Relación de Ejercicios de Clases y Objetos (8): Base de Datos de Canciones

En esta relación, usaremos la base de datos MySQL/MariaDB. Conectaremos desde nuestro programa con una base de datos y mandaremos consultas para insertar información en ella o sacar los listados que nos interesen.

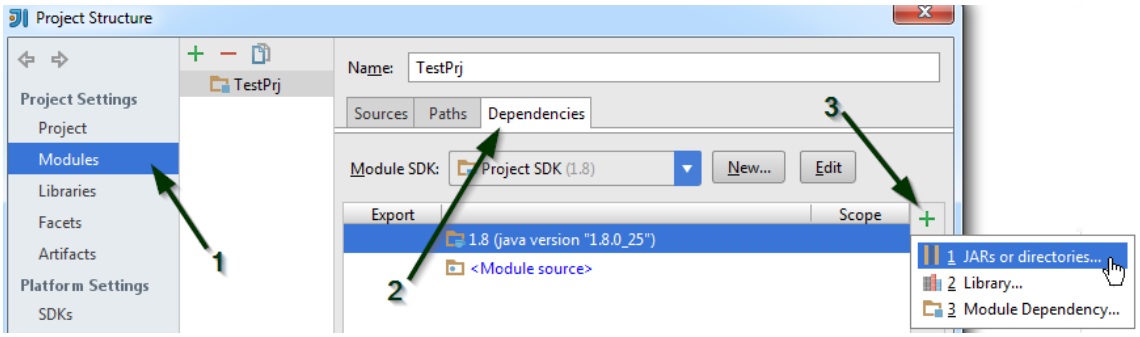
Para ello, antes que nada, tendremos que descargarnos una biblioteca que nos permite el acceso a este tipo de base de datos, que encontraréis en este enlace (coged la versión más reciente):

https://dev.mysql.com/downloads/connector/j/

También podéis encontrarla en nuestro Moodle dentro de una carpeta con recursos para esta relación, con el nombre: **mysql-connector-java-(versión).jdbc**

Una vez descargada, la añadiremos a nuestro proyecto siguiendo estas instrucciones:

- Menú File -> Project Structure
- En la sección Modules, pestaña Dependencies, pulsamos el (+) y le damos a JARs.
- Seleccionamos el fichero .jar que nos hemos bajado y lo añadimos al proyecto.



Una vez añadida esta biblioteca a nuestro proyecto, necesitaremos instalar el motor de base de datos MySQL en algún equipo. Como ya tendréis otros sistemas de bases de datos instalados y es un poco engorroso instalar otro más, os he preparado una máquina virtual que tiene el servidor instalado ya dentro. Para conectarnos a ella, simplemente tendremos que poner sus datos de conexión al crear nuestro *Connection*:

```
Connection conn = DriverManager.getConnection("jdbc:mysql://<IP de la Máquina Virtual>:3306/musica", "usuario", "usuario");
```

Nuestro programa nos servirá para clasificar nuestra colección de MP3s de manera semiautomática. Lo que hará será buscar en la carpeta que le digamos todos los ficheros MP3s e irlos añadiendo a la base de datos. Intentará coger de los ficheros el nombre de la canción y, en su caso, el nombre del artista y el resto de los datos se los pedirá al usuario.

Una vez añadida la información en la base de datos, nos permitirá hacer búsquedas y nos generará listas de reproducción de manera automática.

Para nuestro programa, necesitaremos escribir la siguiente clase

- **MusicaDB**
  - Atributos
    - **conn**, de tipo Connection (`java.sql.Connection`)
  - Constructor
    - Al constructor le pasaremos un 4 *String* que necesitaremos para conectar con nuestra base de datos:
      - IP del servidor
      - Nombre de la base de datos
      - Usuario
      - Contraseña

      Con estos datos, intentará conectar y guardará en **conn** la conexión ya establecida.
  - Métodos
    - *int* **insertaCancion**(*String* ruta, *String* titulo, *int* idArtista, *int* idEstilo, *int* anno, *int* estrellas), nos inserta una canción en la base de datos con los datos proporcionados. El método devolverá un *int* que será el id de la película que insertamos.
      Para esto, al ejecutar la consulta le pasaremos un parámetro especial:

```
ResultSet rs = st.executeUpdate(sql, Statement.RETURN_GENERATED_KEYS);
```

      Al lanzar un "INSERT" de esta forma, nos devolverá un *ResultSet* con el valor del id autoincrementado, de manera similar a como se hacía en SQLite (que requería una consulta adicional).
    - *int* **insertaArtista**(*String* nombre) nos inserta el artista en la base de datos y nos devuelve un *int* con el id del artista.
    - *int* **insertaEstilo**(*String* nombre) nos inserta un estilo en la base de datos y nos devuelve un *int* con el id del estilo.
    - *int* **buscaArtista**(*String* nombre) nos busca el artista en la base de datos y nos devuelve su id (o -1 si no lo encontramos).
    - *int* **buscaEstilo**(*String* nombre) nos busca el estilo en la base de datos y nos devuelve su id (o -1 si no la encontramos).
    - *boolean* **existeFichero**(*String* ruta) buscará si hay alguna canción correspondiente al fichero que le pasamos.
    - *List<Cancion>* **listadoCanciones**() nos devuelve un listado de todas las canciones que hay en la base de datos.
    - *List<Cancion>* **listadoCancionesPorArtista**(*int* idArtista) nos devuelve un listado de todas las películas del artista que le pasamos por parámetro que hay en la base de datos.
    - *List<Cancion>* **listadoCancionesPorEstilo**(*int* idEstilo) nos devuelve un listado de todas las películas del estilo que le pasamos por parámetro que hay en la base de datos.
    - *List<Cancion>* **listadoCancionesPorAnno**(*int* annoInicio, *int* annoFinal) nos devuelve un listado de todas las películas cuyo año esté entre los dos que le pasamos por parámetro.

