Programming Coursework Specification

Introduction

In your role as a software developer, you have been contracted by QUB's Music Department to create a Java application capable of managing digital music (or sound) files. Specifically, you are required to build and test a prototype system that meets the deliverables outlined in the sections below.

Part 1: Core Functionality

In order to demonstrate that the proposed Java application is feasible, the following features must be implemented.

1.1 Implementation of object classes

Implement the object classes, enumerator and interface shown below.

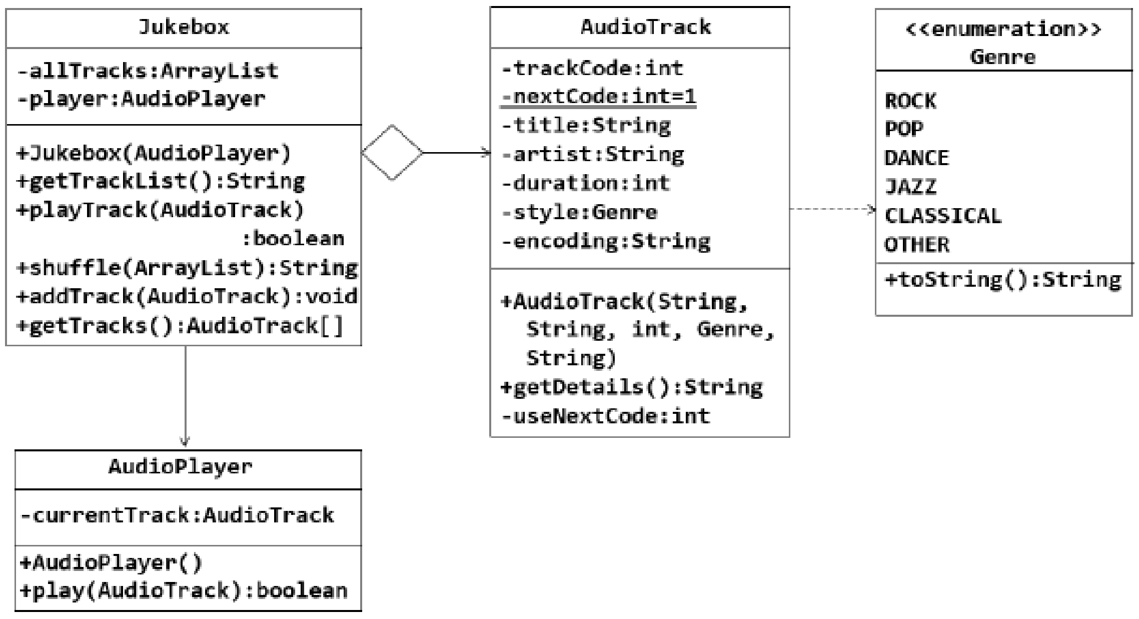

Figure 1: A (partial) UML Class Diagram for QUB Music Management Software.

1.2 Implement a console application

A console (menu-based) application to manage (through a Jukebox instance) the interaction with AudioTrack instances within the system. The following menu options should be provided:

- Display All Tracks: the user should be able to view summary details (trackCode, title, artist, duration & style) of all audio tracks in the system, listed one after another and ordered by title.
- Display Tracks by Artist: the user should be able to view a list of track titles for a user-specified artist.
- Add New Track: the user should be able to create and add a new audio track to the Jukebox instance. A track instance can't be added more than once.
- Play a Track: the user should be able to select and play a specific track.
- Create a Playlist: the user should be able to create and add tracks to a playlist.

- Shuffle Play: the user should be able to play a playlist
- Display the Top 10: user should be able to view a list of the top 10 track (names) in order of number of plays.
- Exit: the user should be able to exit the system.

Notes

A. The console application should create a Jukebox instance using an AudioPlayer instance as constructor parameter.

B. The console application should create and add a number (at least 5) audio tracks with the Jukebox instance for testing purposes.

C. When adding a new audio track, the user should be prompted for the title, artist, duration, style (Genre) and encoding. The encoding value (String) must be one of “mp3” or “wav”.

D. When displaying tracks for a specific artist, the user should be presented with a list of artist names to select from.

E. When playing a track, the user should be presented with the full list of available tracks to select from. A track is played by calling the playTrack method for a Jukebox, which in turn should invoke the play method for the associated AudioPlayer, which should return true if the supplied parameter is a valid AudioTrack and false otherwise. The console should report whether or not a selected track (name, artist & duration) been played, through use of a suitable message.

