recuperar información.

  e) Se han creado programas que utilicen diversos métodos de acceso al contenido de los ficheros.

  f) Se han utilizado las herramientas del entorno de desarrollo para crear interfaces gráficos de usuario simples.

  g) Se han programado controladores de eventos.

  h) Se han escrito programas que utilicen interfaces gráficos para la entrada y salida de información.

- RA 6. Escribe programas que manipulen información seleccionando y utilizando tipos avanzados de datos.

  c) Se han utilizado listas para almacenar y procesar información.

  d) Se han utilizado iteradores para recorrer los elementos de las listas.

  h) Se han identificado las clases relacionadas con el tratamiento de documentos XML.

  i) Se han realizado programas que realicen manipulaciones sobre documentos XML.

Rúbrica de evaluación

Cada apartado puntuable del proyecto se valorará con la siguiente rúbrica.

#	Criterio	Porcentaje
1	El programa/función implementado no cumple los requisitos, no soluciona de forma algorítmica el ejercicio o las soluciones obtenidas por el programa no son las esperadas, el código no compila o contiene errores.	0%
2	El programa/función implementado soluciona de forma algorítmica el ejercicio pero falla con datos de entrada no permitidos.	25%
3	El programa/función implementado tiene fallos inesperados en situaciones específicas o concretas, es decir, falla para un determinado caso o valor de entrada, pero en general el resultado obtenido es válido.	50%
4	El programa/función implementado cumple los requerimientos pero: • El código no es legible o no está bien estructurado.	75%
5	El programa/función implementado se ajusta perfectamente a la especificación: • Se validan los datos de entrada. • El resultado obtenido es válido para cualquier dato de entrada. • El código es modular y se emplean funciones/métodos adecuadamente. • El código es legible y usa comentarios relevantes y/o Javadoc.	100%