        - *List<Cancion>* **listadoCancionesPorEstrellas**(*int* minimo, *int* maximo) nos devuelve un listado de todas las películas cuya valoración esté entre los valores que le pasamos (valores entre 1 y 5).

Para estos últimos métodos de la clase MusicaDB, necesitaremos una clase que contenga los datos de una canción. Como sólo la vamos a usar para esto, será simplemente un registro:

```
public class Cancion
{
    public int id;
    public String ruta;
    public String titulo;
    public int idArtista;
    public int idEstilo;
    public int anno;
    public int estrellas;
}
```

Esta primera clase nos facilitará el acceso a la base de datos. A continuación, escribiremos otra clase que se encargará de buscar los archivos MP3 en nuestro disco y de preguntarnos los datos de las canciones que guardaremos en la base de datos.

- **ClasificaMP3**
    - Atributo
        - **db** de tipo *MusicaDB*
    - Constructor
        - Al constructor le pasaremos la base de datos ya inicializada, que guardará en el atributo para que la puedan usar los diferentes métodos. La sintaxis será:
          ```
          MusicaDB db = new MusicaDB( ... );
          ClasificaMP3 cla = new ClasificaMP3(db);
          ```
    - Métodos
        - **nuevaCancion**(*String* ruta): le pasamos un *String* con la ruta completa del fichero MP3.
          Antes que nada, se deberá comprobar si el archivo ya está en la base de datos. En ese caso, se ignorará para no meterlo dos veces.
          Después intentaremos sacar del nombre del archivo algunos datos, como el título de la canción y el artista. Reconocerá los siguientes formatos:
          ```
          David Bisbal - Bulería.mp3
          Bulería.mp3
          ```
          Si hay un guion, separará el nombre del artista y de la canción. En caso contrario, cogerá el nombre del archivo como nombre de la canción.
          Después, se le preguntarán al usuario todos los datos de la canción (titulo, artista, estilo, año y estrellas), tomando como valores por defecto los que hayamos sacado del archivo.
          Si el artista o el estilo no se encuentran en la base de datos, le daremos al usuario la posibilidad de insertar uno nuevo.
        - **buscaCanciones**(*String* directorio): buscará todas las canciones MP3 que haya en una carpeta y las procesará usando el método *nuevaCancion*.

- **buscaCancionesRecursivo**(*String* directorio): buscará todas las canciones MP3 que haya en una carpeta incluyendo todas las subcarpetas que tenga dentro.
- **generaListaReproduccion**(*String* fichero): nos generará una lista de reproducción en el fichero que le pasamos por parámetro. La lista de reproducción será un fichero de texto con el formato M3U.
- **escribeFicheroM3U**(*String* fichero*, List<Canciones>* lista): este método será privado y lo usarán el resto de las funciones que generan listas de reproducción para crear el fichero M3U (y así no repetir código).
  El formato del fichero M3U (https://en.wikipedia.org/wiki/M3U) será:

  ```
  #EXTM3U

  #EXTINF:0, Titulo
  C:\Documents and Settings\I\My Music\Fichero.mp3
  ```

  Donde 0 sería el número de segundos (que dejaremos en 0 porque no los sabemos), el artista y el título los cogemos de la base de datos y luego pondremos en otra línea el nombre del fichero (y así todos los ficheros MP3 que queramos).
- **generaListaReproduccionAleatoria**(*String* fichero): nos generará la misma lista pero con los ficheros desordenados aleatoriamente.
- *boolean* **generaListaReproduccionArtista**(String fichero, String artista, *boolean* aleatorio): nos generará una lista con todas las canciones de un artista. Si el artista no existe, no creará el fichero y nos devolverá *false.*
- *boolean* **generaListaReproduccionEstilo**(*String* fichero, *String* estilo, *boolean* aleatorio): nos generará una lista con todas las canciones del estilo que le digamos. Si el estilo no existe, no creará el fichero y nos devolverá *false.*
- **generaListaReproduccionEstrellas**(*String* fichero, *int* minimo*, int* maximo, *boolean* aleatorio): nos generará una lista con todas las canciones cuya valoración esté entre los valores que le indicamos.

Una vez montada toda nuestra estructura de clases, tendremos que escribir el programa principal **main**, que simplemente constará de un menú desde el que se podrá llamar a las diferentes funciones de la clase ClasificaMP3.