F. When creating a playlist, the user should be (repeatedly) presented with a full list of tracks to choose from. This should be implemented using an ArrayList, (representing a single playlist) maintained within the main application. This list should be cleared each time a user wants to create a playlist. An audio track can’t be in the playlist more than once.

G. When a user selects shuffle play, the current (one and only) playlist should be used as a parameter to the shuffle method defined as part of Jukebox. The shuffle method (Jukebox) should play each track in the supplied playlist only once and in random order. This method should return a String containing the order of play (name and artist) or play errors (if they occur) for each track – this String should be displayed on the console, alongside suitable messages.

H. For enumeration Genre, the toString() method should return a String corresponding to the current value:

Value	*String*
ROCK	“Rock and Roll”
POP	“Easy Listening Pop”
DANCE	“Techno Dance”
JAZZ	“Smooth Jazz”
CLASSICAL	“Classical”
OTHER	“Unknown Genre”

I. You should make use of the Menu class defined in Menu.java (from the assessment specification on Canvas).

J. No external classes/libraries other than Random, Scanner or ArrayList may be used. When using ArrayList, you are restricted to the get and add methods only.

K. You may not add additional public methods or attributes to Jukebox. You may add private attributes, accessors and mutators to classes AudioTrack and AudioPlayer. However, these must be justified (use comments).

Part 2: Additional Features

A – Adding New Player Classes

1. Copy the application and Menu classes from part01 to part02 – no other files should be copied. Use import statements to reference definitions in part01.
2. Create two new classes, MP3Player and WavPlayer which extend the behaviour of AudioPlayer.
3. For class MP3Player, override the play method, which should return true if the encoding of the supplied AudioTrack is “mp3” and false otherwise.
4. For class WavPlayer, override the play method, which should return true if the encoding of the supplied AudioTrack is “wav” and false otherwise.
5. In the console application, on initialisation, allow the user to construct the Jukebox with either an instance of MP3Player or an instance of WavPlayer. Attempts to play tracks on an AudioPlayer of the wrong (encoding) type should now be rejected.

B – Data Storage

1. Add behaviour to the console application to enable the state of a Jukebox instance to be automatically stored/retrieved to/from a single (CSV) file.
2. Data should be restored when the console application starts and stored when the console application is exited.
3. A sample data file should be provided as part of your submission – avoid using absolute file paths.

To help QUB review any additional features implemented, include a short Microsoft Word document with your submission that provides a title and short description (max 2-3 sentences) for each feature.

Part 3 – Testing the Application

As part of QUBs quality assurance procedure, you are required to submit evidence that the application has been thoroughly tested. This should take the form of a completed testing document. The scope of this testing should cover all areas of core functionality outlined in Part 1 of this assignment specification. Testing documentation must be submitted using the template provided by QUB, available on Canvas.

Class Information

Jukebox Attributes and Methods

- **allTracks** – list of AudioTrack objects managed by a Jukebox
- **player** – reference to an AudioPlayer used by the Jukebox (to play AudioTrack) music
- **constructor parameter** – reference to an AudioPlayer
- **getTrackList()** – should return a String containing a comma-separated list of trackCode and title for all available tracks
- (e.g. “1, track name, 2 another track name, 3, yet another, ….”) or null if there are no tracks
- **playTrack()** – parameter is the track to be played (via the associated AudioPlayer)
- **shuffle()** – parameter is a list of tracks to be played in random order (via the associated AudioPlayer)
- **addTrack()** – parameter is the track to add to Jukebox
- **getTracks()** – returns an array of all tracks within Jukebox

AudioPlayer Attributes and Methods

- **currentTrack** – last track played
- **play()** – parameter is the track to be played, return false if it can't be played. By default, a track is playable if the parameter references an AudioTrack object.

AudioTrack Attributes and Methods

- **trackCode –** a unique track identifier
- **nextCode** – next usable track identifier
- **title** – track title
- **artist** – track artist
- **duration** – duration (in seconds) of the track
- **style** – track genre
- **encoding** – must be either “mp3” or “wav”
- **constructor** – parameters: title, artist, duration, style, encoding in that order
- **getDetails()**- returns a formatted string including all details on a track